z/OS Communications Server

SNA Programming

Version 2 Release 1

SC27-3674-00

Note:
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 903.

First Edition (September 2013)

This edition applies to version 2, release 1, modification 0 of z/OS (5650-ZOS), and to subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You may send your comments to the following address.

International Business Machines Corporation

Attn: z/OS Communications Server Information Development

Department AKCA, Building 501

P.O. Box 12195, 3039 Cornwallis Road

Research Triangle Park, North Carolina 27709-2195

You can send us comments electronically by using one of the following methods:

Fax (USA and Canada):
1+919-254-1258

Send the fax to "Attn: z/OS Communications Server Information Development"

Internet email:
comsvrcf@us.ibm.com

World Wide Web:
http://www.ibm.com/systems/z/os/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number. Make sure to include the following information in your comment or note:

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

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

© Copyright IBM Corporation 2000, 2013.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures . . . . . **xiii**

Tables . . . . . **xvii**

About this document . . . . . **xxi**
Who should read this document xxi
How this document is organized xxi
How to use this document xxiii
Determining whether a publication is current xxiii
How to contact IBM service xxiv
Conventions and terminology that are used in this document xxiv
How to read a syntax diagram xxv
Prerequisite and related information xxix

Summary of changes . . . . . **xxxv**

Chapter 1. VTAM application program concepts **1**
Systems Network Architecture (SNA) concepts in VTAM 1
Network accessible units 1
Sessions 3
Major programming elements in a VTAM application program 7
Opening the program 10
Establishing a session with an LU 11
Receiving requests from LUs 11
Sending a request 12
Sending a response 12
Receiving a response 13
Other exit routines 13
Terminating a session with an LU 13
Closing the program 13
Constants and control blocks 14
VTAM macroinstructions 14
VTAM application program as part of an SNA network 14
VTAM application program 15
Processing part 16
Communication part 16
VTAM part 16
Network control program 16
Logical unit 17
Terminal operator and devices 17
Another VTAM application program 17
Using a VTAM application program to manage the network 18

Chapter 2. VTAM language **19**
Characteristics of the language 19
Keyword operands 19
Manipulative macroinstructions 19
Exit routines 19
Summary description of the VTAM macroinstructions 20
Declarative macroinstructions 21
Manipulative macroinstructions 22
ACB-based macroinstructions 22
RPL-based macroinstructions 23
Relationship between the executable macroinstructions and control blocks 26

© Copyright IBM Corp. 2000, 2013

Opening the application program . . . 27
Establishing sessions with LUs . . . 27
Communicating with LUs . . . 28
Terminating sessions with LUs . . . 28
Closing the application program . . . 28
Exit routines . . . 28
Normal operating system environment for a VTAM application program . . . 30
Use of a single task . . . 30
Mainline program . . . 31
Inline exit routines . . . 31
Asynchronous exit routines . . . 31
Dispatching priorities . . . 32

Chapter 3. Organizing an application program . . . 33
Coding guidelines for application programs . . . 33
Program structure recommendations . . . 33
Simplifying migration and network upgrades . . . 35
Single-thread or multithread operations . . . 36
Using a single-thread program . . . 36
Using a multithread program . . . 36
Multithreading facilities . . . 37
USERFLD field of the NIB . . . 37
Scheduling output . . . 38
Receiving input on any session except those already in communication . . . 39
How a synchronous operation works . . . 40
How an asynchronous operation works . . . 41
Using ECBs . . . 41
Using RPL exit routines . . . 42
Initializing a session . . . 44
Advantages and disadvantages of different forms of operation . . . 44
Some questions that affect program organization . . . 45
Application programming interface . . . 50
Handling control blocks and work areas . . . 51
Techniques for handling control blocks and work areas . . . 52

Chapter 4. Opening and closing an application program . . . 55
Opening an application program . . . 55
Access method control block (ACB) . . . 56
OPEN macroinstruction . . . 57
Vector lists . . . 58
Supplying control vectors with the SETLOGON START . . . 64
Using multiple ACBs in a VTAM application program . . . 65
Using network-qualified names . . . 65
Where OPEN can be issued . . . 66
Using persistent LU-LU session support . . . 66
How an application establishes persistence . . . 67
Session recovery states . . . 67
Application program recovery with single-node persistence enabled . . . 67
Application program recovery with multinode persistence enabled . . . 69
Closing an application program . . . 73
Program initiates closing . . . 74
Program receives a closedown message . . . 74
TPEND exit routine is entered . . . 74
Opening and closing an application program as a generic resource . . . 78
Identifying an application program as a generic resource member . . . 78
Initiating a session using a generic resource name . . . 79
Initiating a session using the name of an application that is a member of a generic resource . . . 79
Closing an application program that is a member of a generic resource . . . 79

Chapter 5. Establishing and terminating sessions with logical units . . . 81

Defining LUs . . . 81
Stages of session establishment . . . 83
Stages of session termination . . . 84
Sources of SNA Initiate and Terminate requests . . . 85
Macroinstructions related to session establishment and termination . . . 87
SIMLOGON macroinstruction . . . 87
OPNDST macroinstruction . . . 89
CLSDST macroinstruction . . . 92
REQSESS macroinstruction . . . 95
OPNSEC macroinstruction . . . 96
SESSIONC macroinstruction with CONTROL=BIND . . . 98
TERMSESS macroinstruction. . . . 99
Exit routines related to session establishment and termination . . . 100
LOGON exit routine . . . 101
SCIP exit routine . . . 102
NSEXIT exit routine . . . 104
LOSTERM exit routine . . . 106
Summary tables of exit routines . . . 107
Session outage notification . . . 110
Queuing a request for a session with an SLU . . . 110
Control blocks used for session establishment and termination . . . 114
Request parameter list (RPL) . . . 114
Node initialization block (NIB) . . . 115
Application-supplied dial parameter control block (ASDP) . . . 119
Establishing parameters for sessions . . . 122
General pattern of agreement on session parameters . . . 122
Defining and naming a set of session parameters (logon mode and class of service) . . . 123
How logon mode names and session parameters are used . . . 123
NIB LOGMODE and BNDAREA operands . . . 124
Examples of how an application program processes session parameters . . . 127
Establishing cryptographic sessions . . . 131
Establishing single-domain cryptographic sessions . . . 132
Establishing cross-domain cryptographic sessions . . . 133
How VTAM determines the level of cryptography for a cryptographic session. . . . 133
Restoring sessions pending recovery. . . . 137
Data tracking . . . 137
Restoring sessions . . . 139
Extended recovery facility (XRF) programming . . . 149

Chapter 6. Communicating with logical units . . . 151
Who is communicating: The VTAM application program and LUs. . . . 151
What is communicated: Requests and responses . . . 151
What a request contains . . . 152
What a response contains . . . 153
How requests and responses are exchanged . . . 157
SEND, RECEIVE, and SESSIONC macroinstructions . . . 157
Normal-flow and expedited-flow requests and responses . . . 158
Sequence numbers. . . . 163
Controlling flow . . . 164
Identifying LUs and sessions . . . 168
Using VTAM to communicate with LUs . . . 168

Chapter 7. Using exit routines . . . 219
How exit routines work . . . 219
RPL exit routines . . . 219
EXLST exit routines . . . 221
Summary of exit routines . . . 224
Deciding whether and how to use exit routines . . . 228
Specifying the DFASY, RESP, and SCIP exit routines in an ACB or NIB . . . 229
Special requirements for LERAD and SYNAD exit routines . . . 230

Exit scheduling versus ECB posting . . . 231
Procedures to follow in writing exit routines . . . 231
Entry procedures . . . 234
Cautions, restrictions and techniques . . . 235
Exit procedures . . . 235
DFASY exit routine . . . 236
LERAD exit routine . . . 237
LOGON exit routine . . . 238
TSO/VTAM Katakana and double-byte character set (DBCS) support . . . 241
LOSTERM exit routine . . . 241
LOSTERM reason codes . . . 242
NSEXIT exit routine . . . 244
Network services procedure error or Notify . . . 245
Cleanup session . . . 252
RELREQ exit routine . . . 253
RESP exit routine . . . 254
RPL exit routine . . . 256
SCIP exit routine . . . 256
Clear . . . 257
Start Data Traffic (SDT) . . . 257
Request Recovery (RQR) . . . 257
Set and Test Sequence Numbers (STSN) . . . 258
BIND . . . 258
UNBIND . . . 260
SYNAD exit routine . . . 263
TPEND exit routine . . . 263

Chapter 8. Setting and testing control blocks and macro global variables . . . 267
Setting and testing control block values . . . 267
Using manipulative macroinstructions . . . 267
GENCB macroinstruction . . . 268
MODCB macroinstruction . . . 269
SHOWCB macroinstruction . . . 270
TESTCB macroinstruction . . . 271
Using INQUIRE OPTCD=TERMS to generate NIBs . . . 271
Using DSECT-creating assembler instructions and macroinstructions . . . 272
Defining the DSECTs . . . 272
Using the DSECTs . . . 273
ISTGLBAL macroinstruction . . . 274
Release-level and component-ID macro global variables . . . 274
Function-list macro global variables . . . 275

Chapter 9. Handling errors and special conditions . . . 277
OPEN and CLOSE errors and special conditions . . . 277
Manipulative macroinstruction errors and special conditions . . . 278
RPL-based macroinstruction errors and special conditions . . . 279
Coding LERAD and SYNAD exit routines . . . 290
Handling exception conditions (register 0=04) . . . 291
Handling retriable completion (register 0=08) . . . 293
Handling data integrity damage (register 0=0C) . . . 293
Handling environment errors (register 0=10) . . . 293
Handling logic errors (register 0=14 and register 0=18) . . . 294

Chapter 10. Operating system facilities . . . 295
VTAM macroinstruction differences across operating systems . . . 295
Assigning operating system authorization . . . 296
Authorization criteria . . . 296
Multitasking . . . 296
Separating data communication activity from other activity . . . 297
Dividing data communication activity among several tasks . . . 297

Using multiple ACBs within one task . . . 299
Authorized path . . . 300
Specifying authorized path macroinstructions . . . 301
Additional coding considerations for authorized path . . . 302
Simple example of authorized path usage . . . 304
Authorized asynchronous exit routines . . . 307
Execution of exit routines . . . 307
EXLST exit routines other than LERAD and SYNAD . . . 307
LERAD and SYNAD exit routines . . . 308
RPL exit routines . . . 309
Serialization of execution . . . 309
Task association . . . 310
Exit routine task association . . . 310
Macroinstruction task association . . . 311
Multiple address spaces . . . 312
Types of address spaces . . . 312
Rules for coding macroinstructions and exit routines . . . 316
Address space used for exit routine execution . . . 316
Cross-memory application program interface (API) support . . . 317
31-bit addressing . . . 319
Opening by the Application Program . . . 320
Specifying macroinstructions . . . 320
Executing exit routines . . . 321
Closing by the Application Program . . . 321
Error handling . . . 322
Isolation of errors . . . 322

Chapter 11. Programming for the IBM 3270 Information Display System . . . 327
Types of 3270 terminals . . . 327
Characteristics of LU type 0 for 3270 terminals . . . 328
Data stream . . . 328
Data flow control . . . 329
Transmission control . . . 331
Exception conditions and sense information . . . 332
Session parameter . . . 333
Device characteristics field . . . 334
Logon message . . . 335
Logoff . . . 335
Test request . . . 335
Summary of differences among LU type 0 3270 terminals . . . 335

Chapter 12. Coding for the communication network management interface . . . 337
CNM interface . . . 337
Functions of the application program . . . 337
Gathering maintenance-related information from a PU . . . 338
Gathering session data from VTAM and NCP . . . 338
Gathering performance data from VTAM . . . 338
Loading a PU . . . 338
Receiving hardware alerts . . . 338
Request unit flow . . . 339
Application program coding requirements for the CNM interface . . . 340
CNM interface requests and responses . . . 341
Protocols and procedures . . . 342
Request unit (RU) formats . . . 342
Inquiry data . . . 352
Constant values . . . 353
Reply data . . . 354
Requirements for receiving session-awareness and trace data . . . 355
VTAM definition requirements . . . 355
Interfaces and interactions . . . 355

Session-awareness data buffer . . . 356
Trace data buffer . . . 357
Requirements for receiving performance data . . . 358
Performance monitor definition requirements for initialization . . . 358
Data collection mechanism . . . 358
Automatic data delivery . . . 360
Data collection dynamics . . . 362
Performance monitor interface termination . . . 363
Performance data types . . . 363
Implications of the multiple monitor environment . . . 363
Request unit formats for the performance monitor interface . . . 364
Sense codes for the performance monitor CNM RUs . . . 368

Chapter 13. Conventions and descriptions of VTAM macroinstructions . . . 371
How the macroinstructions are described . . . 371
How the macroinstructions are coded . . . 371
Name . . . 372
Operation . . . 372
Operands . . . 372
Types of operands . . . 372
Comments and continuation lines . . . 373
Description of the VTAM macroinstructions . . . 375
ACB—Create an access method control block . . . 375
CHANGE—Terminate affinity between LU and generic resource application . . . 384
CHECK—Check request status . . . 387
CLOSE—Close one or more ACBs . . . 390
CLSDST—Terminate sessions, application program is the PLU . . . 394
Network-qualified names with CLSDST . . . 402
EXECRPL—Execute a request . . . 403
EXLST—Create an exit list . . . 404
GENCB—Generate a control block . . . 407
INQUIRE—Obtain logical unit information or application program status . . . 412
INTRPRET—Interpret an input sequence . . . 426
ISTGLBAL—Declare and set macro global variables . . . 430
MODCB—Modify the contents of control block fields . . . 431
NIB—Create a node initialization block . . . 433
NIB fields set by VTAM . . . 442
OPEN—Open one or more ACBs . . . 444
OPNDST—Establish sessions (application as PLU) or recover sessions . . . 453
OPNSEC—Establish a session, application program acts as the SLU . . . 460
RCVCMD—Receive a message from VTAM . . . 463
RECEIVE—Receive input on a session . . . 466
REQSESS—Initiate a session, application program acts as the SLU . . . 477
RESETSR—Cancel RECEIVE operations and switch a session's CA-CS mode . . . 481
RPL—Create a request parameter list . . . 487
RPL fields set by VTAM . . . 505
RPL fields and RPL-based macroinstructions . . . 510
SEND—Send output on a session . . . 514
SENDCMD—Send a VTAM operator command to VTAM . . . 529
SESSIONC—Send a session-control request or response . . . 532
Send a Start Data-Traffic request to the SLU . . . 540
SETLOGON—Modify an application program's capability to establish sessions . . . 543
SHOWCB—Extract the contents of control block fields . . . 551
Control block fields applicable for SHOWCB . . . 553
SIMLOGON—Initiate a session, application program acts as the PLU . . . 555
TERMSESS—Request session termination, application program is SLU . . . 560
TESTCB—Test the contents of a control block field . . . 565
Control block fields applicable for TESTCB . . . 568

Chapter 14. Logic of a simple application program . . . 573

Logic of Sample Program 1 573

Chapter 15. Sample code of a simple application program 579
What SAMP1 does 579
How SAMP1 relates to Sample Program 1 (Chapter 14) 579
Data interface between SAMP1 and LUs 580
Notes on SAMP1 582
Mainline program 582
LOGON exit routine 583
RESP exit routine 584
LERAD and SYNAD exit routines 584
LOSTERM exit routine 585
SAMP1 585

Chapter 16. Logic of a more complicated application program 605
Introduction 605
Organization and flow of Sample Program 2 607
Logic of the 3600 finance communication system I/O routine 614
Logic of the 3600 chaining output routine 617
Routine logic of the 3270 I/O routine 619
Logic of the RESP exit routine 621
Logic of the DFASY exit routine of Sample Program 2 624

Chapter 17. Sample code using authorized path 627
Notes on SAMP3 627
SAMP3 assembler language code 628

Appendix A. Summary of control block field usage 633
ACB 633
CHANGE 634
CHECK 634
CLOSE 634
CLSDST 634
EXECRPL 635
EXLST 635
GENCB 635
INQUIRE 636
INTRPRET 638
ISTGLBAL 639
MODCB 640
NIB 640
OPEN 640
OPNDST 641
OPNSEC 642
RCVCMD 643
RECEIVE 644
REQSESS 644
RESETSR 645
RPL 645
SEND 646
SENDCMD 647
SESSIONC 648
SETLOGON 648
SHOWCB 649
SIMLOGON 649
TERMSESS 650
TESTCB 650

Appendix B. Return codes and sense fields for RPL-based macroinstructions 651
Return code posting 651

RPL return code (RTNCD,FDB2) combinations 655
SNA sense fields 679

Appendix C. Summary of control requests and indicators 681

Appendix D. Request and response exchanges for typical communication operations 693

Appendix E. Control block formats and DSECTs 739
ACB (IFGACB) 740
ASDP (ISTASDP) 743
BLENT (ISTBLENT) 746
Control vector hex 29 (CV29) 747
EXLST (IFGEXLST) 750
MTS override (ISTMTS) 752
NIB (ISTDNIB) 753
NIB DEVCHAR (ISTDVCHR) 756
NIB PROC (ISTDPROC) 761
NRIPL (ISTNRIPL) 761
Request/response header (ISTRH) 763
RPL (IFGRPL) 766
RPL RTNCD-FDB2-FDBK (ISTUSFBC) 774
Access-method-support vector list (ISTAMSVL) 781
Resource-information vector list (ISTRIVL) 785
Application-ACB vector list (ISTVACBV) 789

Appendix F. Specifying a session parameter 793
Session parameter fields (BIND image) 793
Format 795
Type 795
Function management profile 795
Transmission services profile 795
Transmission services usage field 797
Pacing count 798
Logical unit presentation services profile 799
Logical unit presentation services usage field 800
Cryptographic control 815
Primary logical unit name length 816
Primary logical unit name 816
User data length 816
User data 816
BIND area format and DSECT 816

Appendix G. RPL fields associated with VTAM macroinstructions 847

Appendix H. Summary of register usage 853

Appendix I. Return codes for manipulative macroinstructions 855

Appendix J. Summary of operand specifications 857
Address 861
Quantity 862
Fixed value 863
Name 863
Register-indirect value 863
Indirect value 863

Appendix K. Forms of the manipulative macroinstruction 865
Optional and required operands 867
Optional and required operands for the alternative forms of GENCB 868

Optional and required operands for the alternative forms of MODCB . . . 869
Optional and required operands for the alternative forms of SHOWCB . . . 870
Optional and required operands for the alternative forms of TESTCB. . . . 872

Appendix L. Program operator coding requirements . . . 875
Defining a program operator . . . 875
Authorizing a program operator . . . 878
Method for writing a program operator . . . 878
VTAM operator commands. . . . 879
Operational characteristics . . . 880
Messages rerouted to a PPO . . . 880
Programming requirements. . . . 881
Orderly closing of a program operator . . . 882
Limiting VTAM messages queued to a program operator. . . . 883
Data exchanged between a program operator and VTAM. . . . 883
Header . . . 884
Data received from VTAM . . . 885
Data sent to VTAM . . . 887
Format and DSECT of the message and command header . . . 888
POA communication with tuning facility using the MODIFY QUERY COMMAND . . . 889

Appendix M. List of macroinstructions . . . 891

Appendix N. Application program migration . . . 893
Migrating from prior releases of VTAM . . . 893
COS name and logon mode name . . . 893
Increase of ACB size . . . 894
Application program minor node name in BIND . . . 894
Sequence number dependencies for LU type 0 3270 terminals . . . 894
Dynamic network access function . . . 894
Differences between BTAM and VTAM application programs . . . 895
Migrating from a single-domain to a multiple-domain environment . . . 895
Use of INQUIRE for a cross-domain resource. . . . 896
Specifying LOGMODE names with OPNDST for a cross-domain resource . . . 896
Use of INTRPRET for a cross-domain resource . . . 896
Considerations for a multiple-network environment . . . 896
Use of INQUIRE for a cross-network resource . . . 896

Appendix O. Architectural specifications . . . 897

Appendix P. Accessibility . . . 899

Notices . . . 903
Programming interface information . . . 911
Policy for unsupported hardware. . . . 911
Trademarks . . . 911

Bibliography . . . 913

Index . . . 917

Index for Communications Server: SNA Programming . . . 939

Communicating your comments to IBM . . . 961

Figures

1. SSCP, CP, PUs, and LUs in an SNA network . . . 3
2. Example of establishing an LU-LU session . . . 5
3. Major functions of the communication part of a VTAM application program . . . 8
4. Major programming elements in the communication part of a VTAM application program (Part 1 of 2) . . . 9
5. Major programming elements in the communication part of a VTAM application program (Part 2 of 2) 10
6. VTAM application programs in an SNA network . . . 15
7. Macroinstructions by category . . . 20
8. Synchronous operation . . . 40
9. Asynchronous operation with an ECB posted . . . 42
10. Asynchronous operation with an RPL exit routine scheduled . . . 43
11. Possible pattern of asynchronous requests in RPL exit routines . . . 44
12. Element per LU at assembly . . . 53
13. Format of a vector list . . . 58
14. Format of each vector within the application-ACB vector list . . . 60
15. Format of vectors built by VTAM during OPEN processing . . . 62
16. Format of the CV64 Vector List to supply with SETLOGON START . . . 65
17. Recovery data for INQUIRE OPTCD=PERSESS for sessions other than LU 6.1 and LU 6.2 . . . 140
18. Recovery data for INQUIRE OPTCD=PERSESS for LU 6.2 sessions . . . 141
19. Recovery data for INQUIRE OPTCD=PERSESS for LU 6.1 sessions . . . 142
20. Recovery data for OPNDST OPTCD=RESTORE for sessions other than LU 6.1 and LU 6.2 (NIBRPARM from INQUIRE is used) . . . 144
21. Recovery data for OPNDST OPTCD=RESTORE for LU 6.2 sessions (NIBRPARM from INQUIRE is used) 145
22. Recovery data for OPNDST OPTCD=RESTORE for LU 6.1 sessions (NIBRPARM from INQUIRE is used) 146
23. Recovery data for OPNDST OPTCD=RESTORE for sessions other than LU 6.1 and LU 6.2 (NIBRPARM from INQUIRE is not used) . . . 147
24. Recovery data for OPNDST OPTCD=RESTORE for LU 6.2 sessions (NIBRPARM from INQUIRE is not used) . . . 148
25. Recovery data for OPNDST OPTCD=RESTORE for LU 6.1 sessions (NIBRPARM from INQUIRE is not used) . . . 149
26. Legend for request and response flows . . . 152
27. Exchange requests and responses . . . 152
28. Receiving requests from an LU. . . . 155
29. Normal-flow requests are sent sequentially . . . 159
30. Difference between normal-flow and expedited-flow requests . . . 160
31. How sequence numbers are used . . . 164
32. Starting and stopping the flow of requests and responses . . . 166
33. General sequence of events when ECB-posting is specified . . . 171
34. General sequence of events when an RPL exit routine is specified . . . 172
35. Scheduled output . . . 173
36. Responded output . . . 174
37. Example of using any-mode and specific-mode to handle an inquiry on a session. . . . 176
38. Example of using continue-any and continue-specific modes to handle concurrent inquiries . . . 178
39. How input RUs are classified by VTAM . . . 179
40. How VTAM handles DFASY (expedited-flow data-flow-control request) input . . . 180
41. How VTAM handles RESP (normal-flow response) input . . . 181
42. How VTAM handles DFSYN (normal-flow request and DFSYN response) input . . . 182
43. Example showing values in the RECLEN field of an RPL . . . 183
44. LMPEO operation on a message sent to an SNA LU . . . 184
45. LMPEO handling of selected RH indicators . . . 187
46. LMPEO handling of selected RH-chain indicators . . . 188
47. Buffer-list operation . . . 192
48. Buffer-list entry format . . . 193
49. Buffer-list LMPEO-state transitions . . . 195
50. Example of a SEND operation, OPTCD=(LMPEO,BUFFLST,USERRH) with a maximum RU size of 100 200
51. Example of request chaining . . . 203

© Copyright IBM Corp. 2000, 2013

52. Example of sending a chain of requests to an LU that is buffering data (Part 1 of 2) 204
53. Example of sending a chain of requests to an LU that is buffering data (Part 2 of 2) 205
54. Example of an LU quiescing an application program in order to interrupt continuous sending (Part 1 of 2) 208
55. Example of an LU quiescing an application program in order to interrupt continuous sending (Part 2 of 2) 209
56. Quiesce protocol 211
57. Change-direction protocol 213
58. Bracket protocol 215
59. Example of using an RPL exit routine 221
60. ACB-oriented and NIB-oriented exit routines 230
61. Summary of addressability and save-area requirements for the mainline program. 232
62. Situations in which LERAD and SYNAD exit routines do not have to be reentrant 233
63. Situations in which LERAD and SYNAD exit routines must be reentrant. 234
64. Format of a Network Services Procedure Error request unit 247
65. Format of a Cleanup Session request unit 252
66. BIND information presented to SCIP exit 259
67. How OPEN/CLOSE error and special-condition information is organized 278
68. How manipulative-macroinstruction error and special-condition information is organized 279
69. How RPL-based macroinstruction error and special-condition information is organized 281
70. Summary of error and special-condition handling with synchronous operations 283
71. Summary of error and special-condition handling with asynchronous operations (Part 1 of 2). 284
72. Summary of error and special-condition handling with asynchronous operations (Part 2 of 2). 285
73. Multitasking a program 297
74. Multiple tasks using the same ACB 298
75. Multiple tasks, each with its own ACB 299
76. Single task with multiple ACBs 300
77. Categories of VTAM macroinstructions versus the authorized path function 301
78. Example of the use of authorized path 305
79. Example of a multiple-address-space configuration with one multiple-address-space ACB 313
80. Example of a multiple-address-space configuration with more than one multiple-address-space ACB 314
81. Categories of VTAM macroinstructions versus the multiple-address-space functions 315
82. Example of an application program operating in cross-memory mode. 318
83. Bracket-state transitions at the 3270 SLU 331
84. Example of Forward and Deliver request unit flow 339
85. How to code comments and continuation lines 372
86. How to code comments and continuation lines 374
87. Major options for a RECEIVE macroinstruction 467
88. Major RESETSR options 482
89. RPL fields applicable to the macroinstructions that can modify RPLs (Part 1 of 2). 512
90. RPL fields applicable to the macroinstructions that can modify RPLs (Part 2 of 2). 513
91. Major SEND options 517
92. How the POST operand in the SEND macroinstruction is used 522
93. General logic of Sample Program 1 574
94. Data request format 580
95. Header format 581
96. Sense information format 581
97. Possible data communication configuration for sample program 606
98. Organization and flow of Sample Program 2 608
99. Logic of the 3600 I/O routine 615
100. Logic of the chaining output routine 618
101. Logic of the 3270 I/O routine 620
102. Logic of the RESP exit routine 623
103. Logic of the DFASY exit routine 625
104. RTNCD,FDB2 combinations possible for each macroinstruction (Part 1 of 3) 652
105. RTNCD,FDB2 combinations possible for each macroinstruction (Part 2 of 3) 653
106. RTNCD,FDB2 combinations possible for each macroinstruction (Part 3 of 3) 654
107. Device-type LU initiates and establishes a session with a PLU application program 694
108. PLU application program acquires (initiates and establishes) a session with a device-type LU. 695
109. After a warm start, a PLU application program reestablishes a session and resynchronizes sequence numbers (Part 1 of 2) 696
110. After a warm start, a PLU application program reestablishes a session and resynchronizes sequence numbers (Part 2 of 2) 697

111. PLU application program and a secondary logical unit exchange data (Part 1 of 3) 698
112. PLU application program and a secondary logical unit exchange data (Part 2 of 3) 699
113. PLU application program and a secondary logical unit exchange data (Part 3 of 3 about sections A, B, C and D) 700
114. Logical unit sends a chain of data to the PLU application program 701
115. Application program and logical unit use quiesce protocol 702
116. Application program and logical unit use bracket protocol (Part 1 of 2) 703
117. Application program and logical unit use bracket protocol (Part 2 of 2) 704
118. Application program and logical unit use change-direction protocol 705
119. PLU application program resynchronizes sequence numbers with the logical unit. 706
120. Application program and logical unit use the signal request 707
121. Application program and logical unit use the LUSTAT request 708
122. Operations are shut down in an orderly fashion 709
123. Logical unit terminates a session 710
124. PLU application program terminates a session with the logical unit 711
125. PLU application program terminates a session with the logical unit with a CLSDST OPTCD=PASS 712
126. PLU application program and a device-type logical unit use cryptography in a required cryptographic session. 713
127. PLU application program and a device-type logical unit use cryptography in a selective cryptographic session. 714
128. SLU application program requests a session with the PLU application program (Part 1 of 2) 715
129. SLU application program requests a session with the PLU application program (Part 2 of 2) 716
130. PLU application program acquires (initiates and establishes) a session with the SLU application program 717
131. PLU application program issues a SIMLOGON to acquire (initiate a session with) the SLU application program 718
132. PLU application program resynchronizes sequence numbers with the SLU application program (Part 1 of 2) 719
133. PLU application program resynchronizes sequence numbers with the SLU application program (Part 2 of 2) 720
134. PLU application program and SLU application program use bracket protocol 721
135. PLU application program and SLU use-bracket protocol: BID by PLU rejected but Ready-to-Receive follows 722
136. SLU application program sends a conditional request for session termination 723
137. SLU application program sends an unconditional request for session termination. 724
138. SLU application program sends a Request Shutdown request 725
139. PLU application program shuts down the SLU application program 726
140. PLU application program and the SLU application program use cryptography in a required cryptographic session. 727
141. PLU application program and the SLU application program use cryptography in a selective cryptographic session. 727
142. PLU application program stops bracket initiation 728
143. Application program receives a load request during physical unit activation (Part 1 of 2) 729
144. Application program receives a load request during physical unit activation (Part 2 of 2) 730
145. PLU application program acquires (initiates and establishes) a session with an XRF backup-session service-type LU. 731
146. PLU application program acquires (initiates and establishes) a session with an XRF backup-session device-type LU 732
147. PLU application program issues SIMLOGON to acquire (initiate and establish) an XRF session with a device-type LU 733
148. PLU application program issues SIMLOGON to acquire (initiate and establish) an XRF session with a device-type LU 734
149. Device-type LU initiates and establishes a session with an XRF primary-PLU application program 735
150. Switch without XRF primary failing first 736
151. Switch after XRF primary fails 737
152. Normal XRF primary end of session 737
153. Normal XRF backup end of session with XRF primary. 738
154. Format of the ACB. 740
155. Format of ISTASDP 743
156. Dial number subfield 744
157. Direct call line name subfield 744
158. IDBLK/IDNUM subfield. 744
159. CPNAME subfield 744
160. Expanded dial information subfield 745
161. DLCADDR subfield 745

162. Connection name subfield 745
163. Format of BLENT 746
164. Format of the EXLST 751
165. Format of MTS 753
166. Format of the NIB 754
167. Format of NRIPL 762
168. Format of the RH 763
169. Format of the RPL (Part 1 of 2) 767
170. Format of the RPL (Part 2 of 2) 768
171. Format of session parameters area (BIND image) 794
172. Profile 0 presentation services usage field 800
173. Profile 1 presentation services usage field 801
174. Profile 2 and 3 presentation services usage fields. 807
175. Profile 4 presentation services usage field 809
176. Profile 6 for LU 6.2 presentation services usage field 813
177. Format of BNDAREA (ISTDBIND) 818
178. RPL fields associated with the SEND and SESSIONC macroinstructions for various modes of operation (Part 1 of 3) 848
179. RPL fields associated with the SEND and SESSIONC macroinstructions for various modes of operation (Part 2 of 3) 849
180. RPL fields associated with the SEND and SESSIONC macroinstructions for various modes of operation (Part 3 of 3) 850
181. RPL fields associated with the RECEIVE macroinstruction for various modes of operation (Part 1 of 2) 851
182. RPL fields associated with the RECEIVE macroinstruction for various modes of operation (Part 2 of 2) 852
183. VTAM operator control of a VTAM domain 876
184. VTAM operator control of a multiple-domain VTAM network 877

Tables

1. Summary of special-purpose exit routines . . . 29
2. Rules for specifying a session parameter . . . 34
3. Relative advantages of synchronous and asynchronous requests . . . 45
4. Some questions that affect program design and coding . . . 45
5. Vector lists . . . 58
6. Interaction between SETLOGON and the ACB MACRF operand . . . 82
7. OPNSEC macroinstruction PROC options used to send BIND response . . . 97
8. Session outage notification summary . . . 105
9. Summary of exit routines involved in session initiation . . . 107
10. Summary of exit routines involved in session outages or disruption . . . 108
11. Summary of exit routines involved in session termination by session participant . . . 109
12. Session outage notification UNBIND type codes and reason codes . . . 111
13. Language code settings (MVS only) . . . 118
14. How to specify a logon mode name . . . 123
15. Determining session parameters for an INQUIRE macroinstruction . . . 124
16. Determining session parameters for an OPNDST OPTCD=ACCEPT macroinstruction . . . 125
17. Determining session parameters for an OPNDST OPTCD=ACQUIRE macroinstruction . . . 125
18. Determining session parameters for a SIMLOGON or CLSDST OPTCD=PASS macroinstruction . . . 125
19. Determining session parameters for a REQSESS macroinstruction . . . 125
20. Level of cryptography for OPNDST requests . . . 134
21. Establishing cryptographic requirements using logon mode entry and definition for secondary end of session . . . 135
22. Level of cryptography for OPNSEC requests . . . 136
23. Information tracked for recovery of LU 6.1 and LU 6.2 sessions . . . 138
24. Summary of requests and responses transmitted on normal flow and expedited flow . . . 161
25. Location of the initial RH . . . 185
26. Possible chain indicators resulting from initial RH-chain indicator settings . . . 185
27. Negative-response handling by VTAM for SEND OPTCD=LMPEO . . . 190
28. Buffer-list entry format . . . 193
29. Relationship of the user RH field to the request/response header . . . 197
30. Summary of exit routines . . . 224
31. Parameter list for the EXLST exit routines . . . 226
32. DFASY exit routine: Registers upon entry . . . 237
33. LERAD exit routine: Registers upon entry . . . 238
34. LOGON exit routine: Registers upon entry . . . 240
35. LOSTERM exit routine: Registers upon entry . . . 242
36. Format of a Notify request unit (Part 1 of 5) . . . 247
37. Format of a Notify request unit (Part 2 of 5) . . . 249
38. Format of a Notify request unit (Part 3 of 5) . . . 250
39. Format of a Notify request unit (Part 4 of 5) . . . 250
40. Format of a Notify request unit (Part 5 of 5) . . . 251
41. NSEXIT exit routine: Registers upon entry . . . 253
42. RELREQ exit routine: Registers upon entry . . . 254
43. RESP exit routine: Registers upon entry . . . 255
44. RPL exit routine: Registers upon entry . . . 256
45. SCIP exit routine: Registers upon entry . . . 262
46. SYNAD exit routine: Registers upon entry . . . 263
47. TPEND exit routine: Registers upon entry . . . 265
48. DSECT—Creating macroinstructions . . . 272
49. Release-level macro global variables for VTAM V6R1.2. . . . 274
50. Manipulative macroinstructions: Return code values and meanings . . . 278
51. Summary of register and RPL feedback return codes following an RPL-based request . . . 280
52. Recovery action return codes and their general meanings . . . 280
53. Completion conditions acceptance stage of asynchronous requests . . . 286
54. Completion conditions completion of synchronous requests or for CHECK of asynchronous requests 288

© Copyright IBM Corp. 2000, 2013

55. Coding requirements for authorized path . . . 302
56. Valid AMODE and RMODE specifications . . . 319
57. Addressing mode for each kind of exit routine . . . 321
58. SNA sense information received at the application program . . . 332
59. Explanation of USENSEI information . . . 333
60. Actions taken by the network when a test request message is received . . . 335
61. Standard CNM header, SSCP-PU request format . . . 343
62. Standard CNM header, PU-SSCP request format . . . 343
63. Forward request unit format 0 request format . . . 344
64. Deliver request unit format 0 . . . 345
65. Format of additional targets in Deliver requests . . . 348
66. Examples of embedded network services request units . . . 349
67. Initiate load request (NS-INITLOAD) request unit format . . . 350
68. Load status request (NS-LOADSTAT) request unit format . . . 350
69. Types of network services request units not embedded . . . 351
70. CNM request unit format . . . 351
71. Timeout CNM request unit format . . . 352
72. Translate-inquiry request (TR-INQ) request unit format . . . 352
73. Translate inquiry data . . . 353
74. Network ID 1—ID in which the name to be translated was used . . . 353
75. Network ID 2—ID of the network in which the translated name is used . . . 353
76. Class-of-input names . . . 353
77. Input name type code (TR-INQ) . . . 353
78. Input name type code (TR-REPLY) . . . 354
79. Translate reply request (TR-REPLY) request unit format . . . 354
80. Translate reply data . . . 354
81. Network identifier 1 . . . 355
82. Network identifier 2 . . . 355
83. Translated name data . . . 355
84. Session-awareness data buffer header . . . 356
85. PIU trace data buffer header . . . 357
86. PIU trace data buffer entry . . . 357
87. Data categorization for performance monitor interface . . . 359
88. Request code vector format . . . 365
89. Format of all possible resource data descriptions . . . 367
90. Forms of the CLOSE macroinstruction . . . 391
91. Permissible option codes in the INQUIRE macroinstruction . . . 412
92. Forms of the OPEN macroinstruction . . . 445
93. Types of STSN requests and their possible responses . . . 542
94. Control block fields that can be tested with SHOWCB . . . 553
95. Control block fields that can be tested with TESTCB . . . 568
96. Sense field values . . . 680
97. Summary of sending normal-flow data-flow-control requests . . . 681
98. Summary of receiving normal-flow data-flow-control requests . . . 683
99. Summary of sending expedited-flow-control requests . . . 684
100. Summary of receiving expedited-flow data-flow-control requests . . . 685
101. Summary of sending session-control requests . . . 687
102. Summary of receiving session-control requests . . . 689
103. Summary of change-direction indicator . . . 690
104. Summary of bracket indicators . . . 691
105. ACB DSECT (IFGACB) . . . 740
106. ISTASDP DSECT . . . 745
107. ISTASDP subfield DSECT (ASDSUBFD) . . . 745
108. ISTASDP DLCADDR subfield DSECT (ASDDLCSF) . . . 746
109. Buffer list entry DSECT (ISTBLENT) . . . 746
110. Control vector hex 29 . . . 747
111. Control vector hex 29 (constants) . . . 750
112. EXLST DSECT (IFGEXLST) . . . 751
113. MTS override DSECT (ISTMTS) . . . 753
114. NIB DSECT (ISTDNIB) . . . 754
115. NIB's DEVCHAR DSECT (ISTDVCHR) . . . 756

116. NIB's PROC DSECT (ISTDPROC) 761
117. NRIPL DSECT (ISTNRIPL) 762
118. Request/response header DSECT (ISTRH) 763
119. RPL DSECT (IFGRPL). 768
120. RPL's RTNCD-FDB2-FDBK DSECT (ISTUSFBC) 774
121. Session-control requests for each transmission services profile 796
122. Maximum size of request unit (in decimal). 797
123. Session parameter fields: How they are made available and who can change them 819
124. BNDAREA DSECT (ISTDBIND) 821
125. BINPSCHR field of BNDAREA DSECT for logical unit profile 0. 828
126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1. 829
127. BINPSCHR field of BNDAREA DSECT for logical unit profile 2. 838
128. BINPSCHR field of BNDAREA DSECT for logical unit profile 3. 839
129. BINPSCHR field of BNDAREA DSECT for logical unit profile 4. 840
130. BINPSCHR field of BNDAREA DSECT for logical unit profile 6. 843
131. Structure of the XRF vector hex 27 846
132. Register contents upon return of control 853
133. Manipulative macroinstruction register 0 return codes when register 15 is 4. 855
134. Manipulative macroinstruction operands exclusive of control block operands 857
135. Manipulative macroinstruction operands for ACB fields 857
136. Manipulative macroinstruction operands for EXLST fields 858
137. Manipulative macroinstruction operands for RPL fields 858
138. Manipulative macroinstruction operands for NIB fields 860
139. Forms of manipulative macroinstructions 865
140. Some considerations that affect the coding of a program operator 881
141. Format of the VTAM message and command header (ISTDPOHD). 888
142. VTAM message and command header DSECT (ISTDPOHD) 888
143. Wraparound points for sequence numbers in sessions involving LU type 0 3270 terminals 894
144. Major similarities and differences between VTAM and BTAM application programs 895

About this document

This manual describes programming concepts and VTAM® macroinstructions. Use it as a guide when designing and coding application programs. This publication covers all macroinstructions except APPCCMD (see z/OS Communications Server: SNA Programmer's LU 6.2 Reference) and macroinstructions related to the installation of VTAM (see z/OS Communications Server: SNA Resource Definition Reference).

Who should read this document

This document is designed to help programmers (such as application or system programmers) write application programs that use VTAM macroinstructions.

How this document is organized

This document contains the following chapters:

- Chapter 1, "VTAM application program concepts," on page 1 provides an overview of a VTAM application program.
- Chapter 2, "VTAM language," on page 19 contains the following:
  - Characteristics of the VTAM language
  - A summary of the VTAM application program macroinstructions
  - Relationships among the VTAM control blocks as well as their relationship to the macroinstructions
  - A description of a normal operating system environment for a VTAM application program
- Chapter 3, "Organizing an application program," on page 33 discusses the following:
  - General coding guidelines to consider when writing an application program
  - Coding guidelines to facilitate migration of application programs
  - Whether the application program should be single-thread or multithread
  - Whether the application program operation should be synchronous or asynchronous and what posting mechanism should be used
  - VTAM and user control block decisions to be made that affect application program organization
- Chapter 4, "Opening and closing an application program," on page 55 describes aspects of the OPEN, CLOSE, and ACB macroinstructions that apply to all application programs.
- Chapter 5, "Establishing and terminating sessions with logical units," on page 81 describes how VTAM establishes and terminates a session between LUs.
- Chapter 6, "Communicating with logical units," on page 151 provides a general description of communication facilities.
- Chapter 7, "Using exit routines," on page 219 discusses how exit routines work, presents some of their advantages and disadvantages, and describes procedures to follow in using them.
- Chapter 8, "Setting and testing control blocks and macro global variables," on page 267 discusses the ways in which the VTAM application program sets and tests control block values.

© Copyright IBM Corp. 2000, 2013

- Chapter 9, “Handling errors and special conditions,” on page 277 discusses how to analyze information for errors and special conditions and what to do, in general, when the error or special condition is identified.
- Chapter 10, “Operating system facilities,” on page 295 describes a number of operating-system-dependent facilities to use when writing a VTAM application program.
- Chapter 11, “Programming for the IBM 3270 Information Display System,” on page 327 describes VTAM application programming for sessions that use LU type 0 protocols.
- Chapter 12, “Coding for the communication network management interface,” on page 337 describes the coding required for an application program to function on the communication network management (CNM) interface.
- Chapter 13, “Conventions and descriptions of VTAM macroinstructions,” on page 371 describes the format of the macroinstructions and then presents each macroinstruction in alphabetical order.
- Chapter 14, “Logic of a simple application program,” on page 573 shows the logic of a VTAM application program that receives a request for a session with a logical unit (LU), establishes the session, reads input from any session, processes the input, prepares a reply for output, and writes the output on the session.
- Chapter 15, “Sample code of a simple application program,” on page 579 contains the assembler language instructions for a VTAM application program, SAMP1.
- Chapter 16, “Logic of a more complicated application program,” on page 605 contains a more typical example of a VTAM application program than SAMP1.
- Chapter 17, “Sample code using authorized path,” on page 627 contains sample program, SAMP3, which shows an application program using the authorized path facility under the control of both a task control block (TCB) and a service request block (SRB).
- Appendix A, “Summary of control block field usage,” on page 633 serves as a reference for the experienced VTAM application programmer by showing the information for each executable macroinstruction discussed in this document.
- Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651 provides information about return code posting and explains what the different return code and feedback field values mean. It also provides information about SNA sense fields.
- Appendix C, “Summary of control requests and indicators,” on page 681 contains tables that summarize the SNA control requests and indicators sent and received by VTAM application programs.
- Appendix D, “Request and response exchanges for typical communication operations,” on page 693 contains diagrams that show the sequences in which requests and responses are exchanged to perform typical data communication operations using VTAM.
- Appendix E, “Control block formats and DSECTs,” on page 739 contains file description control block formats and DSECTs.
- Appendix F, “Specifying a session parameter,” on page 793 describes the format of the session parameter as seen by a VTAM application program.
- Appendix G, “RPL fields associated with VTAM macroinstructions,” on page 847 shows fields modified by the SEND and SESSIONC macroinstructions.
- Appendix H, “Summary of register usage,” on page 853 shows what VTAM does with the general-purpose registers before it returns control to the application program at the next sequential instruction.

- Appendix I, “Return codes for manipulative macroinstructions,” on page 855 explains return codes for manipulative macroinstructions.
- Appendix J, “Summary of operand specifications,” on page 857 indicates which manipulative macroinstructions apply for each operand and the types of values that can be coded with each operand.
- Appendix K, “Forms of the manipulative macroinstruction,” on page 865 summarizes the actions of various forms of manipulative macroinstructions.
- Appendix L, “Program operator coding requirements,” on page 875 describes how to write the program operator portion of a VTAM application program using the SENDCMD and RCVCMD macroinstructions.
- Appendix M, “List of macroinstructions,” on page 891 contains macroinstructions provided as programming interfaces by VTAM. Do not use as programming interfaces any VTAM macroinstructions other than those identified in this appendix.
- Appendix N, “Application program migration,” on page 893 describes factors to consider in various migration environments.
- Appendix O, “Architectural specifications,” on page 897 lists documents that provide architectural specifications for the SNA protocol.
- Appendix P, “Accessibility,” on page 899 describes accessibility features to help users with physical disabilities.
- “Notices” on page 903 contains notices and trademarks used in this document.
- “Bibliography” on page 913 contains descriptions of the documents in the z/OS® Communications Server library.

How to use this document

Before using this document, you should be familiar with the information in the assembler language documentation for your operating system. You should also be familiar with the following:

- Basic Assembler Language (BAL)
- Operating system facilities that the application program uses
- Characteristics of the devices with which the application program communicates
- SNA protocols
- Data communication concepts

Determining whether a publication is current

As needed, IBM® updates its publications with new and changed information. For a given publication, updates to the hardcopy and associated BookManager® softcopy are usually available at the same time. Sometimes, however, the updates to hardcopy and softcopy are available at different times. The following information describes how to determine if you are looking at the most current copy of a publication:

- At the end of a publication's order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.
- If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy

publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.
- To compare softcopy publications, you can check the last 2 characters of the publication's file name (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk (*) that indicates whether a publication is new or changed.

How to contact IBM service

For immediate assistance, visit this website: http://www.software.ibm.com/network/commserver/support/

Most problems can be resolved at this website, where you can submit questions and problem reports electronically, and access a variety of diagnosis information.

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-IBM-SERV). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m. – 5:00 p.m., local customer time).

Outside the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

If you would like to provide feedback on this publication, see "Communicating your comments to IBM" on page 961.

Conventions and terminology that are used in this document

Commands in this book that can be used in both TSO and z/OS UNIX environments use the following conventions:
- When describing how to use the command in a TSO environment, the command is presented in uppercase (for example, NETSTAT).
- When describing how to use the command in a z/OS UNIX environment, the command is presented in bold lowercase (for example, **netstat**).
- When referring to the command in a general way in text, the command is presented with an initial capital letter (for example, Netstat).

All the exit routines described in this document are *installation-wide exit routines*. The installation-wide exit routines also called installation-wide exits, exit routines, and exits throughout this document.

The TPF logon manager, although included with VTAM, is an application program; therefore, the logon manager is documented separately from VTAM.

Samples used in this book might not be updated for each release. Evaluate a sample carefully before applying it to your system.

Note: In this information, you might see the term RDMA network interface card (RNIC) that is used to refer to the IBM 10GbE RoCE Express feature.

For definitions of the terms and abbreviations that are used in this document, you can view the latest IBM terminology at the IBM Terminology website.

Clarification of notes

Information traditionally qualified as Notes is further qualified as follows:

Note Supplemental detail

Tip Offers shortcuts or alternative ways of performing an action; a hint

Guideline
Customary way to perform a procedure

Rule Something you must do; limitations on your actions

Restriction
Indicates certain conditions are not supported; limitations on a product or facility

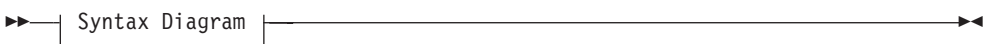
Requirement
Dependencies, prerequisites

Result Indicates the outcome

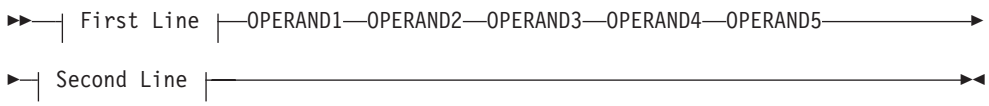
How to read a syntax diagram

This section describes how to read the syntax diagrams 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 two 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.

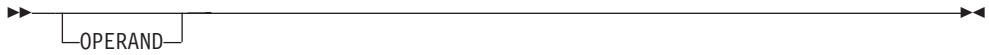


- Required operands and values appear on the main path line.

  You must code required operands and values.

  If there is more than one mutually exclusive required operand or value to choose from, they are stacked vertically in alphanumeric order.

- Optional operands and values appear below the main path line.

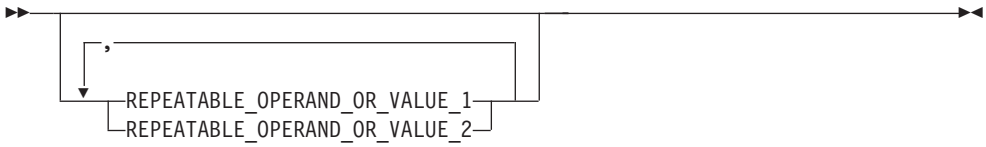


  You can choose not to code optional operands and values.

  If there is more than one mutually exclusive optional operand or value to choose from, they are stacked vertically in alphanumeric order below the main path line.

- An arrow returning to the left above an operand or value on the main path line means that the operand or value can be repeated. The comma means that each operand or value must be separated from the next by a comma.

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



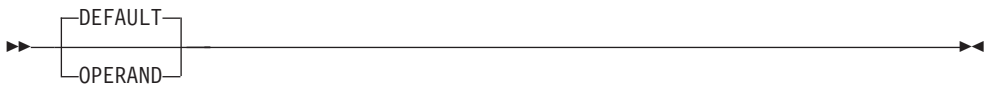
- A word in all uppercase is an operand or value you must spell exactly as shown. In this example, you must code ***OPERAND***.

  Note: VTAM and IP commands are not case sensitive. You can code them in uppercase or lowercase. If the operand is shown in both uppercase and lowercase, the uppercase portion is the abbreviation (for example, OPERand).

  If an operand or value can be abbreviated, the abbreviation is described in the text associated with the syntax diagram.
- If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code ***OPERAND=(001,0.001)***.

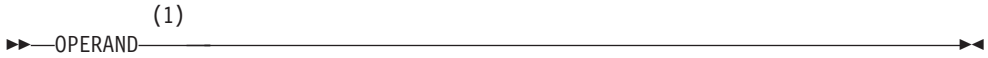
- If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code ***OPERAND=(001 FIXED)***.

- Default operands and values appear above the main path line. VTAM uses the default if you omit the operand entirely.



- 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.

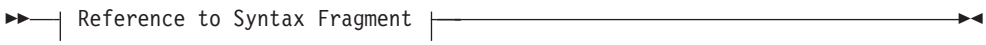
- References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.



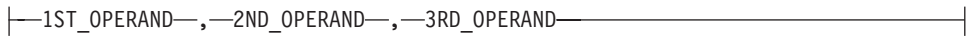
Notes:

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.



Syntax Fragment:



Prerequisite and related information

z/OS Communications Server function is described in the z/OS Communications Server library. Descriptions of those documents are listed in "Bibliography" on page 913, in the back of this document.

Required information

Before using this product, you should be familiar with TCP/IP, VTAM, MVS™, and UNIX System Services.

Softcopy information

Softcopy publications are available in the following collection.

Titles	Order Number	Description
IBM System z® Redbooks Collection	SK3T-7876	The IBM Redbooks® publications selected for this CD series are taken from the IBM Redbooks inventory of over 800 books. All the Redbooks publications that are of interest to the zSeries® platform professional are identified by their authors and are included in this collection. The zSeries subject areas range from e-business application development and enablement to hardware, networking, Linux, solutions, security, parallel sysplex, and many others. For more information about the Redbooks publications, see http://www-03.ibm.com/systems/z/os/zos/zfavorites/.

Other documents

This information explains how z/OS references information in other documents.

When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see z/OS Information Roadmap (SA23-2299). The Roadmap describes what level of documents are supplied with each release of z/OS Communications Server, and also describes each z/OS publication.

To find the complete z/OS library, including the z/OS Information Center, see www.ibm.com/systems/z/os/zos/bkserv/.

Relevant RFCs are listed in an appendix of the IP documents. Architectural specifications for the SNA protocol are listed in an appendix of the SNA documents.

The following table lists documents that might be helpful to readers.

Title	Number
DNS and BIND, Fifth Edition, O'Reilly Media, 2006	ISBN 13: 978-0596100575
Routing in the Internet, Second Edition, Christian Huitema (Prentice Hall 1999)	ISBN 13: 978-0130226471
sendmail, Fourth Edition, Bryan Costales, Claus Assmann, George Jansen, and Gregory Shapiro, O'Reilly Media, 2007	ISBN 13: 978-0596510299
SNA Formats	GA27-3136
TCP/IP Illustrated, Volume 1: The Protocols, W. Richard Stevens, Addison-Wesley Professional, 1994	ISBN 13: 978-0201633467
TCP/IP Illustrated, Volume 2: The Implementation, Gary R. Wright and W. Richard Stevens, Addison-Wesley Professional, 1995	ISBN 13: 978-0201633542
TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols, W. Richard Stevens, Addison-Wesley Professional, 1996	ISBN 13: 978-0201634952
TCP/IP Tutorial and Technical Overview	GG24-3376
Understanding LDAP	SG24-4986
z/OS Cryptographic Services System SSL Programming	*SC24-5901*
z/OS IBM Tivoli Directory Server Administration and Use for z/OS	*SC23-6788*
z/OS JES2 Initialization and Tuning Guide	*SA32-0991*
z/OS Problem Management	*SC23-6844*
z/OS MVS Diagnosis: Reference	*GA32-0904*
z/OS MVS Diagnosis: Tools and Service Aids	*GA32-0905*
z/OS MVS Using the Subsystem Interface	*SA38-0679*
z/OS Program Directory	*GI11-9848*
z/OS UNIX System Services Command Reference	*SA23-2280*
z/OS UNIX System Services Planning	*GA32-0884*
z/OS UNIX System Services Programming: Assembler Callable Services Reference	*SA23-2281*
z/OS UNIX System Services User's Guide	*SA23-2279*
z/OS XL C/C++ Runtime Library Reference	*SC14-7314*
zEnterprise 196, System z10, System z9 and eServer zSeries OSA-Express Customer's Guide and Reference	*SA22-7935*

Redbooks publications

The following Redbooks publications might help you as you implement z/OS Communications Server.

Title	Number
IBM z/OS V1R13 Communications Server TCP/IP Implementation, Volume 1: Base Functions, Connectivity, and Routing	SG24-7996
IBM z/OS V1R13 Communications Server TCP/IP Implementation, Volume 2: Standard Applications	SG24-7997
IBM z/OS V1R13 Communications Server TCP/IP Implementation, Volume 3: High Availability, Scalability, and Performance	SG24-7998
IBM z/OS V1R13 Communications Server TCP/IP Implementation, Volume 4: Security and Policy-Based Networking	SG24-7999
IBM Communication Controller Migration Guide	SG24-6298
IP Network Design Guide	SG24-2580

Title	Number
Managing OS/390® TCP/IP with SNMP	SG24-5866
Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender	SG24-5957
SecureWay Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements	SG24–5631
SNA and TCP/IP Integration	SG24-5291
TCP/IP in a Sysplex	SG24-5235
TCP/IP Tutorial and Technical Overview	GG24-3376
Threadsafe Considerations for CICS	SG24-6351

Where to find related information on the Internet

z/OS

This site provides information about z/OS Communications Server release availability, migration information, downloads, and links to information about z/OS technology

http://www.ibm.com/systems/z/os/zos/

z/OS Internet Library

Use this site to view and download z/OS Communications Server documentation

www.ibm.com/systems/z/os/zos/bkserv/

IBM Communications Server product

The primary home page for information about z/OS Communications Server

http://www.software.ibm.com/network/commserver/

IBM Communications Server product support

Use this site to submit and track problems and search the z/OS Communications Server knowledge base for Technotes, FAQs, white papers, and other z/OS Communications Server information

http://www.software.ibm.com/network/commserver/support/

IBM Communications Server performance information

This site contains links to the most recent Communications Server performance reports.

http://www.ibm.com/support/docview.wss?uid=swg27005524

IBM Systems Center publications

Use this site to view and order Redbooks publications, Redpapers™, and Technotes

http://www.redbooks.ibm.com/

IBM Systems Center flashes

Search the Technical Sales Library for Techdocs (including Flashes, presentations, Technotes, FAQs, white papers, Customer Support Plans, and Skills Transfer information)

http://www.ibm.com/support/techdocs/atsmastr.nsf

RFCs

Search for and view Request for Comments documents in this section of the Internet Engineering Task Force website, with links to the RFC repository and the IETF Working Groups web page

http://www.ietf.org/rfc.html

Internet drafts

View Internet-Drafts, which are working documents of the Internet Engineering Task Force (IETF) and other groups, in this section of the Internet Engineering Task Force website

http://www.ietf.org/ID.html

Information about web addresses can also be found in information APAR II11334.

Note: Any pointers in this publication to websites are provided for convenience only and do not serve as an endorsement of these websites.

DNS websites

For more information about DNS, see the following USENET news groups and mailing addresses:

USENET news groups
comp.protocols.dns.bind

BIND mailing lists
https://lists.isc.org/mailman/listinfo

BIND Users

- Subscribe by sending mail to bind-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind-users@isc.org.

BIND 9 Users (This list might not be maintained indefinitely.)

- Subscribe by sending mail to bind9-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind9-users@isc.org.

The z/OS Basic Skills Information Center

The z/OS Basic Skills Information Center is a web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS systems programmer.

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:

- Provide basic education and information about z/OS without charge
- Shorten the time it takes for people to become productive on the mainframe
- Make it easier for new people to learn z/OS

To access the z/OS Basic Skills Information Center, open your web browser to the following website, which is available to all users (no login required):

http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp

Summary of changes

This section describes the release enhancements that were made.

New in z/OS Version 2 Release 1

For specifics on the enhancements for z/OS Version 2, Release 1, see the following publications:

- z/OS Summary of Message and Interface Changes
- z/OS Introduction and Release Guide
- z/OS Planning for Installation
- z/OS Migration

© Copyright IBM Corp. 2000, 2013

Chapter 1. VTAM application program concepts

This chapter provides an overview of a VTAM application program. It describes a VTAM application program by showing:

- A VTAM application program's relationship to the concepts of Systems Network Architecture (SNA)
- The major programming elements in a VTAM application program
- A VTAM application program as part of an SNA network.

Systems Network Architecture (SNA) concepts in VTAM

VTAM follows SNA protocols and uses SNA concepts to connect and communicate with elements in a data communication network. The following SNA concepts provide helpful background information for the programmer writing a VTAM application program:

- Network accessible units (NAUs)
- Primary and secondary logical units (PLUs and SLUs)
- Sessions
- Domains
- Networks

Network accessible units

Each element in an SNA network to which a data or control message can be sent is assigned a network address. Each element with such an address is known as a network accessible unit (NAU). The network address uniquely identifies the element and contains the information necessary to route a message to its destination.

VTAM recognizes various types of NAUs defined by SNA:

- System services control points (SSCPs)
- Physical units (PUs)
- Logical units (LUs)
- Control points (CPs)

Figure 1 on page 3 shows the location of these types of NAUs in a simplified SNA network.

The system services control point (SSCP) is a unit of coding in VTAM that manages a domain. The SSCP performs functions such as bringing up the domain and shutting it down, assisting in the establishment and termination of sessions between NAUs, and reacting to network problems (such as failure of a link or cluster controller). To perform these functions, the SSCP must be able to communicate with physical units (PUs) and logical units (LUs) under its control.

A physical unit (PU) is not literally a physical device in the network. Rather, a PU is a portion of a device (usually programming or circuitry, or both) that performs control functions for the device in which it is located and, in some cases, for other devices that are attached to the PU-containing device. For the devices under its control, the PU takes action during activation and deactivation, error recovery and

© Copyright IBM Corp. 2000, 2013

resynchronization, testing, and gathering of statistics about the operation of the device. Each device in the network is associated with a PU.

The PU exists either within the device or within an attached controlling device. A PU exists within a host processor, a communication controller, and a cluster controller. For a terminal, however, the PU can be within the terminal, the host processor or the communication controller, or the cluster controller to which the terminal is attached.

A logical unit (LU) is a device or program by which an end user (an application program, a terminal operator, or an input/output mechanism) gains access to the SNA network. To the network, an LU is the source of a request coming into the network, but the LU might not be the original source.

The contents of the request or the information on which the request is based might have originated at a device controlled by the LU. (For example, in a 3601 Cluster Controller, the LU is a program that handles input and output for one or several finance terminals attached to the controller. Input actually originates at one of the terminals, but it is the LU—the program—in the 3601 that uses the input to create a request and begin transmitting the request.) Similarly, the network sees an LU as the destination of a request, but the LU can pass the data on to a device for recording, printing, or displaying to a terminal operator. (For example, data received by an LU—a program—in a 3601 can be passed on to a finance terminal to be displayed on the screen of that terminal.) In some cases, however, the LU is an intrinsic part of the device at which the data is displayed (for example, a 3767 terminal contains the LU and is the input/output device).

Each advanced peer-to-peer network (APPN) has a control point (CP) that manages the node and its resources in the APPN network. In an end node, the CP communicates with the CP in its network node server to obtain APPN network services. In a network node, the CP communicates with the CPs in adjacent network nodes to manage the network. The control point directs the activation and deactivation of CP-CP sessions and helps establish LU-LU sessions that cross the APPN network.

The control point name of an adjacent APPN node can be predefined or learned when the connection is established. The control point name of a VTAM node, or a composite network node, is the SSCP name specified on a VTAM start option.

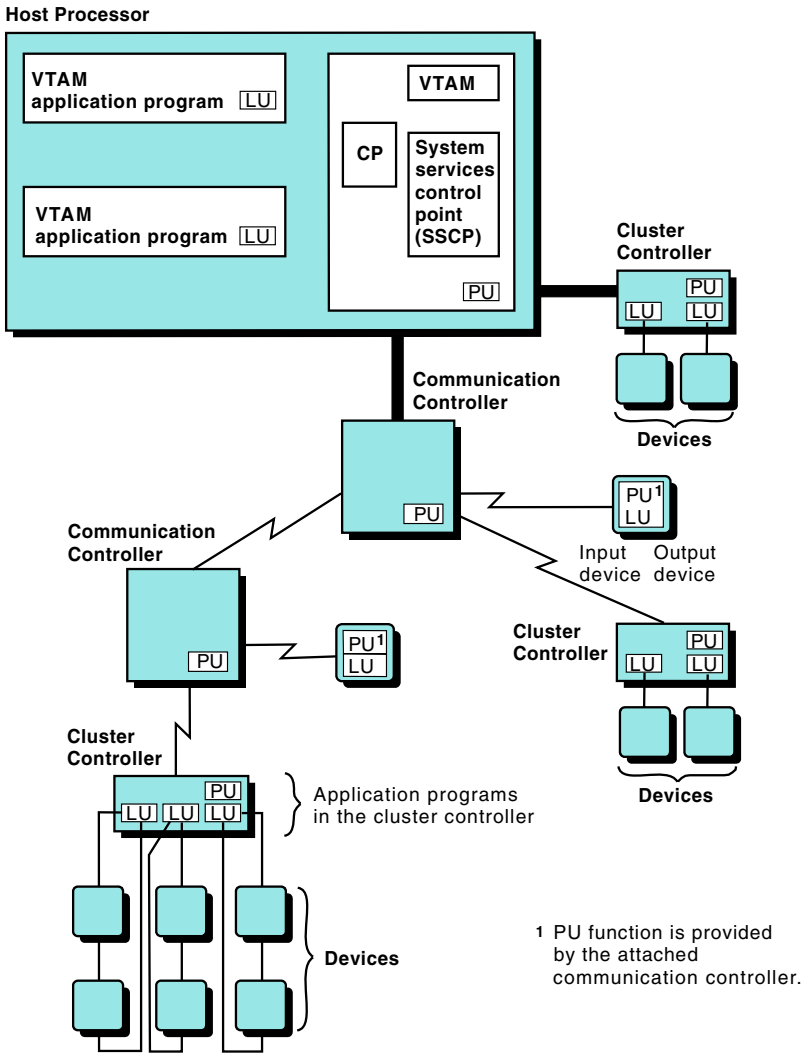


Figure 1. SSCP, CP, PUs, and LUs in an SNA network

A VTAM application program is also an LU. VTAM sees it as an originator of and destination for requests. Other programs in the host processor can interface with a VTAM application program and can receive or transmit the contents of requests. Therefore, although the VTAM application program is the LU, the requests it handles can be used by another program. In this case, the other program is the end user.

LUs are the only type of NAU that generally concern a VTAM application programmer. A programmer is usually not concerned with the SSCP, CP, and PUs. A VTAM application programmer is concerned with LUs because the program communicates with them. A VTAM application program does not generally communicate directly with the SSCP, CP, or PUs; however, requests issued by the application program can lead to actions by the SSCP, CP, or PUs.

Sessions

Before two NAUs can communicate with each other, they must be bound together in what is known as a session. An SNA network establishes the following types of sessions:

- SSCP-SSCP
- SSCP-PU
- SSCP-LU
- LU-LU
- CP-CP.

When a network includes more than one host processor, and therefore more than one VTAM (or VTAM in one or more host processors and other SNA access methods in one or more other host processors), a session called an SSCP-SSCP session must be established between the SSCP in one VTAM and any other SSCP with which the first SSCP communicates.

Within the portion of the network controlled by each SSCP, different types of sessions are established in stages. The SSCP must first establish an SSCP-PU session with each PU that is active in the configuration. Then, for each active LU associated with a PU, the SSCP must establish an SSCP-LU session. And finally, when a pair of LUs indicates that they want to communicate with each other, the SSCP must establish an LU-LU session between them.

In a particular session between two LUs, one LU adheres to a set of SNA-defined primary protocols and is known as the primary logical unit (PLU) for that session. The other LU adheres to a set of secondary protocols and is known as the secondary logical unit (SLU) for that session. More than one session can exist between two LUs. Multiple concurrent sessions between the same two LUs are referred to as parallel sessions. Not all LUs have parallel session capability.

A VTAM application program can act as a PLU, as an SLU, or as both at the same time. It can be a PLU in some sessions while it is also functioning as an SLU in others. In a set of parallel sessions involving an application program and another LU, the application program can operate as the PLU for some sessions, and the SLU for the remainder. The LUs associated with cluster controllers are called device-type LUs in this book; they can act only as SLUs.

CP-CP sessions are between control points in the APPN nodes, similar to SSCP-SSCP sessions between system service control points in subarea nodes. CP-CP sessions are used to exchange network information that enables control points to:

- Search for resources
- Register resources
- Set up LU-LU sessions
- Route network management data
- Learn the location and characteristics of nodes and links.

A CP-CP session is actually a pair of sessions between control points in adjacent nodes in an APPN network. This pair of sessions is an LU 6.2 contention-winner session and an LU 6.2 contention-loser session that together provide a CP-CP session. LU 6.2 protocols are used to communicate network control information and session control information.

A network node establishes a CP-CP session with each adjacent network node and with each served end node. Each end node is required to have a CP-CP session with a network node, which acts as its current network node server. Connections to adjacent LEN nodes do not support CP-CP sessions. After connection has been established between two attached nodes, a CP-CP session can be established.

Identification information is exchanged between the nodes. A CP-CP session is then automatically started between the control points, if the nodes indicate that they want the CP-CP session and if no CP-CP session already exists between them. Each node informs the other about the capabilities of its control point and about the links connected to it. If both nodes are network nodes, they exchange updated information about the nodes and links (and their characteristics) between network nodes within the network.

The sections that follow describe the steps required to establish some of the different types of sessions. The numbers beneath the headings correspond to the numbers in Figure 2. The example used is a device-type LU communicating with an application program LU.

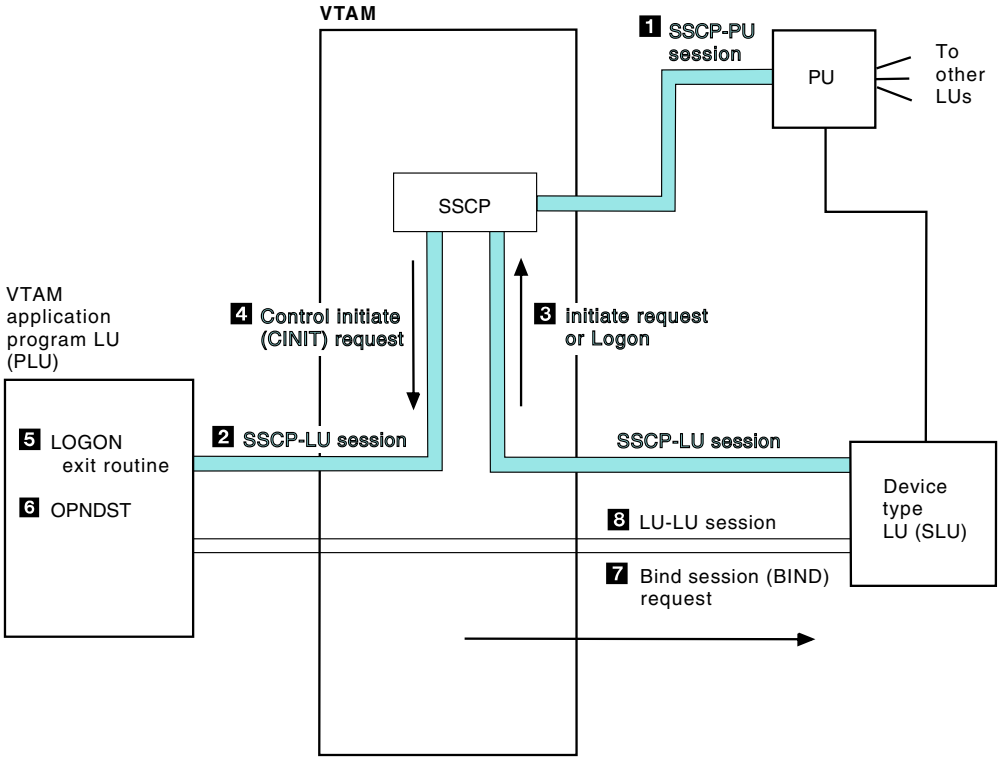


Figure 2. Example of establishing an LU-LU session

SSCP-PU session

1

To prepare for an SSCP-LU session, the SSCP must first establish a session with the PU that controls the LU. This type of session is an SSCP-PU session. It enables the exchange of requests and responses pertaining to startup and shutdown of the configuration or the individual PU, and to the recovery of operations after a device or link failure. After VTAM establishes the SSCP-PU session, the SSCP might attempt to establish a session with any active LU associated with that PU. The SSCP-PU session is established on a nonswitched line as soon as the PU is activated. On a switched line, the session is established following a dial-in or dial-out operation. For channel-attached SNA devices, the session is established when physical connection is established. In VTAM, the SSCP establishes the SSCP-PU session; a VTAM application program itself does not take any direct action to establish that session.

SSCP-LU session

2

Once a session is established between the SSCP and a PU, the SSCP can issue requests to establish a session between itself and any active LU associated with the PU. This type of session, called an SSCP-LU session, enables SNA requests to flow back and forth between the LU and the SSCP. These requests pertain mainly to LU-LU session establishment and termination. For application program LUs, VTAM establishes the SSCP-LU session with the application program when the application program opens its ACB. See "Opening the program" on page 10 for details on opening an ACB. The SSCP-LU session with the application program ends when closing its ACB. See "Closing the program" on page 13 for details on closing an ACB. For other LUs (for example, those associated with cluster controllers), VTAM usually establishes SSCP-LU sessions because of a command issued by the VTAM operator.

Initiate request or logon

3

After the SSCP establishes a session with the device-type LU, that LU might attempt to initiate a session with a VTAM application program LU. Action by the device-type LU to start such a session is usually initiated when a terminal operator communicates with the LU and indicates that he or she wants to work with an application program in the host processor. The LU either uses the logon information entered by the terminal operator to create an Initiate request to be sent to the SSCP, or the LU passes the logon information from the terminal operator to the SSCP in the form in which it is received from the operator. The SSCP immediately translates the logon form of the request into an Initiate request format.

In a similar manner, after its SSCP-LU session is established, a VTAM application program (replacing the device-type LU in Figure 2 on page 5) can request that a session be initiated between itself and another LU (for example, another VTAM application program). It does this by issuing a VTAM macroinstruction that causes an Initiate request (containing logon information) to be sent to the SSCP.

Control Initiate request

4

When the Initiate request is processed by the SSCP, the SSCP notifies the VTAM application program that the Initiate has been received by sending a Control Initiate (CINIT) request on the SSCP-to-application program LU session. Information derived from the original Initiate that requested the session is transmitted in the CINIT request and includes session parameters and, optionally, a user logon message. The user logon message contains particular user data that the terminal operator or LU that initiated the session wants to be passed to the VTAM application program. Session parameters and the user logon message are available in the CINIT for inspection by the program. Session parameters indicate the communication rules that the initiating LU wants used for the session that is about to be established. Parameters specify whether chained or unchained requests are sent, what kinds of responses are requested, which LU starts and ends brackets, and so on. These session protocols are described in Chapter 6, "Communicating with logical units," on page 151.

LOGON exit routine

5

VTAM notifies the application program that the CINIT has been received by scheduling execution of the program's LOGON exit routine. During processing of the CINIT, the application program determines whether the session parameters in the CINIT are the right parameters for the session or whether a different set of session parameters should be used. The LOGON exit routine then either accepts or rejects the CINIT by issuing an OPNDST or CLSDST macroinstruction.

Establishing the LU-LU session (OPNDST macroinstruction)

6

To establish a session with the LU, the application program issues an OPNDST macroinstruction. As a result of the OPNDST macroinstruction, VTAM builds a BIND request and sends it to the LU. To reject the CINIT, the application program issues a CLSDST macroinstruction, and the session is not established.

BIND request

7

The BIND request establishes the LU-LU session. The BIND request contains the session parameters used for the session. The two types of BIND requests are non-negotiable and negotiable. When you use the non-negotiable type, the BIND request contains the session parameters that the PLU decided to use for the session. These parameters might be the same as those suggested in the CINIT. If the SLU agrees with the session parameters, it sends a positive response to the BIND to establish a session. If the SLU does not agree with the session parameters, it rejects the request, and the LU-LU session is not established.

When the negotiable type BIND is used, if the SLU does not agree with the session parameters provided by the PLU, it can change them and return the modified parameters with its positive response. If the PLU agrees with the changes, the session has been established, and communication can begin. Otherwise, the PLU must issue a CLSDST macroinstruction to terminate the session; this causes VTAM to send an UNBIND request to the SLU and the session is terminated.

Completing the LU-LU session establishment

8

When VTAM receives a positive response to the BIND request, it completes the establishment of the LU-LU session and the LUs are ready to communicate. In some cases, the exchange of requests and responses might not begin until a Start Data Traffic (SDT) request is sent from the PLU to the SLU. The need for the SDT request is determined by the transmission services profile in the session parameters.

Major programming elements in a VTAM application program

Figure 3 on page 8 shows the major functions that any VTAM application program performs:

- Opening and closing the program (associating the program with and dissociating it from VTAM)
- Establishing and terminating sessions with other LUs
- Communicating with LUs with which the program is in session.

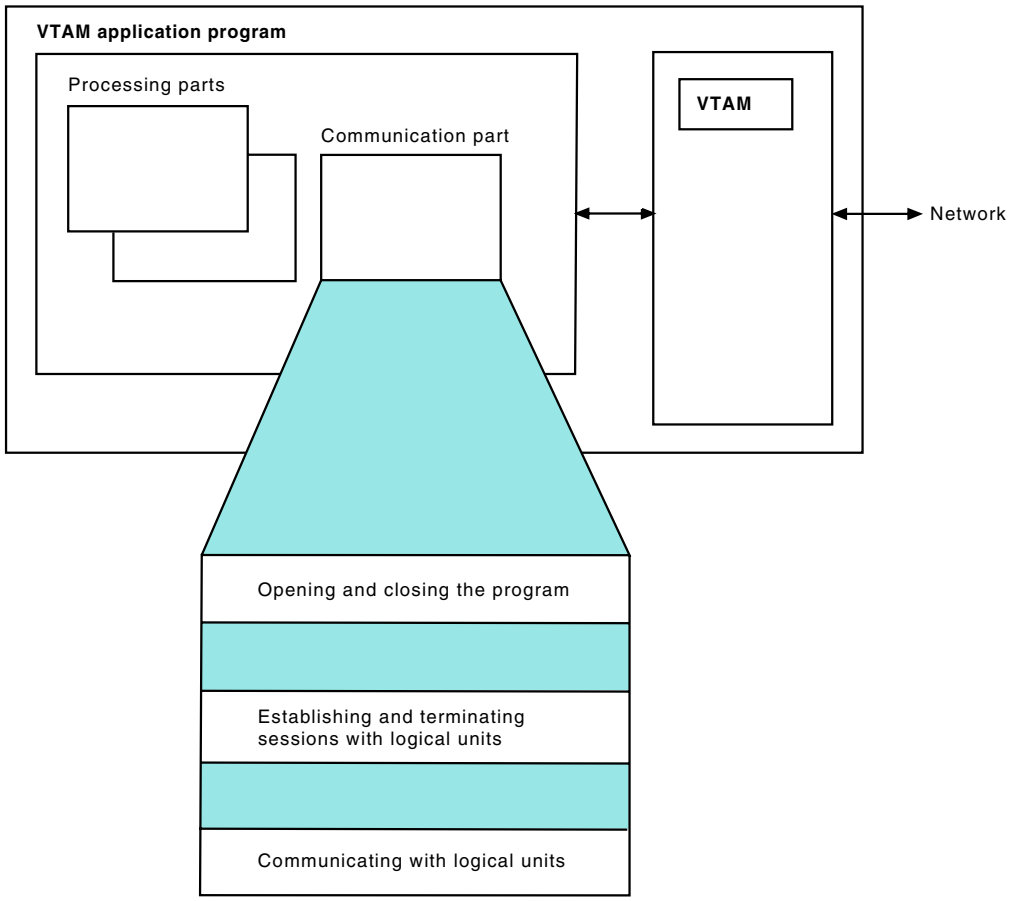


Figure 3. Major functions of the communication part of a VTAM application program

Figure 4 on page 9 and Figure 5 on page 10 show these major functions in more detail in the approximate order in which the functions occur. Although not every VTAM facility is shown, the facilities that are shown give a general idea of a VTAM application program. (In particular, the application program is the PLU; a different set of macroinstructions and exits is used to establish and terminate a session in which the application program is the SLU.) Following Figure 4 on page 9 and Figure 5 on page 10, the numbers beneath the section headings correspond to the numbers in Figure 4 on page 9 and Figure 5 on page 10.

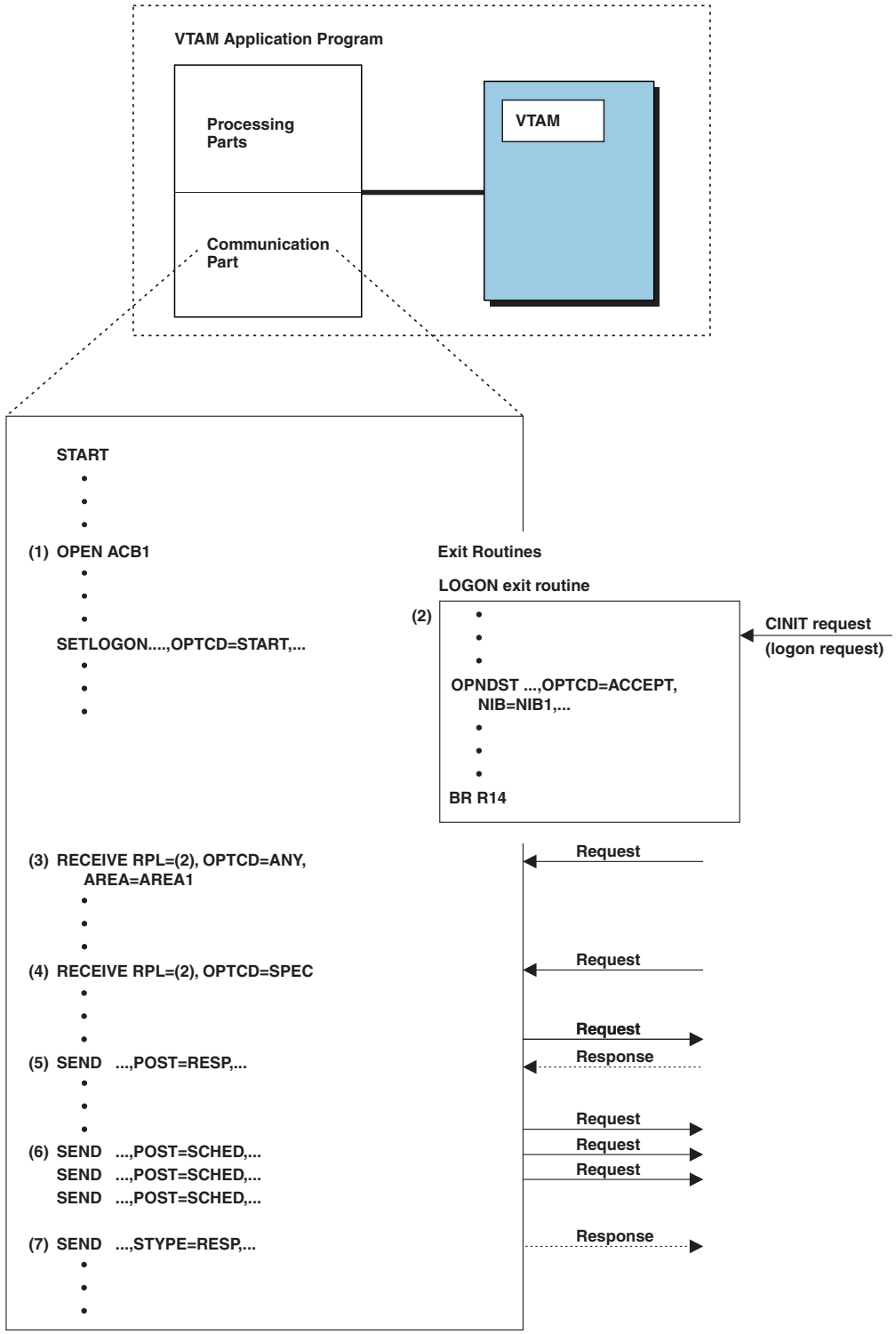


Figure 4. Major programming elements in the communication part of a VTAM application program (Part 1 of 2)

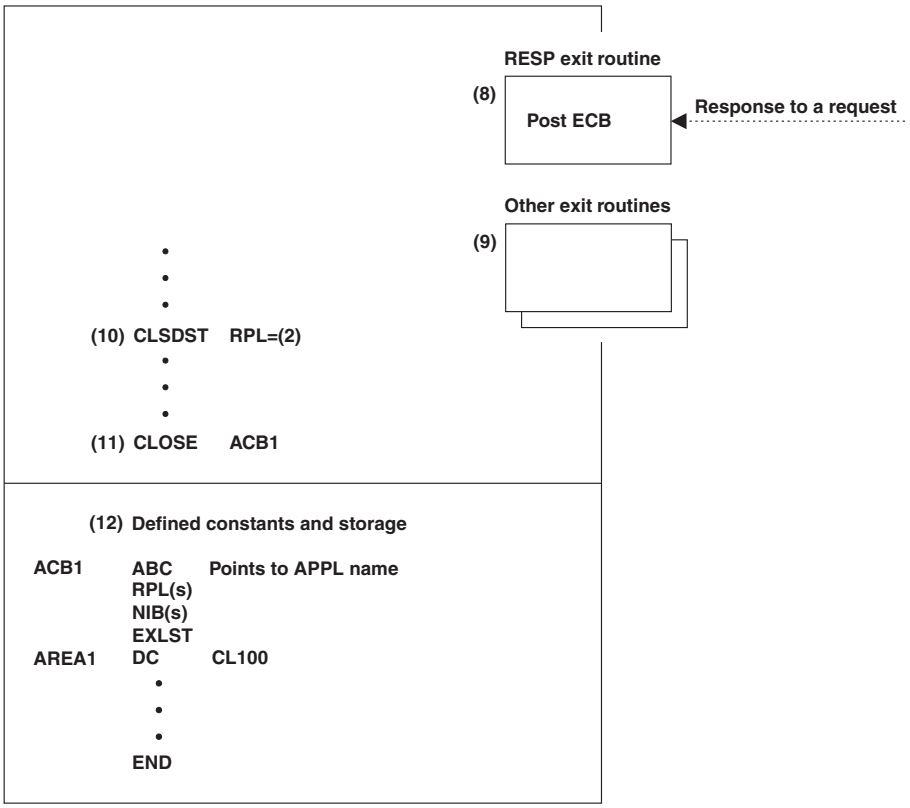


Figure 5. Major programming elements in the communication part of a VTAM application program (Part 2 of 2)

Opening the program

1

Assume that an application program has been started. The program issues an OPEN macroinstruction to open an access method control block (ACB). The ACB, in the data declarations area of the program, enables VTAM to relate the application program to the name of the APPL definition statement used by the system programmer to define the application program to VTAM.

VTAM verifies that a user is authorized to open a VTAM application ACB. VTAM does this by invoking the System Authorization Facility (SAF) router, which directs control to the installed security management product, such as the Resource Access Control Facility (RACF®), to perform the requested task. For information on how to use RACF for this security feature and what restrictions apply, refer to the z/OS Security Server RACF Security Administrator's Guide. For additional information on how SAF is invoked, refer to “Opening an application program” on page 55.

When the OPEN macroinstruction processing is completed, an SSCP-LU session is established between the SSCP and the application program LU. The SETLOGON macroinstruction is required to enable the LOGON exit routine to be driven when a CINIT is received.

Establishing a session with an LU

2

A common way to establish a session with an LU is to have the LU send an Initiate request, which asks for a session with a particular application program. A VTAM application program might have a LOGON exit routine that is automatically entered when a CINIT request resulting from the Initiate request is received. The LOGON exit routine establishes the session with the LU by issuing OPNDST OPTCD=ACCEPT. The OPNDST points to a node initialization block (NIB). The NIB contains information that VTAM associates with the session. When the OPNDST processing is completed, the application program and the LU can exchange requests and responses.

When an application program establishes a session with another LU, a 32-bit communication identifier (CID) is returned in two control blocks used in establishing the session (the request parameter list (RPL) and the node initialization block (NIB)). The CID identifies the session. Whenever the application program sends a request or response unit (RU) to the LU on a particular session, the CID of that session must be in the RPL used to send the RU.

Receiving requests from LUs

3 4

After the VTAM application program establishes one or more sessions, the program can issue the RECEIVE macroinstruction to receive input on a session with an LU.

The RECEIVE and most other VTAM application program macroinstructions must furnish the address of a request parameter list (RPL), shown in the data declarations area of Figure 4 on page 9. Fields in the RPL contain parameters that tell VTAM exactly how to perform the requested operation. On completion of a requested operation, VTAM places feedback information in the RPL, where it can be examined by the application program.

Each request that a VTAM application program sends or receives can contain data or control information or both. Data is information that is meaningful only to the processing portion of an application program. If a request contains data, the RECEIVE operation moves the data from VTAM to a designated area in the application program, for example to AREA1 shown in the data declarations area of Figure 4 on page 9. The application program uses control information to direct the further exchange of requests and responses on the session. The application program puts control information into, or receives the control information from, the RPL that is specified on the SEND or RECEIVE macroinstruction.

A RECEIVE macroinstruction can be issued to receive input on a specific session, or on any session. To receive a request sent on any session, a RECEIVE OPTCD=ANY is issued, as shown at **3** in Figure 4 on page 9. Such a RECEIVE is completed if VTAM is holding a request received on any session or is completed when such a request is received. (A facility exists to preclude selected sessions from completing such a RECEIVE.)

A RECEIVE macroinstruction can also be issued in such a way that only a request received on a specific session satisfies the RECEIVE. To do this, a program issues a

RECEIVE OPTCD=SPEC with the RPL indicating the session from which the input is desired, as shown at **4** in Figure 4 on page 9.

Completion of RECEIVE OPTCD=ANY might be followed by execution of a processing routine of the program and, subsequently, by having the processing part of the program call the communication part with individual requests for input and output. The communication part issues RECEIVE OPTCD=SPEC at **4** or it issues one or more SENDs at **5** or **6**.

As implied by the preceding description, a RECEIVE is not completed until VTAM receives a request from an LU and passes it to the application program.

Sending a request

5 6

In contrast to a RECEIVE macroinstruction, a SEND macroinstruction can be completed at either of two different times:

- When the request is scheduled (that is, when VTAM has accepted the request, moved the data to its own output area, and prepared everything for the transmission)
- When the request has been responded to (that is, after VTAM has sent the request and received a response from the LU at the other end of the session).

To specify completion upon receipt of a response, the programmer uses a SEND POST=RESP (as at **5**), meaning that the results of the operation are immediately available in the RPL when SEND is completed. With POST=RESP, the application program cannot reuse the RPL or output buffer associated with the SEND until the operation is completed.

As an alternative to POST=RESP, a request can be scheduled for output (SEND POST=SCHED). On completion of the SEND **6**, the data has been accepted by VTAM, and the application program can reuse the RPL and the output buffer. The application program itself must determine if a schedule request actually arrives at its destination and is processed successfully. One way is to request the LU to send back a response, which is an indication of whether and how a request arrives and is processed. A response can be requested in each SEND POST=SCHED. When the response arrives, VTAM either completes a RECEIVE that can receive responses (not shown) or enters a VTAM application program's RESP exit routine, such as the one at **8**.

Sending a response

7

The SEND requests in **5** and **6** specify the sending of a request. The application program might also want to send a response to a request that it had previously obtained with RECEIVE. This is done by specifying STYPE=RESP instead of STYPE=REQ in the SEND macroinstruction and by specifying other parameters to indicate the type of response (for example, positive or negative) to be sent. A response is sent because the LU requested it; it is a special kind of RU that is identified as a response to a particular preceding request. Each request is given a sequence number by the sender's access method (VTAM or the LU); the receiver of a request puts the same sequence number in a response for that request, thus indicating within the response which request is being responded to.

Receiving a response

8

When an application program uses SEND POST=SCHED macroinstructions, the application program can take direct action to receive each response; that is, for each response, it can issue RECEIVE RTYPE=RESP. Instead, a program can contain a RESP exit routine (at **8**), which is scheduled each time a response is received. The RESP exit routine can notify the mainline program of receipt of the response, perhaps by posting an event control block (ECB). The mainline program must then correlate the response with the SEND operation that produced it.

Other exit routines

9

In addition to the LOGON and RESP exit routines, VTAM provides automatic scheduling of other special-purpose exit routines. For example, VTAM schedules the LERAD and SYNAD exit routines when the application program issues a macroinstruction that uses an RPL (such as SEND or RECEIVE) and when that macroinstruction results in an error or a special condition. The addresses of these special-purpose exit routines are identified to VTAM in an EXLST macroinstruction (in the data declarations area). The ACB, NIB, or both point to the EXLST.

VTAM also provides another general kind of exit routine, the RPL exit routine. An RPL exit routine is identified in the EXIT operand of an RPL-based macroinstruction (any macroinstruction that uses an RPL). A different RPL exit routine can be identified in each RPL-based macroinstruction, or some macroinstructions can use the same exit routine. When the operation requested in the macroinstruction is completed, control is automatically given to the exit routine specified in the EXIT operand. Having an RPL exit routine scheduled upon completion of a request is an alternative to having VTAM post an ECB upon completion.

Terminating a session with an LU

10

The application program terminates a session with an LU by issuing a CLSDST macroinstruction. If the LU can support only one session at a time, that session must end before the LU can enter into a session with another application program.

Closing the program

11

An application program can be closed when the program determines it should be or when the VTAM operator requests it. To close an application program, the program issues a CLOSE macroinstruction. This terminates any sessions the application program has with other LUs and disassociates the program from VTAM (that is, the SSCP-LU session is terminated).

Constants and control blocks

12

In addition to buffers (data areas) required for input and output messages, and other areas such as status flags for LUs, each VTAM application program must define (or generate dynamically) the following control blocks:

- One ACB to define several facts about the program itself.
- One EXLST (list of exit routine names) if any exit routines are to be written. Although not strictly required, certain exit routines, such as TPEND, NSEXIT, SCIP, and LOSTERM are strongly recommended.
- At least one RPL for each request that is pending concurrently with other requests.
- At least one NIB for each concurrent session-establishment procedure. It is possible to use only one NIB if sessions are established one at a time.

VTAM macroinstructions

VTAM provides a macroinstruction (the GENCB macroinstruction) that allows you to create control blocks and initialize certain fields during program execution. VTAM also provides macroinstructions (the MODCB, SHOWCB, and TESTCB macroinstructions) that allow you to test and change certain control block fields. In addition, macroinstructions are provided that generate DSECTs for the control blocks. These DSECTs enable control block fields to be located and tested with assembler language instructions. VTAM application program macroinstructions and control blocks are discussed in more detail in Chapter 2, "VTAM language," on page 19.

VTAM application program as part of an SNA network

Figure 6 on page 15 shows an example of a VTAM application program as part of an SNA network. (An SNA network is the part of a user-application network that conforms to the formats and protocols of SNA. It enables you to transfer data reliably among end users and provides protocols for controlling the resources of various network configurations.) The numbers in Figure 6 on page 15 refer to major parts of the system and are described in the text that follows.

A VTAM application program can communicate with the following:

- Other VTAM application programs
- SNA devices (channel-attached and SDLC link-attached)
- Non-SNA 3270 devices (channel-attached and BSC link-attached)
- Non-SNA devices in conjunction with the Network Terminal Option (NTO) licensed program
- LUs controlled by another VTAM

When the unqualified term *logical unit* is used in this book, it refers to any or all of the above. In general, the term *device-type LU* refers to any LU other than an application program.

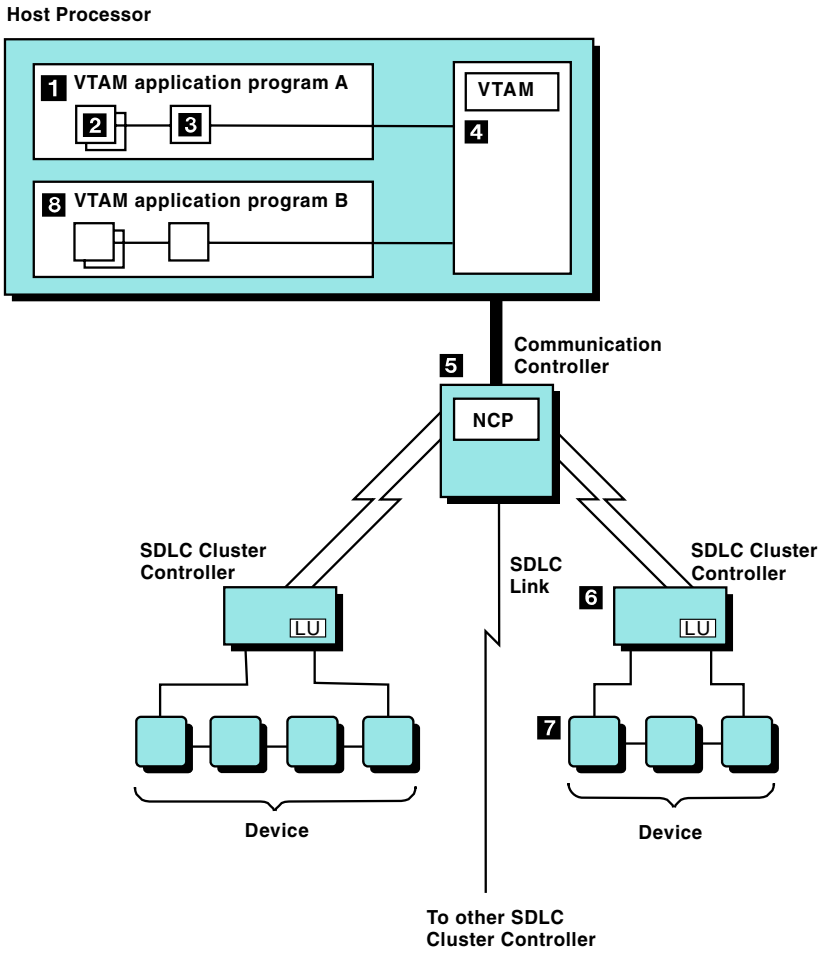


Figure 6. VTAM application programs in an SNA network

VTAM application program

1

A VTAM application program can contain two types of instructions: communication instructions and processing instructions. The application program always contains communication instructions. These instructions send and receive SNA messages, called request/response units (RUs), and control other aspects of communication between the application program and other elements in the network. The application program can also contain processing instructions. These optional instructions manipulate data before it is sent or after it is received.

If a VTAM application program is small and contains processing instructions, those processing instructions can be blended with communication instructions. More commonly, however, the processing instructions are written separately, with an interface defined between one or more processing parts of the program and the communication part of the program. This separation of function enables each part to be created separately and means that changes or additions to one part do not affect other parts.

VTAM application programs can share the resources of the SNA network; that is, multiple application programs can use the same communication controllers, cluster

controllers, and communication lines to reach LUs. For example, in Figure 6 on page 15, VTAM application programs A and B use the same communication controller (at **5**) and the same SDLC links to reach LUs (at **6**). The application programs, however, are not aware that they are sharing these resources because VTAM, the network control program (NCP), and other programming elements in the network handle communication in such a way that the programs do not know they are sharing the resources.

Processing part

2

The instructions in the processing parts of an application program can be written in assembler language or in a higher-level language, such as PL/I or COBOL. If written in assembler language, the instructions can be blended with the communication instructions in the program. But more commonly, as shown in Figure 6 on page 15, the processing parts **2** are separate and request data communication services by calling or branching to the communication part **3** of the program. Many programs contain several processing parts (routines or modules) that use a common communication part.

Communication part

3

The communication part of a VTAM application program contains macroinstructions and associated control blocks used to communicate with LUs. The communication part must be written in assembler language.

VTAM part

4

The part of an SNA network controlled by VTAM is called its domain. LUs are normally defined as part of the domain during VTAM definition and are then activated by start procedures or VTAM operator commands. The VTAM application program, which is itself an LU, then establishes sessions with one or more other LUs (on its own initiative or as the result of session-initiation requests by other LUs). A session is a logical connection between two LUs that allows an orderly series of communications between the LUs. Once a session is established, the program requests VTAM to perform data-transfer operations. In addition to managing the domain and building channel programs, VTAM performs such services as input and output data buffering, automatic scheduling of application program exit routines, and sequence numbering of outbound requests. VTAM requests the operating system to execute channel programs that it has built; the channel programs result in communication with channel-attached communication controllers or cluster controllers, which in turn forward the information toward its ultimate destination. For processors with a communication adapter, the channel programs can be used to communicate directly with terminals, in addition to cluster controllers and communication controllers.

Network control program

5

On receiving the input or output requests and associated data, the network control program (NCP) in the communication controller does what is required to communicate with LUs on data communication lines. Many functions previously

performed by a host-processor access method (for example, basic telecommunications access method, BTAM) or application program are now performed by the communication controller (for example, the communication controller schedules line activity, retries operations after transmission errors, and collects error statistics).

Logical unit

6

A VTAM application program communicates with other LUs. In this example, the LUs with which the application program is communicating are associated with an SNA cluster controller.

In general, for programmable SNA devices, the user defines which processing functions take place in a program in the programmable device (such as a cluster controller) and which take place in the VTAM application program in the host processor. The user must coordinate the cluster controller program and the VTAM application program to enable them to work together.

Terminal operator and devices

7

If the VTAM application program communicates with a cluster controller program rather than with a non-programmable terminal, the VTAM application program might not need to know about the terminal operator or device actions. The LU (implemented by the cluster controller program) determines whether and how data received from a terminal operator or device goes to the VTAM application program and whether and how data received from the VTAM application program goes to the terminal operator or device.

A VTAM application program can also communicate with a batch-transmission or batch-reception program in a cluster controller (such as the 3791 batch function). The VTAM application program does not need to know the original source of, or the eventual disposition of, data received from or sent to the subsystem batch program.

Another VTAM application program

8

A VTAM application program can also communicate with another VTAM application program. The two application programs can be in the same host processor or in different host processors.

LU 6.2 components are an addition to VTAM. They perform many, but not all, LU 6.2 functions. LU 6.2 components enable an LU representing one application to hold a conversation with an LU representing another application. The applications participating in the conversation can be in the same domain or in different domains. LU 6.2 initiates and terminates conversations between LUs using normally established sessions. Although there can be multiple conversations using one LU 6.2-initiated session, they cannot be concurrent.

LU 6.2 associates resources with application programs. The application programs must further associate the resources with the appropriate transaction programs.

LU 6.2 components of VTAM together with the application program form a type 6.2 LU. VTAM supports most of the functions defined in the LU 6.2 architecture, but not all of them. For further information on LU 6.2, refer to the z/OS Communications Server: SNA Programmer's LU 6.2 Guide.

Using a VTAM application program to manage the network

VTAM provides two independent facilities through which properly authorized application programs can manage a network. With the first facility, a program operator application program can issue most of the VTAM operator commands and can receive VTAM messages normally delivered to a VTAM operator console. Chapter 2, "VTAM language," on page 19, describes the SENDCMD and RCVCMD macroinstructions. Refer to Appendix L, "Program operator coding requirements," on page 875, for information on program operator coding requirements.

With the second facility, an authorized application program can use the communication network management (CNM) interface. Such an application program can collect data related to SNA sessions from VTAM and NCP, collect maintenance-related information from a PU, or load a PU type 2 within its domain. This capability, described in Chapter 12, "Coding for the communication network management interface," on page 337,

Chapter 2. VTAM language

An application program uses VTAM macroinstructions to request VTAM services. VTAM provides assembler language macroinstructions to:

- Associate an application program with or disassociate it from VTAM
- Establish and terminate sessions between the application program and specific LUs
- Communicate with logical units (LUs)
- Build and initialize control blocks used when the application program requests session establishment, communication, or other services from VTAM
- Manipulate a control block (for example, to test the value of a field in a control block)
- Communicate with the SSCP to perform communication network management (CNM) functions
- Transfer VTAM operator commands and messages between an application program and VTAM.

VTAM macroinstructions have the same format as assembler instructions and follow the same rules. See "How the macroinstructions are described" on page 371 for coding rules that apply to these instructions.

Characteristics of the language

The following sections describe some characteristics of the VTAM language.

Keyword operands

The operands in VTAM macroinstructions, with the exception of OPEN and CLOSE, are keyword operands rather than positional operands. Keyword operands make the coding easier to read. The keywords themselves identify control block fields. Most keyword operands are optional.

Manipulative macroinstructions

These macroinstructions provide an easy way to build control blocks and gain access to particular control block fields, usually to test or display values after a requested operation. Because you specify fields symbolically, you do not have to know field displacements.

Exit routines

The VTAM application program specifies that VTAM schedule special-purpose exit routines automatically. These routines handle conditions such as receiving a request for session establishment from the SSCP or receiving a certain type of control request on an LU-LU session. In addition, a VTAM application program can specify that VTAM complete a particular operation by scheduling an RPL-specified exit routine instead of posting an ECB. RPL exit routines provide several advantages over posting an ECB. These advantages include additional programming flexibility and convenience, as well as greater priority in handling an event's completion.

© Copyright IBM Corp. 2000, 2013

Summary description of the VTAM macroinstructions

The following sections provide a summary description of the VTAM application program macroinstructions. For a complete description of each macroinstruction, refer to "Description of the VTAM macroinstructions" on page 375.

In the descriptions of VTAM macroinstructions, you encounter the terms ***declarative***, ***manipulative***, ***ACB-based,*** and ***RPL-based***. These terms refer to categories of VTAM macroinstructions that have related functions. Figure 7 shows these categories and identifies the macroinstructions that are included in each one.

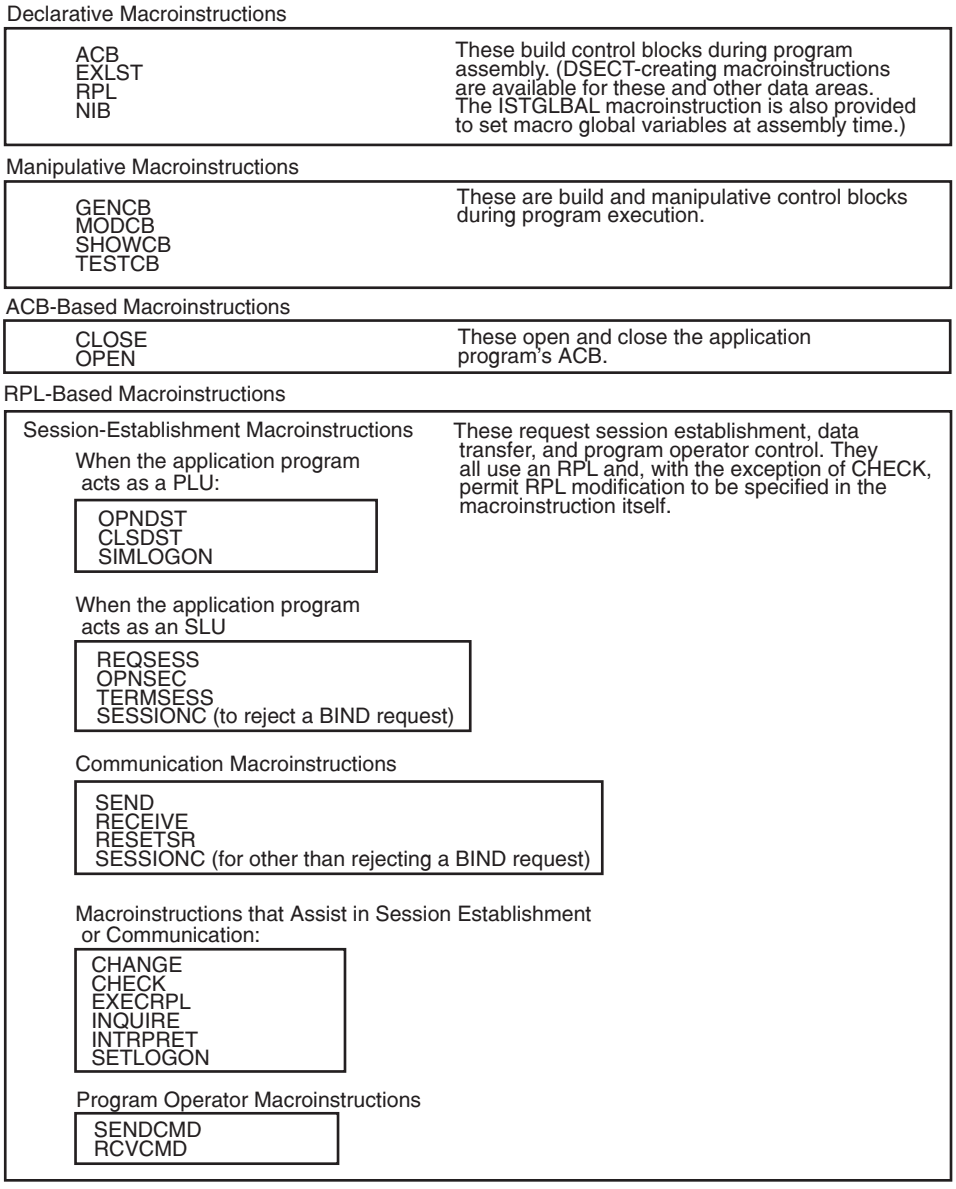


Figure 7. Macroinstructions by category

Declarative macroinstructions

Control blocks are built and initialized by coding a macroinstruction for each ACB, EXLST, NIB or RPL control block. The operation codes of the macroinstructions are identical to the names of the control blocks that they build and initialize. The following macroinstructions build control blocks:

ACB

Builds and initializes an access method control block (ACB). An ACB contains information that the application program provides VTAM about the application program in its entirety. Primarily, it names the application program and the list of exit routines associated with the application program. The ACB contains information about the *application program*.

EXLST

Builds and initializes an exit list (EXLST). An EXLST contains the addresses of application program exit routines that VTAM schedules when certain conditions occur (for example, when a CINIT request resulting from an Initiate request by an LU is received). The EXLST contains the addresses of *exit routines*.

NIB

Builds and initializes a node initialization block (NIB). An NIB contains information that the application program provides VTAM about general communication characteristics that exist on a session. This information is provided to VTAM as part of a session-establishment request and it remains in effect for the duration of the session. The NIB contains information about a *session*.

RPL

Builds and initializes a request parameter list (RPL). An RPL contains information (parameters) that the application program provides VTAM when requesting session establishment, communication, or any other RPL-based operation. On completion of the requested action, the RPL contains information that VTAM has put there for the application program. The RPL contains information about a *request for an operation*.

An ACB, EXLST, NIB, or RPL control block can be assembled in the application program by using the appropriate control-block-building macroinstruction described in the preceding sections, or the control block can be created and initialized during program execution by using the GENCB macroinstruction described in "Manipulative macroinstructions" on page 22.

DSECT-creating macroinstructions are provided by VTAM to generate maps for the ACB, EXLST, NIB and RPL control blocks. The DSECT-creating macroinstructions are designated IFGACB, IFGEXLST, ISTDNIB and IFGRPL. See "Using DSECT-creating assembler instructions and macroinstructions" on page 272 for details.

Also, at assembly time, the ISTGLBAL macroinstruction can be used to declare and set macro global variables that describe the VTAM product installed. For further information on ISTGLBAL macro global variables, refer to "ISTGLBAL macroinstruction" on page 274.

Manipulative macroinstructions

VTAM provides a group of macroinstructions that manipulate certain control block fields. These macroinstructions are more convenient than assembler language instructions because they refer to fields symbolically rather than by specific control-block field location. The manipulative macroinstructions are:

GENCB

Builds an ACB, EXLST, NIB, or RPL during program execution and can initialize certain fields with specified values. GENCB can also generate the control blocks in dynamically allocated storage. One GENCB can build multiple copies of one control block.

SHOWCB

Obtains the values from certain fields of a control block and places them in an area in the application program where they can be examined. In addition to fields that are set by the application program's use of macroinstruction keyword operands, a number of control block fields can be shown that are set by VTAM.

TESTCB

Tests the contents of certain fields against a value and sets the condition code in the program status word (PSW).

MODCB

Changes the contents of certain fields by inserting specified values in the fields.

Several different forms of the manipulative macroinstructions exist. In addition to the standard form, there are the following:

- List form
- Generate form
- Remote list form
- Execute form.

These alternate forms can be used by reentrant programs or by programs sharing with other programs the parameter lists created when the macroinstructions are executed. See Appendix K, “Forms of the manipulative macroinstruction,” on page 865, for the other forms of the manipulative macroinstructions.

Rather than using the manipulative macroinstructions, the program can include macroinstructions that are supplied by IBM to generate DSECTs for each kind of control block. Each DSECT shows the field names and possible values that each field can contain. These names and values can be used in assembler language instructions to set and test designated fields. See “Using DSECT-creating assembler instructions and macroinstructions” on page 272 for details.

ACB-based macroinstructions

The OPEN and CLOSE macroinstructions fall into this category. These macroinstructions inform VTAM that an application program is beginning or ending its use of VTAM services.

OPEN

Identifies an application program to VTAM and allows it to issue VTAM macroinstructions. After the program is identified, VTAM can schedule exit routines associated with the program, and accept requests for sessions with the application program.

CLOSE

Indicates to VTAM that an application program is terminating its association with VTAM and the SNA network.

RPL-based macroinstructions

These macroinstructions request session establishment, data transfer, and VTAM program operator control. They all use an RPL and, with the exception of CHECK, permit you to specify RPL modifications in the macroinstruction itself. See the following sections for a description of the RPL-based macroinstructions:

- "Session-establishment macroinstructions"
- "Session-termination macroinstructions"
- "Communication macroinstructions" on page 24
- "Macroinstructions that assist in session establishment or communication" on page 25
- "Program operator macroinstructions" on page 26

Session-establishment macroinstructions

OPNDST:

Requests VTAM to establish a session with a designated LU in which the application program acts as the PLU or requests VTAM to restore sessions pending recovery.

OPNSEC:

Informs VTAM that the application program accepts the session parameters or wishes to change the session parameter transmitted to it in a BIND request, and that VTAM should complete establishing the session in which the application program acts as the SLU.

REQSESS:

Requests VTAM to initiate a session with a designated LU in which the application program acts as the SLU. The designated LU cannot be an independent LU.

SIMLOGON:

Requests VTAM to initiate a session with a designated LU in which the application program acts as the PLU.

Session-termination macroinstructions

CLSDST:

Requests VTAM to terminate a session or reject a request for a session (CINIT) between the application program acting as the PLU and an LU.

SESSIONC:

Used by an application program to reject a BIND request, thus indicating that the application program does not wish to establish the session.

TERMSESS:

Requests VTAM to terminate a session between the application program acting as the SLU and an LU.

Communication macroinstructions

RECEIVE:

Requests VTAM to transfer a request or response to the application program's data area (if the input is data), or to appropriate fields of the RPL (if the input is control information or a response), or to both. A particular RECEIVE can be restricted to get input from a specific session or to get input from any of a group of sessions.

The RECEIVE macroinstruction allows the application program to receive unsolicited formatted requests that contain maintenance data from physical units (PUs) in the SSCP's domain, or to receive replies from PUs about requests for maintenance data.

RESETSR:

Changes the mode of receiving input for a particular session. The modes are ***continue-any*** (have input for the session satisfy an outstanding RECEIVE that accepts input from any session) and ***continue-specific*** (have input satisfy an outstanding RECEIVE that specifies only that particular session). RESETSR can also be used to cancel outstanding RECEIVEs for the specified session.

SEND:

Requests VTAM to transmit a request or response on a specific session. Data in a request is transferred from an output area in the application program; control information in a request or response is specified symbolically in the SEND macroinstruction.

When used by a communication network management application program, the SEND macroinstruction allows the application program to send formatted request units to the SSCP requesting maintenance data from PUs with which the SSCP is in session.

SESSIONC:

When it is used by an application program acting as a PLU, SESSIONC requests VTAM to send to an SLU those session control requests that do at least one of the following:

- Start or stop the exchanging of requests and responses with the SEND and RECEIVE macroinstructions
- Clear out all pending requests and responses for that session
- Assist in synchronizing request sequence numbers.

When used by an application program acting as an SLU, SESSIONC does the following:

- Requests the PLU application program to begin recovery action
- Sends a response to a sequence number request
- Sends a response to a request to start (or resume) request and response exchange with SEND and RECEIVE macroinstructions
- Sends a negative response to the BIND.

Macroinstructions that assist in session establishment or communication

CHANGE: Allows a generic resource to end the association between a session partner (LU) and an application program of the generic resource.

CHECK:

Checks and, if necessary, awaits completion of a previously requested RPL-based operation; marks as inactive the RPL associated with the operation, thus freeing it for further use; and, if a logic or other error or special condition is detected, and a LERAD or SYNAD exit routine exists, causes the appropriate routine to be entered.

EXECRPL:

Reissues a specified request. One use of this macroinstruction is to re-execute a request without changing any fields in the RPL. This is done, for example, in a SYNAD exit routine when the return code from the first attempt to perform the operation indicates that an error has occurred and that a retry is possible.

INQUIRE:

Obtains certain information that the application program might need and places it in a specified area of the program. That information includes:

- User data associated with a session-initiation request
- Session parameters associated with a particular logon mode name or with a pending active session
- Session cryptography key and initial chaining value, if cryptography is supported
- Number of active sessions and queued CINIT requests
- Status of another application program (whether it is active or inactive and whether it is accepting CINITs)
- The network-qualified name for an LU
- The name of a generic resource's application program that is associated with a specified partner LU.
- Sessions pending recovery.

INTRPRET:

Translates a character string into another character string by using an installation-defined table. For example, INTRPRET can be used to obtain the real symbolic name of an application program when the program is identified in a logon by a character string other than the name of the application program.

SETLOGON:

The forms of this macroinstruction are START, STOP, HOLD, QUIESCE, NPERSIST, PERSIST, GNAMEADD, GNAMEDEL, and GNAMESUB. Descriptions of these forms follow:

- START tells VTAM that the application program LOGON exit routine is scheduled upon the receipt of a CINIT request, and that BIND requests are sent to the application program to establish sessions in which the application program acts as the SLU.
- STOP indicates that the application program temporarily does not receive CINITs.
- HOLD paces session-establishment requests by holding the LOGON and SCIP exits from CINIT and BIND requests until a SETLOGON OPTCD=START request is issued.
- QUIESCE indicates that the application program wants to stop establishing sessions.
- The application uses PERSIST to enable persistence and NPERSIST to disable persistence.
- The application uses GNAMEADD to create an association between the network name and the generic name. It uses GNAMEDEL to delete that association. An application can be a subordinate of another application that is using a generic name. In this case, the subordinate application uses GNAMESUB to have its sessions included in the other application's session count for workload balancing.

Program operator macroinstructions

The following program operator macroinstructions allow an authorized application program (called a program operator) to do the following:

- Issue VTAM operator commands (except START and HALT) and the operating system reply command
- Receive operator messages from VTAM.

RCVCMD:

Receives an unsolicited VTAM operator message or receives replies to commands that are issued using SENDCMD.

SENDCMD:

Enters a VTAM operator command or the operating system reply command.

Refer to Appendix L, “Program operator coding requirements,” on page 875, for information on the program operator interface.

Relationship between the executable macroinstructions and control blocks

The relationships among the VTAM control blocks, as well as their relationship to the macroinstructions that refer to them, are described in the context in which they are used. To establish that context, the following sections describe the relationships and use of the control blocks in terms of the operations that every application program must perform:

- Opening and closing the application program
- Establishing and terminating sessions

- Communicating over sessions

Opening the application program

The OPEN macroinstruction associates an application program with VTAM so the application program can use VTAM facilities. The OPEN macroinstruction specifies an ACB; the ACB, in turn, points to a location in the program that contains the name of the application program as specified in an APPL definition statement during VTAM definition. The ACB can also point to an EXLST control block containing the names of exit routines that are to be associated with the application program. An EXLST can also be pointed to when a session is established with an LU. See "Establishing sessions with LUs" in this chapter. When the open process has completed, exit routines specified in the EXLST are eligible for scheduling by VTAM.

Because a program can open multiple ACBs, a program that performs related functions (for example, communicating with LUs or acting as a program operator application program) can be defined in such a way that VTAM views the program as being more than one application program. However, most VTAM users might find it satisfactory to open only one ACB for each program.

Establishing sessions with LUs

Before communicating with an LU, an application program must be in session with the LU. A session can be initiated by the LU, by the VTAM operator, by VTAM, or by an application program. Regardless of how the session is initiated, if the application program is to be the PLU in the session, it formally establishes the session by issuing an OPNDST macroinstruction. The OPNDST macroinstruction specifies an RPL that contains the address of an NIB. The NIB contains information that applies to subsequent communication with the LU for that session. If necessary, the address of a unique storage area to be associated with the session can be specified in the NIB. This area could include an I/O area and a place for flags that keep track of communications on the session. If a number of sessions are to be initiated by an application program, a single SIMLOGON or a single OPNDST OPTCD=ACQUIRE can be used, and the RPL can point to a list of NIBs instead of to a single NIB.

Optionally, for certain types of exit routines (DFASY, RESP, and SCIP), an NIB can point to a list of exit-routine names in an EXLST control block. For the session being established, these exit routines are generally used in preference to the corresponding exit routines identified for the entire application program when the ACB was opened.

When a session is established as the result of an OPNDST macroinstruction, VTAM returns information about the session in the RPL and the NIB. In both the RPL (usually) and the NIB (always), VTAM places a communication identifier (CID) that it has assigned to the session with the LU. On all subsequent communication requests for the session, the application program must be sure that this CID is present in the RPL. In addition to the CID, VTAM also places the LU name (for an OPNDST OPTCD=(ACCEPT,ANY)) and other information in the NIB; if desired, the application program can use this information to determine how to communicate on this session. Once an NIB is used for session establishment, it can be reinitialized and reused to establish another session.

SIMLOGON and OPNDST initiate and establish sessions in which the application program acts as the PLU. REQSESS and OPNSEC initiate and establish sessions in which the application program acts as the SLU. In general, REQSESS and OPNSEC use the RPL and NIB in the same manner as do SIMLOGON and OPNDST.

Communicating with LUs

After opening its ACB and establishing sessions with one or more LUs, the application program can communicate on each session by issuing SEND, RECEIVE, SESSIONC, and RESETSR macroinstructions. VTAM obtains the name of the application program that made the request and the identity of the session (if a specific session is being addressed) from the RPL. The communication macroinstruction specifies an RPL; the RPL contains the address of an ACB and the identity of the session.

The SEND and RECEIVE macroinstructions write and read requests and responses. A request contains data or control information or both. A response contains information that tells whether a request requiring a response arrived at the destination LU and was processed successfully or unsuccessfully.

Only data is written from or read into an application program data area. Control information is sent by being specified symbolically in a SEND macroinstruction or its associated RPL. Control information and responses that are received are not read into a data area, but are detected by analyzing fields in the RPL associated with a RECEIVE macroinstruction or in an RPL associated with the scheduling of a special exit routine that handles the receipt of responses and control requests.

SESSIONC is used to send session control requests and responses. The RESETSR macroinstruction is used to control the continue-any and continue-specific modes of a session. Both macroinstructions use the RPL and ACB control blocks in the same way as SEND and RECEIVE to identify the session and application program LU.

Terminating sessions with LUs

When a session is no longer needed, the application program terminates it by issuing the appropriate macroinstruction. If the application program is the PLU, it issues CLSDST. If the application program is the SLU, it issues TERMSESS.

Closing the application program

The application program issues a CLOSE macroinstruction to disassociate itself from VTAM. CLOSE specifies the same ACB that was originally used with OPEN. If all sessions are to be terminated at the same time, the program can issue a single CLOSE macroinstruction instead of first issuing a series of CLSDST and TERMSESS macroinstructions to terminate sessions individually. As a result of the CLOSE macroinstruction, VTAM terminates each application program session.

Exit routines

VTAM allows use of exit routines by which a VTAM application program can gain control to handle certain conditions. An exit routine handles a specific event. When that event occurs, VTAM gives control to the exit routine as soon as possible.

Following are two kinds of application program exit routines:

Exit-list (EXLST) exit routines

These are special-purpose exit routines that VTAM schedules as needed. The exit list, created with the EXLST macroinstruction, specifies the exit-routine addresses (entry points). A program can have more than one exit list. An exit list can be specified in an ACB and thus be used by VTAM when an exit-routine event occurs for any session with the program. For certain exit routines—DFASY, RESP, and SCIP—an exit list can be specified in an NIB and can be used by VTAM only when an exit-routine event occurs for the session associated with the NIB. See Figure 40 on page 180 and Figure 41 on page 181 for more information.

RPL-specified exit routines

These are exit routines that contain instructions to be executed when particular RPL-based operations are completed. In any individual session-establishment, communication, or other RPL-based macroinstruction, if an RPL exit-routine address is specified, the exit routine is scheduled as an alternative to VTAM's posting an ECB when the requested action is completed. A program can use a mixture of ECB-posting and RPL exit routines, or it can use solely one or the other.

The names of the special-purpose exit routines and the events that schedule them are summarized in Table 1.

Table 1. Summary of special-purpose exit routines

Exit routine name	Event
ATTN	One of the following LU 6.2 events has occurred: • VTAM receives an FMH-5 for the application program. • VTAM processes a CNOS request for the application program. • The last LU 6.2 session, or all LU 6.2 sessions, with another LU using a given mode name group is lost.
DFASY	An expedited data-flow-control request has been received on a session.
LERAD	A logic error has been detected following an RPL-based request.
LOGON	An application program is being requested to establish a session as a PLU.
LOSTERM	A session with an LU has been temporarily interrupted or permanently lost; the LU or VTAM operator requested that the session be terminated; or an event occurred that can affect future operation of the session.
NSEXIT	A network services request unit has arrived for the application program, indicating, for example, that: • A session with an LU has been terminated because of a disruption of the path used by the session • A session-establishment procedure has completed or failed.
RELREQ	Another application program has requested a session with an LU that is presently in session with this program.
RESP	A response has been received.

Table 1. Summary of special-purpose exit routines (continued)

Exit routine name	Event
SCIP	One of the following session-control requests has been received: • Clear (CLEAR) • Start Data Traffic (SDT) • Request Recovery (RQR) • Set and Test Sequence Numbers (STSN) • Bind Session (BIND) • Unbind Session (UNBIND).
SYNAD	An error other than a logic error has been detected for an RPL-based request.
TPEND	The VTAM operator is shutting down the network or shutting down this application, VTAM has abended, or an alternate application is taking over sessions from an application that has enabled persistence.

Normal operating system environment for a VTAM application program

A VTAM application program directly relates to an ACB for which an OPEN macroinstruction successfully issues. A program which runs under the control of a host operating system can open several ACBs and thus appear as multiple VTAM application programs.

On the other hand, a VTAM application program can be divided among several tasks controlled by the operating system. This is discussed further in Chapter 10, "Operating system facilities," on page 295.

The operating systems under which VTAM runs offer several facilities (for example, multitasking and service request blocks (SRBs)) that allow you to structure your programs for more efficient operation.

By using the 31-bit addressing facility, you can use virtual storage more efficiently.

However, for most VTAM application programs, a less complicated operating system environment, called the normal environment, is sufficient. The following sections describe the normal environment. This book assumes that the environment is the normal environment, except where explicitly stated otherwise. Some of the ways an application program can use the special operating system facilities are described in Chapter 10, "Operating system facilities," on page 295. That chapter also describes how the environment is changed from the normal environment by the use of the special facilities.

Use of a single task

In the normal operating system environment, a VTAM application program runs under a single operating system task. All of the mainline application program instructions and all of the VTAM exit routine application program instructions run under that task. When performing services for the application program, VTAM itself executes at times under the application program task, and at times under the control of system tasks that are created by VTAM.

Mainline program

The mainline part of the application program issues the OPEN and CLOSE macroinstructions. The mainline program executes until an event occurs that gives control to an exit routine. Exit routines are either inline exit routines (LERAD and SYNAD) or asynchronous exit routines (all RPL exit routines and all EXLST exit routines except LERAD and SYNAD). The following sections give an overview of exit routines in the normal environment. See Chapter 7, "Using exit routines," on page 219, for more information on exit routines.

The normal operating system environment described in this book is a VTAM application program running in 24-bit addressing mode and resident in 24-bit storage. See Chapter 10, "Operating system facilities," on page 295, for special considerations when an application program uses 31-bit addressing.

Inline exit routines

An inline exit routine is considered to be an extension of the part of the application program (either mainline or asynchronous exit routine) that is executing when the inline exit routine is invoked. Effectively, an inline exit routine is entered through a branch from an RPL-based or CHECK macroinstruction just before that macroinstruction returns to the next sequential instruction in the application program.

After it completes processing, an inline exit routine can return to VTAM. If it is coded to return to VTAM, the application program receives control at the next sequential instruction immediately after the RPL-based or CHECK macroinstruction that caused the inline exit routine to be invoked. If the inline exit routine is coded not to return, the application program branches to another location after issuing the RPL-based or CHECK macroinstruction.

Asynchronous exit routines

Asynchronous exit routines, in contrast to inline exit routines, do not act as extensions to the part of the application program that was executing when the event associated with the exit routine occurred. The events that cause invocation of asynchronous exit routines are unpredictable, whereas inline exit routines can be invoked only at predictable points (that is, immediately after the associated CHECK or RPL-based macroinstruction).

Asynchronous exit routines can interrupt the mainline program at any time, even if the mainline program is currently suspended (for example, because it issued a CHECK or WAIT macroinstruction). However, with the exception described in the next paragraph (TPEND with reason code 8), no asynchronous exit routine can interrupt another asynchronous exit routine; thus, each asynchronous exit routine must return to VTAM before the next asynchronous exit routine can be given control. When an asynchronous event occurs, the associated exit routine (if defined by the application program) is scheduled by VTAM. If the mainline program is currently in control, execution of the mainline program is suspended and control is immediately given to the exit routine. If another asynchronous exit routine is in control, that exit routine must return to VTAM before the next exit routine can be given control. If the asynchronous exit routine currently in control suspends execution (for example, by issuing CHECK or WAIT), it prevents other asynchronous exit routines from gaining control. When the final asynchronous exit routine returns to VTAM, the mainline program resumes control at the point where it was interrupted.

The TPEND exit routine does not obey the preceding rules. It can interrupt any part of the application program, including another asynchronous exit routine, at any time. It must return to VTAM before the interrupted asynchronous exit routine or mainline program can resume control.

Dispatching priorities

The order of dispatching priority in the normal operating system environment is:

- TPEND exit routine with reason code 8
- All other asynchronous exit routines (RPL and EXLST exit routines, except LERAD and SYNAD)
- Mainline program.

Inline exit routines assume the dispatching characteristics of the part of the application program from which they were invoked.

Chapter 3. Organizing an application program

The organization of a VTAM application program affects how much storage it uses, how well it performs, how easy it is to write, and how easy it is to migrate the application program to another release of VTAM, if required. Before you begin writing a VTAM application program, it might be helpful to understand some of the ways in which an application program can be organized. This chapter discusses:

- General coding guidelines to consider when writing an application program
- Coding guidelines to facilitate migration of application programs
- Whether the application program should be single-thread or multithread
- Whether the application program operation should be synchronous or asynchronous and what posting mechanism should be used
- VTAM and user control block decisions to be made that affect application program organization

Chapter 10, "Operating system facilities," on page 295, discusses additional options available to the application program.

Coding guidelines for application programs

When writing a VTAM application program, use the guidelines in "Program structure recommendations" to minimize the programming changes needed when:

- The application program is moved to another release of VTAM.
- The application program is moved from a single-domain environment to a multiple-domain environment.
- New SNA products or other SNA enhancements are introduced into the network.
- The VTAM application program is updated (for instance, when program temporary fixes (PTFs) are applied).

Note: A detailed description of migration considerations is contained in Appendix N, "Application program migration," on page 893.

Program structure recommendations

When coding the application program or changing application-program code, follow these recommendations:

- Code SCIP, NSEXIT, and LOSTERM exit routines for the application program. If the application program does not have the applicable exit routines, notification of session outage or session disruption might not occur, and you might no longer be able to communicate with the network resource. See "Exit routines related to session establishment and termination" on page 100 and "Session outage notification" on page 110 for information about these exit routines.
- When using multinode persistent sessions (MNPS), structure the application so that the OPEN ACB macroinstruction is done early in the application processing. This allows VTAM, during recovery OPEN processing, to begin the recovery process (rebuilding sessions) in parallel with application setup processing.

© Copyright IBM Corp. 2000, 2013

- Reinitialize all VTAM control blocks to the values provided by VTAM before reusing the control block. Failure to do so can result in the application program not being able to complete macroinstructions issued later that reference one of the control blocks. For example, if you are reusing an RPL or NIB for another session, reinitialize the control block fields before issuing the OPNDST macroinstruction to open the session.
- Code an NSEXIT exit routine (see Chapter 7, "Using exit routines," on page 219) in application programs that issue SIMLOGON, REQSESS, TERMSESS, or CLSDST OPTCD=PASS macroinstructions. This exit routine receives Notify or NSPE request units. These macroinstructions are posted as complete when processing is started; however, processing might fail later, particularly in a cross-domain environment. The only notification of such an error is through the NSEXIT exit routine. Without an NSEXIT exit routine, the session-initiation or termination request cannot be completed until VTAM is terminated.
- When using this system, specify the following program attributes: an addressing mode (AMODE) of 31 and a residence mode (RMODE) of ANY. These specifications allow the application program to run in 31-bit addressing mode and to reside anywhere in 31-bit storage, wherever space is available. See Chapter 10, "Operating system facilities," on page 295, for more information about 31-bit addressing. For coding or referencing application programs with programming attributes other than those recommended above, see z/OS MVS Programming: Assembler Services Guide.
- When your application program shares printers with other IBM subsystems (for example, CICS® and IMS™) or other application programs, make provisions within the application program to end the printer session without operator intervention. This can be accomplished by one of the following procedures:
  - Terminate the session with the printer when there is no output waiting to be printed (or when there has been no output requested for a given length of time).
  - Include a RELREQ exit routine (described in Chapter 7, "Using exit routines," on page 219) in the application program.
  - If the subsystem has a generation procedure, specify that the printer resource is (or is not) released when the application program's RELREQ exit routine is driven.
  - Use the SIMLOGON macroinstruction with OPTCD=RELRQ,Q options to initiate the session with the printer resource. RELREQ does not apply if the primary logical unit (PLU) is an independent LU.
- Whether the PLU application program receives a session-initiation request or initiates a session, specific session parameters might need to be requested. To specify a session parameter, use the rules in Table 2.

Table 2. Rules for specifying a session parameter

Program action	Session initiated by PLU application program	Session not initiated by PLU application program
PLU application program modifies default parameters	Allow generated name or name in master terminal requirements. Issue OPNDST OPTCD=ACCEPT with BNDAREA.	Issue an INQUIRE OPTCD=SESSPARM in the LOGON exit routine. Then issue OPNDST OPTCD=ACCEPT with the new parameters.
PLU application program accepts default parameters	Allow generated name or name in master terminal requirements. Issue OPNDST OPTCD=ACQUIRE. Or, issue SIMLOGON with name and then issue OPNDST OPTCD=ACCEPT with LOGMODE=0.	Issue an INQUIRE OPTCD=SESSPARM in the LOGON exit routine. Then issue OPNDST OPTCD=ACCEPT.

Table 2. Rules for specifying a session parameter (continued)

Program action	Session initiated by PLU application program	Session not initiated by PLU application program
PLU application program uses CLSDST OPTCD=PASS	Specify the name to establish a new COS.	Issue SIMLOGON for the desired resource and interrogate the session parameters in the LOGON exit routine. Then issue CLSDST OPTCD=PASS with AREA containing the appropriate session parameter.

Simplifying migration and network upgrades

In addition to the program structure recommendations listed in the preceding section, the following guidelines help to ease future migrations and network upgrades.

- Leave reserved fields in VTAM control blocks set to the value given to them by VTAM. The control blocks are described in Appendix E, "Control block formats and DSECTs," on page 739.
- Do not reference the labels or contents of reserved fields in VTAM control blocks. Such fields do not contain information for the application program.
- Do not create application program dependencies on internal VTAM functions, such as SVCs, macroinstruction expansions, or internal VTAM control blocks. Use only the VTAM facilities described in this book.
- Code the application program to allow for fields to be added to the end of VTAM control blocks. Use the IBM-supplied DSECTs to determine control block size at assembly time. Use the GENCB and SHOWCB or TESTCB manipulative macroinstructions to determine control block size at execution time.
- Use the IBM-supplied DSECTs and manipulative macroinstructions to access control block fields, rather than coding actual displacements into control blocks.
- Code the application program to allow for fields to be added to the end of the SNA RUs that can be presented to the application program (BIND, UNBIND, CINIT, NOTIFY, NSPE, CLEANUP).
- Code the application program so that it is not dependent upon the sequence of vectors or the length of vectors (for example, the access-method-support vector list).
- Code the application program's NSEXIT exit routine to handle unrecognized input. See "NSEXIT exit routine" on page 244 for a description of this routine.

  Check for specific input (for example, reason code, RU type, and sense data), and avoid assuming that the input has a certain value or type, when a test for another value or type fails.
- Code the application program to allow for new ERROR codes for OPEN and CLOSE, new feedback codes (RTNCD,FDB2) for RPL-based macroinstructions, and new SNA sense codes. If an unknown code is received, it can be saved for program debugging; the associated operation should be considered to have had a permanent failure.

  The OPEN macroinstruction is discussed in detail in "OPEN—Open one or more ACBs" on page 444. The CLOSE macroinstruction is discussed in "CLOSE—Close one or more ACBs" on page 390. A summary of the valid feedback codes can be found in Figure 104 on page 652.
- Code the application program to allow for new UNBIND type codes. Refer to Chapter 5, "Establishing and terminating sessions with logical units," on page 81

81, for a complete description of the UNBIND function. You should treat any unknown type code as a normal UNBIND (X'01').

- Use the function-list vector and macro global variables to determine the presence or absence of specific VTAM facilities. These can be found in Chapter 4, "Opening and closing an application program," on page 55, and Chapter 8, "Setting and testing control blocks and macro global variables," on page 267 respectively.
- Code the application program to allow for a multiple-domain network. This facilitates possible future migration from a single-domain network. Appendix N, "Application program migration," on page 893, contains detailed information about coding considerations related to migration.
- Code the application program to handle all LU type 0 3270 terminals the same; do not become dependent on attachment and implementation differences among the various 3270 terminals. See "Summary of differences among LU type 0 3270 terminals" on page 335 for suggestions about how to accomplish this.
- Code the application program to handle UNBIND with control vector hex 35. The UNBIND that is received can contain a control vector hex 35 that contains a sense code that reflects why a session was terminated or why the session was not established. Also, control vector hex 35 identifies the failing node.

Single-thread or multithread operations

A VTAM application program is either a single-thread program—capable of processing the request of only one session at a time—or a multithread program—capable of processing the requests of many sessions concurrently. These terms describe how the application program works in general. In practice, many single-thread programs can do some overlapping of processing, and many multithread programs can do some processing that momentarily ties up the application program for an action on behalf of only one session. In general, a single-thread program requests synchronous operations and waits until each operation is completed before continuing. A multithread program requests asynchronous operations and continues processing on behalf of other sessions while waiting for an operation for a particular session to be completed.

Using a single-thread program

A single-thread program is easier to design and code than a multithread program. Sample Program 1 in Chapter 14, "Logic of a simple application program," on page 573, is basically a single-thread program.

You should use a single-thread design when the application program never handles more than a few sessions at a time or when, if it handles more than a few, response time is not a consideration. Additionally, use a single-thread design for an application program that does nothing more than send a continuous series of requests to an LU (which might in turn forward the data to a printer or to a data base on a disk) or receive a continuous series of requests from an LU (perhaps from the disk associated with an LU) and write them to a database on disk storage at the host processor.

Using a multithread program

You should use a multithread design for any VTAM application program that must communicate concurrently on a number of sessions. This implies the use of

asynchronous operations—determining completion of operations either by having VTAM post an ECB or by having it schedule an RPL exit routine.

In a multithread program, the control blocks for each session must be managed efficiently. The control blocks reflect the status of the session (for example, whether the LU has begun to communicate on that session, what address is to be branched to when a requested output operation is completed on that session, or whether the LU has sent in a logoff message for that session). Sample Program 2 in Chapter 16, "Logic of a more complicated application program," on page 605, shows the general logic of a multithread program.

You can use multitasking to transfer control between the communication and data processing parts of an application program. See Chapter 10, "Operating system facilities," on page 295, for more information on data separation using the multitasking facility. It is also possible for the same routines to be shared among what VTAM perceives as more than one VTAM application program. This arrangement can be used for communicating with two different types of LUs. Two ACBs can be defined in an application program. One kind of LU is associated with control blocks that point to one ACB while another kind of LU is associated with the other ACB. Because VTAM sees each ACB as an application program, each type of LU can have separate logic associated with it, including its own exit routines and its own I/O routines. Data processing parts of the application program, a wait routine, and other routines can be shared.

Multithreading facilities

In addition to the asynchronous handling of input and output requests, VTAM also provides the following facilities as aids to handling sessions in a multithread program:

- A special field that is used to associate a unique storage area with each session
- The ability to schedule the sending of a request
- The ability to receive input from any session except those sessions that are specifically precluded.

These facilities are discussed in more detail in the following sections.

USERFLD field of the NIB

In handling a series of input and output actions for a particular session, the application program might need some way of associating a particular piece of information with the session. For example, the application program might need to know:

- The LU associated with the session
- The city in which the LU is located
- The LU type
- The symbolic name that the application program uses
- The storage area containing the input buffer and other application-program-manipulated control information about operations with this LU.

If this information does not change (for example, the city in which the LU is located), the information can always be available, and assembled into the application program. More frequently, however, the information that the application program wants to associate with the LU is not available until after the

application program starts execution or it changes during program execution. For the dynamic information, the application program needs a mechanism for associating the desired piece of information with the session.

The mechanism provided by VTAM involves the user field of the NIB, which contains space for 4 bytes of information. Whatever information is in the user field of the NIB at the time the session with the LU is established is saved by VTAM, and whenever input is subsequently received on the session, that 4 bytes of information is provided in the user field of the RPL used for the operation. This mechanism has many uses, including those described in the following paragraphs:

- **Identifying the session from which input has been received.** Each time the application program establishes a session with a logical unit, the application program puts its own version of the identification of the session into the USERFLD field of the NIB before the OPNDST or OPNSEC macroinstruction is issued. Later, the application program issues RECEIVE OPTCD=ANY, which accepts an input request from any session. When an input request is received, the application program examines the user field of the RPL to determine the session from which the input request came.
- **Associating a storage area with a session.** For each session, the application program might want to have a session-associated storage area that contains an RPL, possibly an ECB (if ECB-posting is used), a data area (to be used as a buffer for input and output requests), and a status information area. When the application program establishes a session, it specifies in the USERFLD field of the NIB the address of the storage area it wants to be associated with that session. VTAM saves this address and, when input is received on the session, VTAM places the address in the user field of the RPL. The application program can use the user field contents to process the input rather than having first to identify the session.
- **Identifying the specific REQSESS or SIMLOGON in a SCIP or LOGON exit routine.** Application programs can issue multiple SIMLOGON requests for the same secondary logical unit (SLU); likewise, an application program can issue multiple REQSESS macroinstructions for the same PLU. Each time an application program issues a REQSESS or SIMLOGON macroinstruction, VTAM saves the NIB's USERFLD contents. The USERFLD is made available to the application program in the exit parameter list for the corresponding SCIP, LOGON, or NSEXIT exit routine. The application program can then associate the USERFLD contents with a previously issued REQSESS or SIMLOGON macroinstruction, and thus associate the session with the original request for that session.

Scheduling output

VTAM allows an application program to ask that a request be scheduled for sending, and that the operation be considered complete as soon as the request has been scheduled for output rather than when the request is actually sent with arrival confirmed by the receiving LU. If the application program wants to determine whether the request actually arrived, it can, as part of the output scheduling request, specify that a definite response be returned by the LU. On receiving the response, the application program knows that the request arrived successfully or unsuccessfully. (For many LUs, the return of a positive response indicates not only that the request arrived successfully but also that it was processed successfully.) Because scheduling output usually takes relatively little time, a request to schedule the sending of a request can often be specified as a synchronous operation. However, because, under some circumstances, the scheduling might take a long time (for example, waiting for pacing responses), synchronous operation should not be used if you do not want the sending task or

SRB to wait. Scheduled output can also be specified as an asynchronous operation with ECB-posting or RPL exit-routine scheduling specified.

A good approach is to request ECB posting and to test the ECB immediately upon return from VTAM; usually the ECB is already posted complete, so the RPL can be checked right away. Under unusual circumstances, the ECB might not yet have been posted; however, because only a test was done (no WAIT), the application program is free to continue with other work and to test or WAIT on the ECB at some later time.

In scheduling output, the application program might not require that a response be returned to every request; it might require that a response be returned only to the last in a series of requests. Receipt of a positive response confirms successful arrival and processing of the request or series of requests, while receipt of a negative response indicates an error. Successful arrival and processing of a request can also be assumed if the resultant input request contains what the application program expects or, in the event of an error, it can be assumed that a terminal operator notifies the application program that he or she is waiting for a request that has not arrived.

By scheduling the sending of a request, the application program reserves for itself the determination of whether confirmation of arrival and processing is necessary. When fewer responses are requested, greater request throughput is possible. The user, however, does not have complete control, because SNA protocols dictate when some responses must be requested.

Receiving input on any session except those already in communication

About this task

VTAM provides a way of receiving input on any session. To do this, RECEIVE OPTCD=ANY is issued. On completion, the identity (the CID) of the session from which input has been received is in the RPLARG field associated with the RECEIVE request. (The INQUIRE macroinstruction with the CIDXLATE option can be used to translate the CID into the symbolic name of the LU with which the application program has that session.) Typically, RECEIVE OPTCD=ANY is issued to receive the initial input that leads to communication in a particular session with a LU.

Once RECEIVE OPTCD=ANY has been used to get initial input on a session, that session can be switched to another mode called continue-specific mode. When a session is in this mode, a request on that session does not satisfy RECEIVE OPTCD=ANY; the request can satisfy only a RECEIVE OPTCD=SPEC whose RPL identifies the session on which the request was received. While the session is in continue-specific mode, the application program maintains specific control over each request sent or received on the session.

Thus, an application program can consist of a single RECEIVE OPTCD=ANY that is reissued each time it is completed, and of sets of specific RECEIVE and SEND macroinstructions, with each set of specific macroinstructions controlling communication on a particular session. To obtain the continue-specific facility, OPTCD=CS is specified in the request at the point at which the session is to be switched to continue-specific mode. For example, the RECEIVE that reads input on any session (except those already in continue-specific mode) specifies OPTCD=(ANY,CS). This places the session whose input satisfied RECEIVE in

continue-specific mode; the next issuance of RECEIVE OPTCD=ANY excludes this session from being able to complete RECEIVE. Chapter 16, “Logic of a more complicated application program,” on page 605, shows use of RECEIVE OPTCD=(ANY,CS).

How a synchronous operation works

In a synchronous program, operations are performed serially. A request for synchronous operation (for example, SIMLOGON, SEND, or RECEIVE OPTCD=SYN) means that VTAM does not return control to the next sequential instruction in the application program task or SRB (under MVS) from which the macroinstruction was issued until after the requested operation is completed. See “Serialization of execution” on page 309 for information about running several parts of an application concurrently. Figure 8 illustrates a synchronous operation.

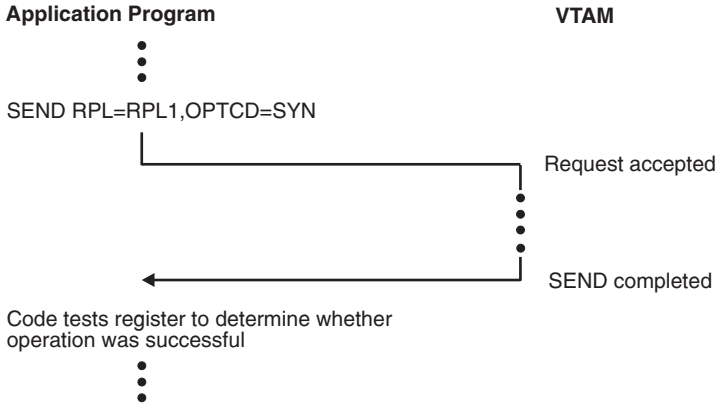


Figure 8. Synchronous operation

Note: While the application program is waiting for the event to be completed, an asynchronous event such as a HALT command could cause the application program's TPEND exit routine to be entered. Only the application program task or SRB (under MVS) from which the macroinstruction was issued is suspended while waiting for completion of a synchronous operation. The exit routines associated with the application program are scheduled and executed regardless of whether the mainline program logic is awaiting completion of a synchronous operation.

In general, avoid issuing a synchronous request in a task because it suspends all execution under the task until that request completes. Also avoid issuing a synchronous request within an exit routine identified in an ACB exit list. For more information on ACB EXLST exit routines, see Chapter 7, “Using exit routines,” on page 219.

Designing an application program to use synchronous requests while running under an SRB is more reasonable because the number of system resources forced to wait for the completion of the operation is more limited. Refer to “Synchronous versus asynchronous operations” on page 169 for further information. The differences between the TCB and SRB modes of execution are described in detail in “Execution of exit routines” on page 307.

When a synchronous operation is completed, the application program must determine whether the operation was successful or unsuccessful. The application program does this by testing values in registers 15 and 0 and by examining fields in the RPL used for the operation. For more information on testing return codes

from RPL-based macroinstructions, see Chapter 9, “Handling errors and special conditions,” on page 277, and Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651.

How an asynchronous operation works

In an asynchronous operation, VTAM returns control to the next sequential instruction as soon as it has accepted the request, not when the requested operation has been completed. Accepting a request consists of screening the request for errors and scheduling the parts of VTAM that eventually carry out the operation. While the operation is being performed, the application program is free to initiate other data-transfer operations or do other processing. For example, an application program can issue a RECEIVE macroinstruction and indicate that the operation is to be handled asynchronously (OPTCD=ASY). While the input operation is being performed, the application program can begin to write to a direct-access storage device or receive input on another session.

When an asynchronous operation is specified, there are two ways that VTAM can notify the application program that the requested operation has been completed. If the application program associates an event control block (ECB) with the request, VTAM posts the ECB when the operation is completed. Alternatively, the application program can designate that a particular RPL exit routine is to be executed as soon as the operation is completed. When the operation is completed, VTAM schedules the exit routine. The method of notification is controlled by the setting of the ECB operand or EXIT operand in the RPL used for the request. Figure 9 on page 42 illustrates asynchronous processing in an application program using ECBs; Figure 10 on page 43 illustrates the use of an RPL exit routine.

Regardless of whether an application program waits on an ECB or uses an RPL exit routine, a CHECK macroinstruction must be issued after an asynchronous operation to mark the RPL inactive and to make it available for another operation. The CHECK macroinstruction also clears the ECB. Prior to CHECK, the RPL active flag is still on. The RPL and associated data areas (for example, NIB, RPLAREA, and ECB) cannot be freed, altered, or reused.

Using ECBs

By using ECBs, the application program can issue one WAIT macroinstruction for a combination of VTAM requests and any non-VTAM requests that use ECBs.

For example, an application program can issue three VSAM requests and three VTAM requests. By issuing one WAIT for all six ECBs, the application program resumes processing when any one of the six operations is completed.

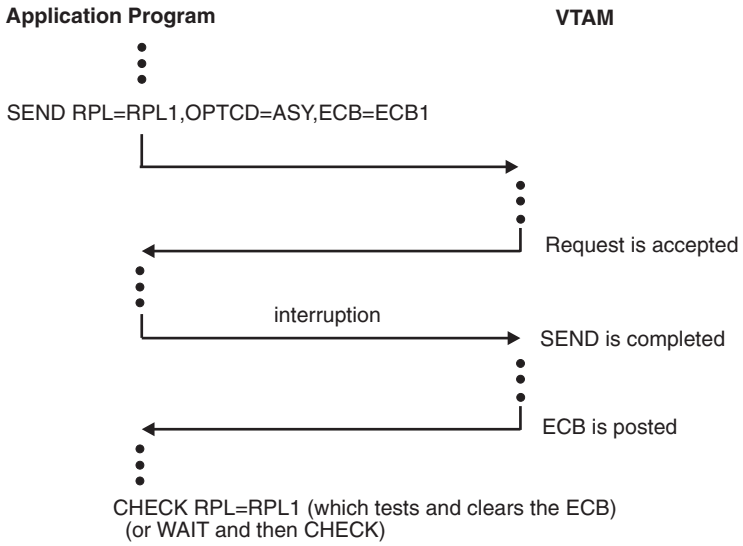


Figure 9. Asynchronous operation with an ECB posted

Using ECBs, the application program can test ECBs itself and continue to wait only if no ECB has been posted. The application program can prioritize requested operations for LUs by testing some ECBs before testing others. The order of checking can be varied during program execution as circumstances change.

The ECB and RPL exit routines are different. The RPL exit routine schedules automatically when the requested operation completes, thereby saving the application program the trouble of testing ECBs and branching to subroutines. ECBs provide the application program with greater control over the order in which events are handled.

If neither an ECB address nor an RPL exit-routine address is specified in the RPL-based macroinstruction, VTAM uses the ECB-EXIT field of the RPL as an internal ECB, and VTAM (for synchronous operations) or the user (for asynchronous operations) checks and clears it. The ECB-EXIT field can be set to point to an external ECB by using an RPL-based macroinstruction that specifies ECB=*ecb address*. Once set, it can be reset to an internal ECB by using an RPL-based macroinstruction that specifies ECB=INTERNAL.

Using RPL exit routines

Instead of having VTAM post an ECB when a request for an asynchronous operation is completed, the application program might have VTAM schedule and cause control to be given to an RPL-specified asynchronous exit routine. The RPL exit routine can supply the logic that would have been branched to by the mainline program after discovering a posted ECB. An RPL exit routine is any exit routine whose symbolic name has been provided in the EXIT operand of the macroinstruction or in the RPL used for the request.

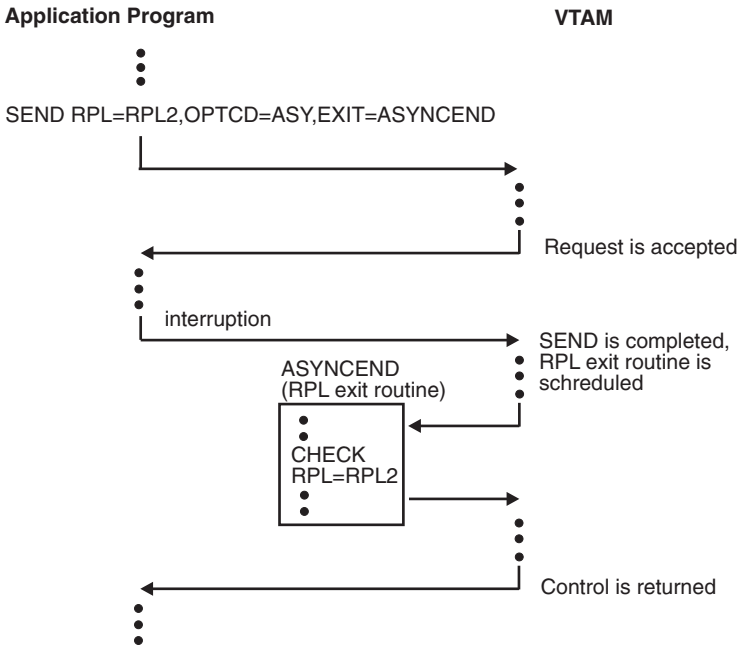


Figure 10. Asynchronous operation with an RPL exit routine scheduled

One advantage to using an RPL exit routine instead of an ECB is that it is easier to code for that type of processing than it is to code the logic associated with discovering a posted ECB and relating the ECB to a branch address. Also, the RPL exit routine is given control almost immediately after the associated RPL-based operation completes, and thus has priority over the mainline program. A disadvantage of an RPL exit routine is that more system instructions must be executed to schedule an exit routine than must be executed to post an ECB. Also, as discussed in the previous section, RPL-exit scheduling does not provide as much flexibility as ECB-posting for giving a higher priority to selected operations. An application program can use a combination of ECB-posting and RPL exit routines. See Chapter 16, “Logic of a more complicated application program,” on page 605, for more information on the general logic of a multithread program.

An RPL exit routine can itself issue asynchronous requests, continue executing, and return to VTAM. The asynchronous request in an RPL exit routine can specify that, upon completion of the request, an ECB is to be posted or an RPL exit routine is to be scheduled. If the RPL exit routine option is chosen, the exit routine might be the same one in which the request was issued. (This is also shown in Sample Program 2 in Chapter 16, “Logic of a more complicated application program,” on page 605.) Figure 11 on page 44 shows a possible pattern of asynchronous requests within RPL exit routines. Further information about RPL exit routine operation is given in “Normal operating system environment for a VTAM application program” on page 30 and in “RPL exit routines” on page 309.

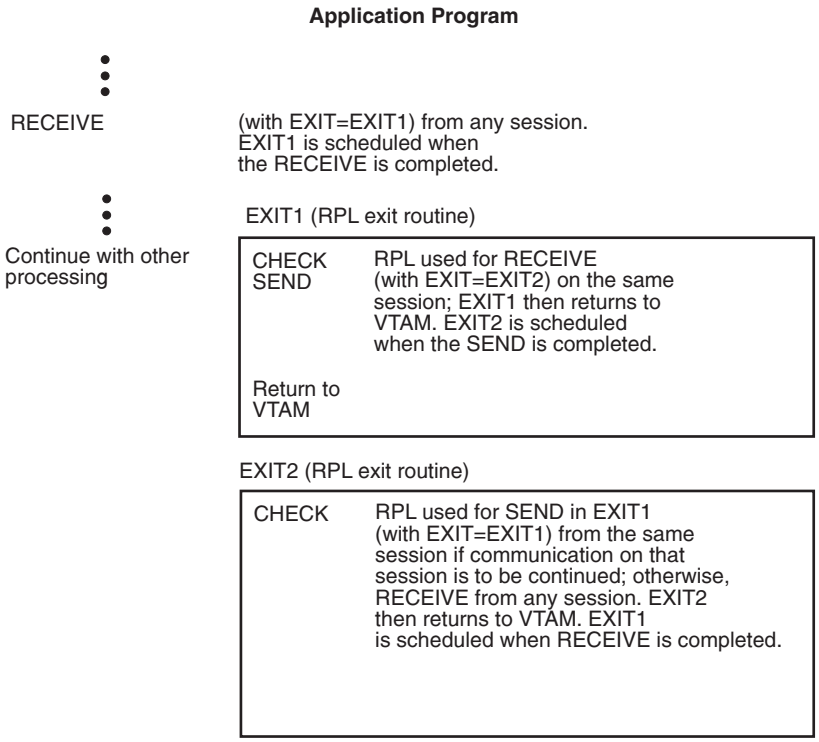


Figure 11. Possible pattern of asynchronous requests in RPL exit routines

Initializing a session

During session initiation, error recovery procedure (ERP) processing can cause an application program task to be indefinitely suspended. This happens when an RPL exit routine is not coded and either SIMLOGON OPTCD=ASY or OPNDST OPTCD=(ACQUIRE,ASY) is followed by a CHECK macroinstruction for the associated RPL. Neither the SIMLOGON nor the OPNDST operation completes until ERP processing is finished. The CHECK macroinstruction causes the task that issued CHECK to be suspended for as long as it takes to complete the ERP processing.

To avoid suspension, use SIMLOGON OPTCD=ASY, and code an RPL exit routine to handle its completion. Then use OPNDST OPTCD=ACCEPT after the LOGON exit routine has been scheduled.

Advantages and disadvantages of different forms of operation

Table 3 on page 45 summarizes the advantages and disadvantages of synchronous operations and the two general forms of asynchronous operations, ECB-posting and RPL exit-routine scheduling.

Table 3. Relative advantages of synchronous and asynchronous requests

Type of request	Performance	Storage requirements for RPLs and data areas	Programming complexity
Synchronous (OPTCD=SYN)	Adequate for batch-type program or for programs serving few simultaneous sessions.	Small, because only one request can be outstanding at a time. Can reuse RPL and data areas.	Simplest to code.
Asynchronous (OPTCD=ASY) • ECB posting (ECB= *address* or INTERNAL)	Requires fewer system instructions than scheduling an RPL exit routine. The application can issue one WAIT for a combination of VTAM and non-VTAM requests. ECBs are not automatically scheduled when the requested operation is complete.	Might require more storage because many pending requests can be outstanding, tying up RPLs and data areas.[1]	Most complex.
Asynchronous (OPTCD=ASY) • RPL exit routine scheduling (EXIT= *address*)	Requires more system instructions than posting an ECB. The RPL is automatically scheduled when the requested operation is complete. Some advantages if used to give priority of handling to a session (for example, first input after logon).	About the same as ECB posting.[1]	Less complex than ECB posting.

Note:

1. While an asynchronous request is outstanding, the RPL and associated data areas (for example, NIB, RPLAREA, EXTERNAL ECB, etc.) cannot be freed, altered, or reused.

Some questions that affect program organization

Table 4 lists some of the questions that must be answered when deciding how to design and code a VTAM application program.

Table 4. Some questions that affect program design and coding

Program function	Questions to answer
Opening the program	• Is there one or more than one ACB? • In which addressing mode should the application program run? (Ensure that the application program is in the same mode as it is when you open the ACB.) • Is the application capable of persistence? (See "Opening the ACB during recovery from an application failure" on page 68.) • Is the application multinode persistent capable? (See "Response to an application failure with MNPS" on page 70.)

Table 4. Some questions that affect program design and coding (continued)

Program function	Questions to answer
Establishing sessions with LUs	• Who initiates sessions? (The program logic might not have to be aware of this.) – Automatic logon (network operator procedure) that automatically initiates sessions on behalf of certain LUs? – Dependent LU (that is, an Initiate Self sent from the LU)? – Independent LU (that is, a BIND sent from the LU)? – Terminal operator associated with an LU (the LU forwards request from the terminal operator after perhaps modifying it in some way)? – The application program itself, by issuing OPNDST or SIMLOGON? – Another application program, by issuing CLSDST OPTCD=PASS? – An application program, by issuing REQSESS? – More than one of the preceding? • Does something other than the application program initiate the session? – Analyze the user logon message and session parameter before session establishment. (Use LOGON exit routine, INQUIRE to obtain the user logon message and session parameters, and OPNDST OPTCD=ACCEPT to establish the session.) – If no analysis is required, use OPNDST OPTCD=ACCEPT in the main program logic. • Does the application program always initiate the session? – Identity of LUs known to program? (Use OPNDST or SIMLOGON to establish sessions with the LUs). - Are you using network-qualified names? If so, put the name in NIBSYM and the network identifier in NIBNET. - Acquire as many as are available? (Use OPNDST OPTCD=(ACQUIRE,CONALL). - Acquire any (single) one of them? (Use OPNDST OPTCD=(ACQUIRE,CONANY).) - Simulate a logon so that all LUs appear to be logging on to the application program and thus can be handled by the same logic? (Issue SIMLOGON OPTCD=(CONALL or CONANY) and then issue OPNDST OPTCD=ACCEPT in the LOGON exit routine or in the main program.) • Identity of LUs not known to the program? (Use INQUIRE OPTCD=TERMS to obtain the identities of the defined LUs, then issue OPNDST or SIMLOGON.) • Are sessions pending recovery? – Application program can use INQUIRE OPTCD=PERSESS to determine whether there are sessions pending recovery. – Use OPNDST OPTCD=RESTORE to restore the session pending recovery. – Use CLSDST or TERMSESS to terminate the session pending recovery.

Table 4. Some questions that affect program design and coding (continued)

Program function	Questions to answer
Establishing sessions with LUs (continued)	• Is a session parameter used? – Is a session parameter specified by the Initiate (logon mode name)? - Is an INQUIRE macroinstruction needed to investigate the session parameter? - Does the application program ever have to modify the parameters? • Is the session parameter always supplied solely by the primary application program? – Is a logon mode name to be provided in the LOGMODE field of the NIB? – Is a BIND area address to be provided in the NIB? – Are negotiable BINDs used? • Does the program share LUs with another program? (Use the RELREQ exit routine to handle requests from other programs for your LUs. RELREQ does not apply if the PLU is an independent LU.) • Is it advantageous for identical, multiple applications to be defined to a single generic name for workload balancing (using SETLOGON OPTCD=GNAMEADD)? If yes and LU 6.1 or 6.2 protocols are used or your program issues SETLOGON GNAMEADD specifying AFFIN=APPL or your program issues OPNDST and/or OPNSEC and specifies LUAFFIN=APPL, use of the CHANGE macroinstruction might be in order. • Is it necessary to pace the processing of session-establishment requests (with SETLOGON OPTCD=(START or HOLD)) to prevent storage shortages?
Closing the program	• Is there more than one ACB? • Does the application program send final request on sessions before closing the program, or does it simply terminate the sessions? • How is the application program to terminate normally? – By VTAM operator closing down the network? (Use a TPEND exit routine.) – By special request from one or more LUs? – By some internal logic, such as time-of-day? – By system operator message (for example, through WTOR)? • Are sessions terminated normally prior to CLOSE ACB?

Table 4. Some questions that affect program design and coding (continued)

Program function	Questions to answer
Communicating with LUs • Receiving	• Identity of session known (CID of session in the RPLARG field)? (Use RECEIVE OPTCD=SPEC.) • Identity of session unknown? (Use RECEIVE OPTCD=ANY to read input, then RECEIVE OPTCD=SPEC.) • Exclude session from having its input complete an any-mode RECEIVE while it is in the midst of a transaction? (Specify OPTCD=CS.) • Expect to receive responses to requests (using SENDs with other than POST=RESP)? Can use either: – An RESP exit routine (for other than DFSYN responses) – RECEIVE RTYPE=RESP (for other than DFSYN responses) – RECEIVE RTYPE=DFSYN (for DFSYN responses only). • Expect to receive expedited-flow requests? Can use either: – RECEIVE RTYPE=DFASY – A DFASY exit routine. • Want to be able to receive session-control requests? (Use a SCIP exit routine.) • Can length of input request vary greatly? (Use PROC=KEEP in establishing the session and reissue RECEIVEs until the entire request is read.) • Expect that the LU might quiesce your application program (temporarily stop it from sending)? Have logic to receive QEC and RELQ requests as a result of DFASY input? • In which addressing mode should RPL exit routines run? (The exit routine enters in the mode of the caller at the time the RPL request issues.)

Table 4. Some questions that affect program design and coding (continued)

Program function	Questions to answer
Communicating with LUs • Sending	• Send all of a request at once? – Wait for the request to arrive at its destination before proceeding? (Use SEND OPTCD=SYN, POST=RESP.) – Start the request on its way and have VTAM post an ECB or schedule an RPL exit routine when VTAM receives a response to the request? (Use SEND OPTCD=ASY, POST=RESP.) – Have VTAM schedule the sending of the request and determine its arrival yourself? (Use SEND OPTCD=ASY (or SYN), POST=SCHED,RESPOND=*values*.) - Have the LU return a definite response that causes completion of RECEIVE RTYPE=RESP specified or that causes entry to an RESP exit routine? (Specify RESPOND=(NEX,FME,NRRN), (NEX,FME,RRN), or (NEX,NFME,RRN), according to SNA protocols.) - Have the LU return only an exception (negative) response and either assume successful arrival or determine from the next received request that the input arrived successfully? (Specify RESPOND=(EX,FME,RRN), (EX,FME,NRRN), or (EX,NFME,RRN), according to SNA protocols.) - Have the LU return no response and determine successful arrival yourself? (Specify RESPOND=(NEX,NFME,NRRN).) • Send an element in a chain of elements? (Specify SEND POST=SCHED CHAIN=(FIRST,MIDDLE, LAST) with RESPOND=EX on all but the last SEND.) – Request a definite response on the last SEND (to determine that the entire chain arrived successfully)? (Specify RESPOND=(NEX,FME,RRN), (NEX,FME,NRRN), or (NEX,NFME,RRN), according to SNA protocols, and receive the response with RECEIVE RTYPE=RESP or with a RESP exit routine, or specify POST=RESP on the last element and the operation is not posted complete until the response comes back.) – Request only a negative response and assume by subsequent action of the receiver that the chain was received successfully? (Specify RESPOND=(EX,FME,RRN), (EX,FME,NRRN), or (EX,NFME,RRN), according to SNA protocols.) • Send a long message with a single SEND? (Employ the large-message performance-enhancement outbound option, SEND OPTCD=LMPEO.) • Send a message from a set of non-contiguous buffers? (Use the buffer list option, SEND OPTCD=BUFFLST.) • Specify the RH indicators for each request? (Use SEND OPTCD=USERRH to specify your own RH.)
Handling errors and special conditions	• What kind of information should be saved in a LERAD exit routine? Does the logic error affect only one session or the entire program? • Which physical errors can be retried in a SYNAD exit routine? Which require that the session be terminated? Which require sending a system operator message? Which require that the program terminate? • What action should be taken in the SCIP, NSEXIT, and LOSTERM exit routines? • What action should be taken if VTAM abnormally terminates while running under a user's task? – Should the application program have a STAE, or ESTAE exit routine to investigate and clean up its own files?

Table 4. Some questions that affect program design and coding (continued)

Program function	Questions to answer
Interfacing with the z/OS operating system and VTAM	• What kind of interface is offered by z/OS and VTAM?
Handling control blocks and work areas • For session establishment	• Acquiring a session with one in a list of known LUs? Decide whether to: – Assemble an NIB or a list of NIBs and an RPL (for OPNDST) into the application program. – Generate and initialize the NIB or list of NIBs and the RPL dynamically using the GENCB macroinstruction or DSECTs. – Obtain the NIB or list of NIBs and the RPL from a pool, assembled or created dynamically, and initialize them. If you are using a pool, you cannot use a negotiable BIND. • Acquiring a session with one or a list of unknown LUs? (Use INQUIRE OPTCD=TERMS to create and initialize NIBs and use an assembled, generated, or pool-obtained RPL.) • Are parallel sessions established? • Accepting a session? Decide whether to: – Have a LOGON exit routine and reuse the same NIB and RPL for each session-establishment request – Have one or more OPNDSTs in the mainline program, each of which requires an RPL and an NIB. • Do you need to create a buffer (RPLAAREA) to contain a negotiable BIND response? • Do you need to create a BIND area for a session parameter? • An SCIP exit is recommended for receipt of UNBIND. (A negative response to BIND is not returned by a higher-level node.)
Handling control blocks and work areas • For communication	• Simple program with synchronous requests? (Assemble RPLs and data areas in the program and reuse them for each request.) • Asynchronous program? • Assemble, generate, or obtain from an assembled or generated pool one RPL, one ECB (if an ECB is posted), and one work area (data area and flags) for each concurrent session? Decide whether to use this storage for the duration of the following: – The session with the LU – A receive and a related SEND – A series of RECEIVEs and SENDs. • Put the address of session-related storage in the USERFLD of the NIB if the storage was obtained for the session?

Application programming interface

Be aware of the interface an application program has with the operating system and with VTAM. For example, VTAM depends on the MVS Data Facility Storage Management System (DFSMS) for initial processing of OPEN and CLOSE macroinstructions.

General requirements

You can write application programs that execute in either 24- or 31-bit addressing mode. However, to maintain an interface with existing programs, your 31-bit

addressing mode application program may need subroutines or portions of code that execute in 24-bit addressing mode. If your 31-bit addressing mode application program resides in 24-bit storage, it can change to 24-bit addressing mode when necessary.

When developing an application program that executes in 31-bit addressing mode, consider the following:

- Mode of the caller
- Desired mode of a routine being called
- Location of control blocks passed to other routines
- Location of routines whose addresses are passed as parameters
- Length of the address parameter fields

VTAM receives control from the macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

Specific requirements

The application programming interface (API) of VTAM supports application programs in 31-bit as well as 24-bit addressing mode. Control blocks that use the API can reside in either 24-bit or 31-bit storage but must be consistent with the addressing mode of the application program. For example, an application program running in 24-bit addressing mode should not reference control blocks allocated (that is, resident) in 31-bit storage.

Further considerations for an application program interfacing with VTAM include:

- All EXLST exit routines except LERAD and SYNAD are given control in the addressing mode of the application program at the time the ACB was opened.
- LERAD and SYNAD exit routines are given control in the mode of the application program at the time the CHECK macroinstruction was issued. The application program is responsible for issuing the CHECK macroinstruction in the same addressing mode as the addressing mode of the application program at the time the original request was made.
- RPL-specified exit routines are entered in the mode of the application program at the time the RPL request was issued.

For more information about 31-bit addressing, see "31-bit addressing" on page 319. For detailed descriptions of each exit routine and macroinstruction, see Chapter 7, "Using exit routines," on page 219, and Chapter 13, "Conventions and descriptions of VTAM macroinstructions," on page 371, respectively.

Handling control blocks and work areas

About this task

The application program handles control blocks and session-related work areas (data areas and status flags) in a number of ways. It can:

- Define RPLs, NIBs, or EXLSTs in the application program during assembly or generate them during program execution by using the GENCB macroinstruction or DSECTs
- Define RPLs, NIBs, or EXLSTs that reside in 31-bit storage

- Assign one RPL or NIB to a specific session during assembly, or assemble or generate RPLs and NIBs that are to be available for any session as the need arises
- Retain the RPL used in establishing the session for all further communication on the session
- Use one RPL for all session-establishment requests and use another RPL or group of RPLs for all communication requests
- Define the RPLs, NIBs, and any other control blocks associated with sessions as a pool so that a limited amount of control block storage is not exceeded.

In application programs that must handle many sessions concurrently, it might be useful to have a control block other than the RPL or NIB associated with a particular session. VTAM provides a way of associating a storage area with a particular session. See "USERFLD field of the NIB" on page 37 for more information on associating specific information with a session.

Techniques for handling control blocks and work areas

The following techniques are useful for handling control blocks and work areas.

Element per LU at assembly

This method works well if a known set of LUs is placed in session with the program, and each LU has only a single session with the application program. Each LU session uses a separate, predefined RPL. The RPL points to an NIB and to a data area. The USERFLD field of the NIB can be set to point to a status save area for the LU.

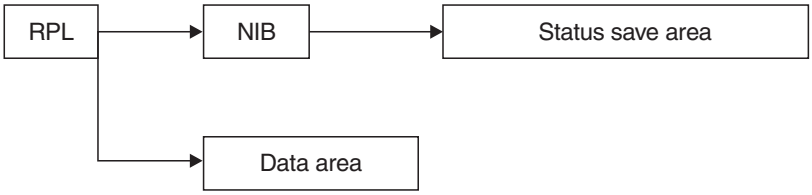


Figure 12. Element per LU at assembly

To do this, a correspondence can be set up between each session and its RPL. For example, a session table can be constructed that matches a session with its RPL. Because fixed session establishment is being used, a separate OPNDST macroinstruction can be coded for each session; each OPNDST can be coded to use a specific RPL. When a data-transfer request completes, the RPL's USER field points to the status save area. The RPL continues to point to the data area.

Element per session at session establishment

This method can be used if the application program accepts logons from LUs whose names are not known at assembly. A pool of RPLs, NIBs, data areas, and status save areas can be set up as needed. The pool can consist of elements; each element contains one RPL, one NIB, one data area, and one status save area. As each CINIT is received, the application program selects an available element from the pool and uses some technique to associate the element with the session.

One technique is to put the address of the element in the USERFLD field of the NIB prior to issuing the OPNDST macroinstruction. Subsequently, whenever execution of an RPL-based macroinstruction is completed, the address is available in the USER field of the RPL.

After selecting a pool element for a session, that element can be used with the same session for all subsequent requests. When the session is terminated, the element is returned to the pool.

Element per transaction

Here again, a pool is used for storage management. However, instead of assigning an element to a session for the life of the session, a new element is obtained for each transaction. A transaction is a two-way interchange: data goes both from and to the LU. An element is obtained for establishing a session and returned when the session is established. A new element is obtained for each transaction and returned when the transaction is completed. Using this technique, an NIB does not have to be included in the elements used for the transactions.

Element per request

This is a more dynamic version of the element-per-transaction method. Here, you use a new element for each request. As you complete each request, the element returns to the pool. As a modification to this method, you can use one NIB and one RPL for all session-establishment and termination requests, and obtain additional pool elements for subsequent data-transfer requests. You can assemble the NIB and RPL for session establishment and termination in the application program; they do not need to come from a pool.

To maintain a strict “element per request” technique, the data area portion of an element can be used to hold an NIB for session-establishment and termination

requests, and as a data area for data transfer. To do this, one NIB can be set up in the application program. For each request, the contents of the NIB can be moved into the data area in the pool element, or a GENCB can be used to build a complete NIB in the data area. This reduces the size of the pool elements.

Combinations

These techniques can be combined to suit particular needs. Here are two ways to combine storage management techniques:

- At assembly, establish a fixed status save area for each session to be established. Each status save area can contain a user identification to be compared to one contained in a CINIT, or the save area can be used to count the number of times a particular session has been established with an LU during the day. At session establishment, a pool element is obtained (containing RPL, NIB, and data area) and retained for the duration of the session. The fixed save area provides a permanent place to keep session information.
- Assign one status save area per session at session establishment. This is more dynamic than method 1 because the programmer does not have to know at assembly which session is created. One RPL and data area per request or per transaction can be used. Again, the status save area can be used to keep track of session status, but only for the life of the session. The RPL and data area, selected from a pool, allow dynamic data-transfer requests.

Note: It is good programming practice to set control block fields in user-written application programs to 0 after they are used if the control block is to be reused. Desired operands should be specified or reset to the system default. ACB, NIB, and RPL fields that are not used by a macroinstruction should be set to 0 unless otherwise indicated.

Chapter 4. Opening and closing an application program

This chapter describes aspects of the OPEN, CLOSE, and ACB macroinstructions that apply to all application programs. If the application program uses special functions that require authorization or that are operating system dependent, additional considerations apply. See Chapter 10, "Operating system facilities," on page 295, and Chapter 12, "Coding for the communication network management interface," on page 337, for additional information.

For multiple, identical application programs that provide the same function and that are known and accessed by a single generic name, see "Opening and closing an application program as a generic resource" on page 78.

Opening an application program

After a VTAM application program starts, it issues an OPEN macroinstruction to notify VTAM that it is an active element in the network.

VTAM uses the RACROUTE macroinstruction to verify user authority for opening an application. RACROUTE invokes the System Authorization Facility (SAF) MVS router, which conditionally directs control to the security management product. For additional information on RACROUTE, refer to the z/OS Security Server RACF Security Administrator's Guide.

VTAM grants the application access to the network when:

- SAF indicates that the application is authorized to become an element in the network. VTAM then bypasses the password checking, and the security management product provides resource access control.
- SAF cannot make a decision, and VTAM successfully performs VTAM password verification.

During the OPEN processing, VTAM establishes an SSCP-LU session with the application program logical unit (LU). VTAM can now accept requests for LU-LU sessions with this application program. In addition, the application program can now make further requests of VTAM. Normally, the application program remains open until it issues a CLOSE macroinstruction.

VTAM considers each open access method control block (ACB) to be a separate LU. Therefore, if an application program opens more than one ACB, VTAM sees each open ACB as a different LU, even though the ACBs are related to the same application program.

Opening an application program requires the following:

- An ACB defined with an ACB or GENCB macroinstruction
- An OPEN macroinstruction

Because an ACB can point to a list of exit routines, defined with an EXLST macroinstruction, an EXLST macroinstruction is usually also required.

© Copyright IBM Corp. 2000, 2013

Access method control block (ACB)

The ACB contains the following information that describes the application program to VTAM:

- The name of the access method used in operating the ACB (VTAM).
- The address of an application program identifier. The application program identifier must match the name that is specified on the APPL definition statement provided as part of the VTAM definition. When the application program opens an ACB, VTAM verifies that the ACB name is defined to VTAM. If the name is not defined, VTAM does not process the ACB.

  If an application program name is not specified on the APPLID operand of the ACB macroinstruction, VTAM uses a name supplied by the operating system. If the name supplied by the operating system does not match a name on the APPL definition statement, VTAM does not process the ACB. The following indicate what name is used for each operating system:

  If the application program is started by a job step that is a procedure invocation, the name is the procedure step name in the job control language (JCL) for the application program. Otherwise, it is the job step name in the JCL.
- Optionally, the address of a password associated with the application program. When an ACB is opened, VTAM compares the ACB password with a password that is defined on the APPL definition statement. If the passwords do not match, VTAM does not process the OPEN macroinstruction. If a password is not specified on the APPL statement, you do not need to specify a password on the ACB macroinstruction.
- The address of an exit list (EXLST control block) containing the names of exit routines written in the VTAM application program to handle specific events. These are described in detail in Chapter 7, "Using exit routines," on page 219.
- An indication of whether the application program accepts sessions that it has not initiated (MACRF=LOGON). If MACRF=NLOGON is specified, an application must use OPNDST OPTCD=ACQUIRE to establish sessions.

  If the application program is to establish sessions in which it acts as the SLU, you must code MACRF=LOGON.
- An NIB, if the application program is an authorized CNM application program, to establish access to an SSCP-LU session between the SSCP and the application program.
- Indicates whether the application program supports LU 6.2 architecture protocol extensions for full-duplex and expedited data transmission.
- An indication of whether the application program is capable of persistent LU-LU session support.
- An indication of whether the OPEN associated with this ACB is to be considered a multinode persistent sessions forced takeover request.
- An indication of whether the application program is authorized to use the performance monitor interface.
- A 4-byte field set by the application program to provide the address of the Application-ACB vector list.
- A 4-byte field that can be set by the application program and used for any application program purpose. VTAM ignores this field. For example, the field can be set with the address of a control block to be used by the application program when an exit routine is invoked.
- A 4-byte field containing the address of the access-method support vector list. During the OPEN processing, VTAM fills in this field. The vector list contains

the release-level, component-identification, and function-list vectors and is described in “The access-method-support vector list” on page 62.

- A 4-byte field containing the address of the resource-information vector list. During the OPEN processing, VTAM fills in this field. The vector list contains information about the application program that opened the ACB and is described in “Resource-information vector list” on page 63.

The ACB and its related storage (APPLID, password, EXLST, NIB, and Application-ACB vector list) must be allocated in the same storage key. This key can be the storage key of the program status word (PSW) at the time the OPEN macroinstruction was issued, or the storage key of the task control block (TCB).

For each ACB opened to VTAM, there is approximately X'250' bytes of storage used as a VTAM work area. This work area is released when CLOSE ACB processing completes.

The following sample ACB macroinstruction is used to build an access method control block:

```
ACB1     ACB   AM=VTAM,APPLID=APID1,                                C
               PASSWD=PSWD1,EXLST=EXIT,MACRF=LOGON,                 C
               PARMS=(USERFLD=A(MYLU))
         .
         .
         .
APID1    DC    AL1(L'AP1NAME)
AP1NAME  DC    C'MYPROG'
PSWD1    DC    AL1(L'PASSCHAR)
PASSCHAR DC    C'JOE007'
EXIT     EXLST AM=VTAM,NSEXIT=NSRTN,SCIP=SESSCTL,                   C
               LOGON=CINIT,                                         C
               LOSTERM=LTRTN
MYLU     DS    XL100
```

- ACB1, the symbolic name for this ACB, is included in the OPEN macroinstruction that is used to open this ACB.
- AM=VTAM identifies VTAM as the access method for this ACB.
- APID1 is the address of the application program identification (MYPROG). A request to initiate a session with this application program can specify MYPROG.
- MYLU is a control block to be associated with the ACB. It might contain the status flags and other variables associated with the application program LU.
- PSWD1 is the address of the password (JOE007).
- EXIT is the name of the exit list created by the EXLST macroinstruction.
- MACRF=LOGON specifies that:
  - Other LUs can initiate a session with this application program
  - VTAM schedules the SCIP exit routine when a BIND request is received by the application program.

An ACB can also be created when VTAM issues a GENCB macroinstruction to execute an application program.

OPEN macroinstruction

Having created an ACB, the application program opens it by issuing an OPEN macroinstruction. For example:

```
OPENPROG      OPEN       ACB1
```

This macroinstruction opens an ACB with the name ACB1.

Note: The OPEN and CLOSE macroinstructions use a positional rather than a keyword operand.

Vector lists

Vector lists provide an open-ended interface for application programs to exchange information with VTAM. Each vector list contains a 2-byte length field followed by a set of contiguous vectors in random order, as shown in Figure 13.

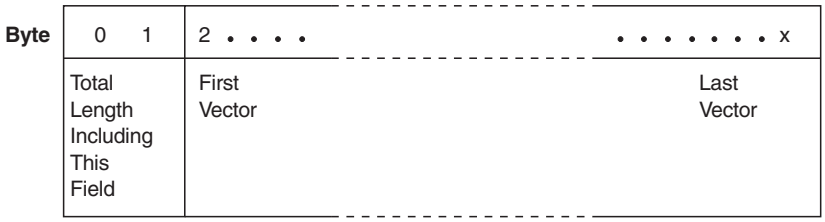


Figure 13. Format of a vector list

Each vector in the vector list is a set of contiguous data containing a length field, an identifier, and a value field containing vector data. The fields containing vector data can have trailing blanks. Some vector lists are built before or during OPEN processing, while others are built during processing of the APPCCMD macroinstruction. Table 5 shows the vector lists available to VTAM applications, the macroinstruction with which they are associated, and their function.

Table 5. Vector lists

Name	Pointer	Macroinstruction	DSECT	Purpose	Page
Access- Method-Support	ACBAMSVL field in the ACB	OPEN ACB	ISTAMSVL	Supplies information to the application about the VTAM that opened the ACB	For more information about this vector list, see "The access-method-support vector list" on page 62.
Resource-Information	ACBRIVL field in the ACB	OPEN ACB	ISTRIVL	Supplies information to the application about the resources available to the application.	For more information about this vector list, see "Resource-information vector list" on page 63.
Application- ACB	ACBAVPTR field in the ACB	OPEN ACB	ISTVACBV	Supplies information to VTAM about the application's capabilities.	For more information about this vector list, see "Vector lists supplying information to VTAM" on page 59.

Table 5. Vector lists (continued)

Name	Pointer	Macroinstruction	DSECT	Purpose	Page
VTAM- APPCCMD	VTRINA field in the RPL extension	APPCCMD	ISTAPCVL	Supplies information to the application about a particular conversation with a partner LU.	For more information about this vector list, refer to z/OS Communications Server: SNA Programmer's LU 6.2 Guide.
Application-APPCCMD	VTROUTA field in the RPL extension	APPCCMD	ISTAPCVL	Supplies information to VTAM about a particular request on a conversation with a partner LU	

There are two types of vector lists associated with OPEN processing:

- Those used by the application program to supply information to VTAM
- Those used by VTAM to supply information to the application program.

Vector lists supplying information to VTAM

The application program can pass information to VTAM for OPEN processing through the application-ACB vector list (pointed to by the ACBAVPTR field in the ACB). The IBM-supplied DSECT ISTVACBV enables you to refer to the fields in the application-ACB vector list symbolically.

OPEN processing vector lists built by the application have a 2-byte length field. The format of the application-capabilities vector is shown in Figure 14 on page 60.

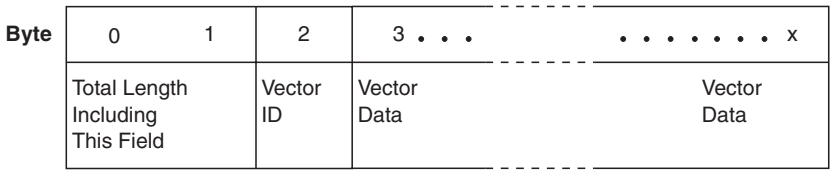


Figure 14. Format of each vector within the application-ACB vector list

The application-ACB vector list consists of the following vectors:

- **Application-capabilities vector:**

  This vector is used by VTAM applications to provide the following information about the application's capabilities. The vector ID is X'81'.

  Field Capability

  VAC81MLE (X'80')
  LU 6.2 application supports having its logon exit driven multiple times per session request. Applications with LOGON exits must set this indicator to benefit from verification reduction. For more information about this function, refer to the z/OS Communications Server: SNA Network Implementation Guide.

  VAC81FPR (X'40')
  LU 6.2 application can receive data directly into CSM buffers by specifying OPTCD=XBUFLST on the APPCCMD macroinstruction.

  VAC81PWS (X'20')
  LU 6.2 application can use password substitution.

  VAC81ESS (X'10')
  LU 6.2 application can handle extended security sense codes.

  VAC81FPS (X'08')
  LU 6.2 application can send data directly from CSM storage by specifying OPTCD=XBUFLST on the APPCCMD macroinstruction.

  VAC81EOM (X'04')
  Programmed operator application can handle the command-complete message (IST1746I) after completion of the DISPLAY, VARY, and MODIFY commands. IST1746I is issued to a program operator application and does not go to the VTAM operator console. This message is not issued when command completion is not apparent to VTAM (for example, when a VARY command is successful).

  VAC81ACO (X'02')
  Application indicates that it requests AUTOSES for CNOS only (that is, it will not attempt to automatically restart AUTOSES sessions following session outage).

  VAC81FAA (X'01')
  Application indicates that it requests ATNLOSS=ALL (that is, it overrides specification of the ATTNLOSS parameter on the APPL definition statement).

  VAC81UCV (X'0080')
  Application indicates that it supplies user control vectors with the SETLOGON OPTCD=START macroinstruction. See “Supplying control vectors with the SETLOGON START” on page 64 for more information.

- **Local-application's-DCE-capability vector:** This vector is used by LU 6.2 applications to inform VTAM about the application's DCE security capabilities. During session establishment, VTAM passes this information to the partner LU in byte 22 of the BIND.

  The vector ID is X'82'.

To supply the application-ACB vector list, an application specifies PARMS=(APPLVCTR=*address*) on the OPEN ACB macroinstruction, where *address* is the location of the application-built vector list. To build the ACB vector list, the application program:

1. Obtains storage for the vector list
2. Initializes the fields using the ISTVACBV DSECT and the DSECTs contained within the ISTVACBV DSECT.

For the complete layout of the ISTVACBV DSECT, see "Application-ACB vector list (ISTVACBV)" on page 789.

Vector lists supplying information to the application

When the application program's OPEN macroinstruction successfully completes, each of two address fields in the ACB is set with a pointer to a variable-length storage area. These areas are located in storage that is read-only for the application program. The storage is addressable from the MVS address space in which OPEN is issued.

Each address field contains the address of a vector list. The two vector lists are:

- The access-method-support vector list (pointed to by the ACBAMSVL field in the ACB DSECT). This list describes the VTAM that processed the OPEN macroinstruction.
- The resource-information vector list (pointed to by the ACBRIVL field in the ACB DSECT). This list specifies resource identifiers which might be unknown to the application program.

The format for vectors contained in the access-method-support vector list and the resource-information vector list is shown in Figure 15 on page 62.

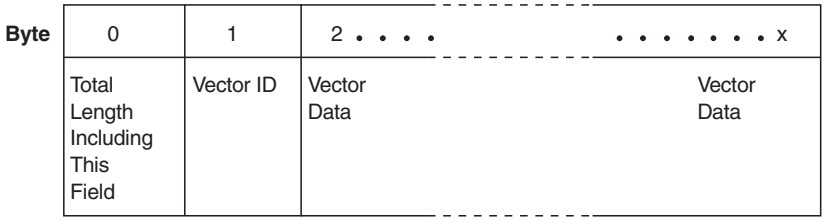


Figure 15. Format of vectors built by VTAM during OPEN processing

You can reference the vector lists any time after the OPEN macroinstruction completes, and until the CLOSE macroinstruction or equivalent terminations, such as an ABEND, occurs. Information similar to this in the access-method-support vector list is also available at assembly time. See "ISTGLBAL macroinstruction" on page 274 for a description of the information available at assembly time.

Applications that use the APPCCMD macroinstruction can also request information from VTAM in the APPCCMD vector list. Refer to the z/OS Communications Server: SNA Programmer's LU 6.2 Guide for more information.

The access-method-support vector list: The access-method-support vector list is pointed to by the ACB's ACBAMSVL field in the ACB DSECT. This list describes the global variables for the VTAM program that processed the OPEN macroinstruction.

For the complete layout of the ISTAMSVL DSECT, see "Access-method-support vector list (ISTAMSVL)" on page 781.

The access-method-support vector list consists of the following vectors:

Release-level vector
This vector contains the access method version and release number. The vector ID is X'01'.

Component-identification vector
This vector contains product identification information about a major component or feature of the VTAM licensed program. This information is used by IBM for VTAM program maintenance. When a vector list contains multiple component-identification vectors, the first vector designates the base VTAM product; subsequent vectors designate features or other major components of VTAM. The vector ID is X'04'.

Function-list vector
This vector contains a variable-length bit string in which each bit corresponds to a particular VTAM function. If a bit is on, the corresponding function is present in the particular release of VTAM. If a bit is off, the function is not available. If the vector is not present, or if it is shorter than expected, you can assume the missing bits to be 0. The vector ID is X'05'.

The information contained in the function-list vector is also available at assembly time in global variables created with the ISTGLBAL macroinstruction. "ISTGLBAL macroinstruction" on page 274 describes the use of global variables.

LU 6.2-support-function-list vector
This vector is provided to indicate which LU 6.2 options are supported by

this release of VTAM. This vector is not present for application's that do not use VTAM's APPC API. The vector ID is X'06'.

Each subvector in this vector list can have one of the following values that correspond one-to-one with LU 6.2 global variables:

X'00' Option is not supported

X'01' Option is supported

X'02' Pass-through (VTAM offers support for the function, but the application program must implement the function.)

The vector list information is not available until execution time. Refer to the z/OS Communications Server: SNA Resource Definition Reference.

Resource-information vector list: The resource-information vector list provides information about the application program that opened an ACB at execution time. The ACB's ACBRIVL field points to the list. The application program must search the vector list to find a particular vector.

For the complete layout of the ISTRIVL DSECT, see "Resource-information vector list (ISTRIVL)" on page 785.

The resource-information vector list provides the following vectors:

Application-network name vector
This vector contains the network name of the application program LU. This name is specified by the name field of the APPL definition statement. The vector ID is X'02'.

Application-ACB-name vector
This vector contains the ACBNAME of the application program. This is the name specified by the APPLID operand of the ACB statement; if the ACBNAME operand is not present, the network name of the application program LU is used. The vector ID is X'03'.

Network-name vector
This vector contains the name of the network in which the host resides. This name is specified by the NETID start option; if the NETID start option is not specified, this vector contains 8 bytes of blanks. The vector ID is X'06'.

SSCP-name vector
This vector contains the name of the SSCP. This name is specified by the SSCPNAME start option. The name contained in this vector is used to identify the SSCP to the NetView® program. The vector ID is X'07'.

Host-subarea-PU-network-name vector
This vector contains the network name of the host subarea physical unit (PU) contained in this host. This name is specified by the HOSTPU start option; if the HOSTPU start option is not specified, this vector contains the default host subarea PU name, ISTPUS. This name is part of the session awareness data provided to the NetView program or session monitor when a resource is contained within the subarea of the host PU. The vector ID is X'08'.

Host-subarea-PU-network-address vector
This vector contains the network address of the host subarea PU. The vector ID is X'09'.

Maximum-subarea vector
This vector contains the maximum subarea number (in binary) that is valid for this host's domain. This is obtained from the MAXSUBA start option. The vector ID is X'0A'.

LU 6.2-application-definition vector
This vector is present in this list for LU 6.2 application programs. The LU 6.2 application program may use this vector to determine the values coded on the APPL definition statement.

For more information about the APPL definition statement, refer to the z/OS Communications Server: SNA Resource Definition Reference. The vector ID is X'0B'.

Common-application-definition vector
This vector is present in this list for all application programs. The application program may use this vector to determine the values coded on the APPL definition statement.

For more information about the APPL definition statement, refer to the z/OS Communications Server: SNA Resource Definition Reference. The vector ID is X'0C'.

The APPCCMD-vector-area-length vector
This vector is present in this list for LU 6.2 application programs. It contains the absolute minimum length and the recommended minimum length for full use of the APPCCMD vector area. The vector ID is X'11'.

Application-to-VTAM-vector-keys vector
This vector contains a list of all ACB vector keys presented by the application program on the Application-ACB vector. The vector ID is X'12'.

Performance monitor vector
This vector, if present, identifies any fields in the performance data parameter list of the installation-wide performance monitor exit routine (ISTEXCPM) that are no longer supported. The vector points to a table of field entries, each of which contains information needed to locate the position of the unsupported field within the affected vector. Refer to z/OS Communications Server: SNA Customization for more information. The vector ID is X'13'.

Supplying control vectors with the SETLOGON START

If VTAM indicates (by the ISTAMSVL DSECT) that it supports applications supplying user vectors on the SETLOGON START and if the application turns on VAC81UCV when it issues its OPEN ACB, the application can provide a TCP/IP Information Control Vector (CV64) when it issues the `SETLOGON OPTCD=START` macroinstruction. The CV64 contains IP address and port information, as well as other IP characteristics. After this information is associated with the application, it can be propagated to the PLU of a session with this application and some of it can be displayed by a VTAM operator, depending on the settings of the IPINFO start option.

To pass the CV64, the application needs to turn on RPLNIB and set RPLARG in the SETLOGON RPL to point to a NIB and the application needs to set NIBUCVA in that NIB to point to a vector list. The format of vector list is shown in Figure 16 on page 65.

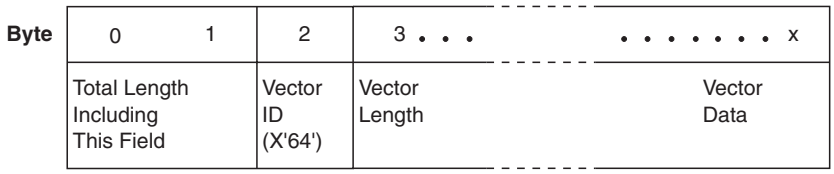


Figure 16. Format of the CV64 Vector List to supply with SETLOGON START

The lengths and the subvectors of the CV64 will be checked. If all is fine, the CV64 will be stored and associated with the SETLOGON issuer for later use in a session setup or a display.

If the application issues another SETLOGON START with CV64 information, the old CV64 information stored by VTAM will be deleted and it will be replaced with the new CV64.

If the application attempts to pass the CV64 on the SETLOGON START without having indicated that it will exploit the new function, the CV64 information will be ignored. Likewise, if the application attempts to pass the CV64 on the SETLOGON START to a VTAM that does not support the function, the CV64 information will be ignored.

Using multiple ACBs in a VTAM application program

Normally, a VTAM application program has only one ACB; the program is known to VTAM by only one APPL identification. However, a program might be known under two or more different APPL identifications, and each requires that a separate ACB be opened. One OPEN macroinstruction can be used. For example:

```
OPENPROG OPEN  (ACB1,,ACB2)
```

Using network-qualified names

Network-qualified names provide an alternative to alias name translation if there are duplicate resource names within interconnected networks. By using network-qualified names for resources that are session-capable, such as LUs, control points (CPs), and application programs, resource names can be duplicated among networks in a multiple-network environment.

With network-qualified names, only names for session-capable resources have to be unique within a single network. Names can be duplicated in SNA interconnected networks because they are network-qualified. A network-qualified name is in the form of *netid.resource_name*. VTAM automatically appends the network identifier to each resource name so that an LU name LU1 in NETA would be known as NETA.LU1 and an LU named LU1 in NETB would be known as NETB.LU1.

A name that is not network-qualified has from 1 to 8 characters. A network-qualified name, *netid.resource_name*, has from 3 to 17 characters, where *netid* has from 1 to 8 characters and *name* has from 1 to 8 characters. The application program will pass the network-qualified name across the API as an 8-byte NETID and an 8-byte name.

Establishing network-qualified names capability

An application indicates that it can support network-qualified names by specifying PARMS=(NQNAMES=YES) on the ACB macroinstruction. See "ACB—Create an access method control block" on page 375 for more information.

Where OPEN can be issued

Normally, the communication part of the VTAM application program issues the OPEN macroinstruction. The OPEN macroinstruction cannot be issued from an exit routine.

When using persistent LU-LU session support, the recovering application program can issue the OPEN ACB from the same or a different address space. With MNPS, the ACB can be OPENed on another VM within the sysplex.

See Chapter 10, "Operating system facilities," on page 295, for additional information pertaining to ACBs.

Using persistent LU-LU session support

Persistent LU-LU session support can be used by a VTAM application program to facilitate session recovery following a failure.

Single-node persistent sessions (SNPS) is used to help recover sessions that are disrupted by an application failure. Single-node persistent session support can also be used to manage a planned takeover. When a VTAM application program fails after it has enabled persistence, VTAM retains the LU-LU sessions. In order for the sessions to be retained, VTAM and the operating system must remain active. See "Application program recovery with single-node persistence enabled" on page 67 for information about how VTAM and the application react to application failures.

Multinode persistent sessions (MNPS) is used to help recover sessions that are disrupted by a node failure, such as a failure in the hardware, operating system, or VTAM. Multinode persistant sessions can also be used to move an application to another VTAM, either after an application failure, or as part of a planned takeover, or as part of a forced takeover. Multinode persistent session support is available in a sysplex environment. Information used to re-establish the session is maintained in the coupling facility. For information about setting up an environment that supports mulitnode persistent sessions, refer to the z/OS Communications Server: SNA Network Implementation Guide. See "Application program recovery with multinode persistence enabled" on page 69 for information about how the sysplex and the application react to node failures.

There are no differences in the programming interface for applications using single-node or multinode persistent sessions, with one exception discussed in "Response to an application failure with MNPS" on page 70. For both functions, the application must enable persistence as described in "How an application establishes persistence" on page 67.

When an application closes its ACB or terminates (whichever occurs first), VTAM terminates its sessions unless the application is both capable of persistence and enabled for persistence.

Following the recovery of the application program, the sessions must be restored to permit continued use. See "Restoring sessions pending recovery" on page 137 for more information.

How an application establishes persistence

An application indicates that it is capable of persistence by specifying PARMS=(PERSIST=YES) on the ACB macroinstruction. The application enables persistence by specifying OPTCD=PERSIST on the SETLOGON macroinstruction. At any time the application program can disable persistence by issuing SETLOGON OPTCD=NPERSIST.

Note: An application program must be capable of persistence before it can enable persistence. Throughout this book, the term *enabled* (relative to persistence) implies that the application program is also capable of persistence.

Applications that are capable of, but not enabled for, persistence cannot recover sessions disrupted by application or node failures.

Session recovery states

Following a failure, a VTAM application program that has enabled persistent LU-LU session support is in one of three states:

Recovery pending
The application program or node has failed with persistence enabled, and the recovering application has not opened its ACB.

Recovery-in-progress
The recovering application has opened its ACB and there are sessions pending recovery. The application remains in this state until the last session is either restored or terminated.

Recovery complete
The application has either restored or terminated all sessions pending recovery.

Application program recovery with single-node persistence enabled

About this task

Single-node persistence is used to facilitate session recovery after an application failure. When a failure occurs, the application program has two ways to manage the recovery:

- Another instance of the failing application program can restart, recover the connection to VTAM, and restore sessions. VTAM reconnects the failed application when the recovering application issues an OPEN ACB using the original ACBNAME.
- An alternate application program that has been started already can recover the connection to VTAM and restore sessions. See "Takeover OPEN" on page 69 for more information.

Response to an application failure with single-node persistence

When a failure occurs that involves an application program that has enabled persistence, VTAM closes the ACB on behalf of the application program but retains the LU-LU sessions. The application program enters the recovery pending state

until a recovering application issues OPEN ACB. VTAM changes the primary logical unit (PLU) state and secondary logical unit (SLU) state to inhibited. The PLU and SLU states determine whether the application program can accept new sessions. (See Table 6 on page 82 for more information.) All resources that will be needed if the sessions are restored remain allocated to that application program.

If an application program has enabled persistence and an operator issues a VARY INACT, ID=*appl_name*, TYPE=FORCE command for that application, VTAM overrides persistence and terminates the application and its sessions normally. With SNPS persistence is also overridden by a HALT command.

If the application has set a persistent session timer (PSTIMER) on the SETLOGON macroinstruction, VTAM ensures that recovery occurs within the specified time limit. The timer is started by application failure and continues until either the time limit expires or the recovering application issues the OPEN ACB. If the timer expires, VTAM terminates the retained sessions as if persistence had not been enabled and releases the held resources. If PSTIMER is not set, the application remains in pending recovery state until the application reopens its ACB. See "SETLOGON—Modify an application program's capability to establish sessions" on page 543 for more information about application programs' session-establishment capability.

Opening the ACB during recovery from an application failure

During the recovery process, the application program issues the OPEN ACB, using the original ACBNAME, to reconnect to VTAM. Upon return of the OPEN ACB macroinstruction, VTAM will indicate that this application is a persistent instance (the opening application program has taken over or recovered an ACB that was previously used by an application program that was enabled for persistence).

During the OPEN processing, VTAM verifies that the following conditions are true.

- Original application had enabled persistence
- Recovering application is capable of persistence
- Recovering application's MACRF value matches the original application's MACRF value.
- Recovering application's support for network-qualified names matches that of the original application.

If any of the preceding requirements are not met, the OPEN ACB fails. The state of the application and sessions remains unaltered. If all requirements are met, VTAM begins the recovery process.

The recovering application does not need to re-establish persistence (SETLOGON OPTCD=PERSIST). When the application program opens an ACB for which persistence had been previously enabled, persistence remains enabled.

During the application recovery process, VTAM ensures that the allocated resources will be available during session recovery. VTAM then cancels the persistent session timer and completes the OPEN processing. After processing the OPEN ACB, VTAM changes the application's state from inhibited to the appropriate state as shown in Table 6 on page 82.

Takeover OPEN

The OPEN ACB macroinstruction also allows an alternate application to take over an active application that has enabled persistence. VTAM requires that the opening and original applications have the same ACBNAME.

When a takeover occurs, the takeover application program connects to VTAM and restores sessions, even though the original application has not failed. The takeover application program opens its ACB (PARMS=(PERSIST=YES)), and VTAM schedules the TPEND exit of the original application program and closes the application program. If the original application program does not have a TPEND exit, VTAM closes it without any notification. Any sessions are placed in the recovery pending state. The alternate application program must restore or terminate these sessions.

An application can indicate that it can be the target of an SNPS takeover attempt by specifying PARMS=(FORCETKO=ALL), or PARMS=(FORCETKO=SINGLE), on a SETLOGON OPTCD=PERSIST macroinstruction invocation. In addition, by default, an application is considered to support SNPS takeover attempts, but not MNPS forced takeover attempts, until the application indicates otherwise. If the original application has not indicated its support of an SNPS takeover request, and a takeover OPEN ACB is processed, then the OPEN ACB will be failed with an OPEN error code X'7C'.

Requirement: If the original application supports network-qualified names, the recovery application also must support network-qualified names.

Application program recovery with multinode persistence enabled

Multinode persistent sessions is used to facilitate session recovery after a node failure. It can also be used to move applications to another VTAM in the event of an application failure or as part of an MNPS takeover. The application can recover on the same VTAM (if that VTAM is restarted) or any other VTAM that is part of the sysplex and that is connected to the MNPS coupling facility structure. Refer to z/OS Communications Server: SNA Network Implementation Guide for complete information about setting up a sysplex that supports multinode persistent sessions.

Note: If both session partners are on the same VTAM prior to node failure, the sessions will not be recovered. If the application recovers on the same VTAM as its session partner, those sessions will be terminated.

For an application to be capable of multinode persistent sessions, PERSIST=MULTI must be coded on the APPL definition statement for this application progam. There are no differences in the programming interface for applications using single-node or multinode persistent sessions, with one exception. For both functions, the application must enable persistence as described in “How an application establishes persistence” on page 67, and for both functions the application can indicate support of forced takeover processing by using the SETLOGON OPTCD=PERSIST macroinstruction. The exception in the interfaces between the two types of persistence is that in order to initiate MNPS forced takeover, an application needs to specify an additional parameter on the ACB macroinstruction, as described in “Response to an application failure with MNPS” on page 70.

If one of the following operator commands is issued prior to a node failure, multinode persistent sessions support is overridden:

- HALT QUICK
- Normal HALT
- VARY INACT, TYPE=FORCE, ID=*appl_name*, where *appl_name* is the name of a persistence-enabled application.

A HALT CANCEL command does not override multinode persistent sessions and the session remain recoverable.

Response to a node failure

When a node failure occurs that affects an application program with persistence enabled, information required to restore the session has been saved in the multinode persistent sessions coupling facility structure. Other VTAM nodes in the sysplex detect the failure and the application enters recovery pending state.

The persistent session timer (PSTIMER), if specified on the application's SETLOGON macroinstruction, is started on all VTAMs connected to the multinode persistent sessions structure. The PSTIMER indicates the amount of time an application can remain in recovery pending state. If the timer expires, one of the VTAMs in the sysplex performs cleanup of the coupling facility structure for the application program data. If PSTIMER is not set, the application remains in recovery pending state until the ACB is reopened by a VTAM in the sysplex.

Response to an application failure with MNPS

Opening the ACB during recovery from a node failure:

During the recovery process, the application program issues an OPEN ACB, using the application network name of the original application, to reconnect to VTAM. Applications using multinode persistent sessions should be structured so that the OPEN macroinstruction is issued early in the application's processing. This allows VTAM to perform recovery processing in parallel with other application start-up work. Upon return of the OPEN ACB macroinstruction, the recovery VTAM indicates that this application is a persistent instance (the opening application program has recovered an ACB that was previously used by an application program that was capable of persistence). Recovery can occur on the same VTAM that the application program resided on (if that VTAM has been restarted) or a different VTAM.

If the recovery VTAM does not have connectivity to the coupling facility, OPEN processing continues as if it were an initial OPEN. If VTAM regains connectivity to the multinode persistent sesions coupling facility structure and determines that the application of the same name has opened its ACB in this sysplex, VTAM schedules the TPEND exit of the recovering application with a return code of 12.

During multinode persistent sessions recovery OPEN processing, the recovery VTAM obtains information needed to re-establish sessions from the coupling facility and updates the coupling facility to show this VTAM as the owning CP of the application program. The coupling facility is updated to show that the application is enabled for persistence and the application enters *recovery in progress* state. The recovery VTAM cancels its persistent sessions timer. Other VTAMs in the sysplex will not clean up the coupling facility structure data when their persistent sessions timers expire.

During OPEN processing, VTAM verifies that the following conditions are true.

- Original application is in a multinode persistent sessions recovery pending state
- Recovering application is capable of persistence
- Application's MACRF value matches the original application's MACRF value.
- Recovering application's APPC characteristics are identical to the original application's APPC characteristics. The following operands on the APPL definition statement are checked [1]:
  - SECLVL
  - VERIFY
  - SYNCLVL
  - LIMQSINT
- Recovering application's full-duplex capability (as specified on the ACB) is identical to the original application's full-duplex capability.
- Recovering application program's application-capabilities vector matches the original application program's application-capabilities vector. See "Vector lists" on page 58 for more information about this vector.
- If the original application was a generic resource, then the recovery VTAM can provide the generic resource function and is connected to the same generic resource structure (for example, STRGR specification matches).
- Recovery application program cryptographic support level is at least at the same level, or higher, than the cryptographic support level used by the original application. See "How VTAM determines the level of cryptography for a cryptographic session" on page 133 for information about the ranked levels of support for cryptography.
- Recovering application's support for network-qualified names matches that of the original application.
- Recovery application program Triple DES encryption support requirement is the same or higher than the original application program's requirement.

The recovering application does not need to re-establish persistence (SETLOGON OPTCD=PERSIST). When the application program opens an ACB for which persistence had been previously enabled, persistence remains enabled. The application can issue the SETLOGON OPTCD=PERSIST nonetheless to ensure that the proper setting for the FORCETKO PARMS operand is in effect.

Planned Takeover OPEN:

Multinode persistent session support allows an alternate application, on a different VTAM node, to take over an application that is in a "SNPS recovery pending" state (entered when CLOSE ACB is issued or the application abends while in the enabled MNPS state). If OPEN ACB is issued in this state, VTAM will proceed as normal until the point where VTAM as about to initiate recovery.

At that time, the takeover VTAM will signal the current owning VTAM, informing it that the application wants to move. The owning VTAM will prevent any further updates to the application data in the coupling facility from the original application program, and will prevent any new data from flowing across RTP pipes associated with the application. The current owner then replies to the

1. For SECLVL, VERIFY, and SYNCLVL, the recovery VTAM verifies that the values of these parameters are the same as what was coded at the original VTAM. For LIMQSINT, the recovery VTAM verifies that LIMQSINT is defined if it was also defined at the original VTAM. Different intervals for LIMQSINT are allowed at the two VTAMs.

takeover VTAM, and initiates a non-persistent CLOSE ACB operation for the original application program (as if the PSTIMER had expired) to clean up awareness of the application at the original owner.

When the reply signal arrives at the takeover owner, an OPEN ACB response is returned to the application program, and recovery continues as in the case of a node failure.

If, for some reason, the MNPS takeover request was rejected by the current owning VTAM, the recovery OPEN ACB would be failed with an OPEN error code of X'58'. The recovering application does not need to reestablish persistence (SETLOGON OPTCD=PERSIST). If the application program opens an ACB for which persistence had been previously enabled, persistence remains enabled; however, the application can issue the SETLOGON OPTCD=PERSIST to ensure that the correct setting for the FORCETKO PARMS operand is in effect.

VTAM treats failure of an MNPS-capable application in the same manner as described for SNPS applications (see "Application program recovery with single-node persistence enabled" on page 67), with the addition of changing the state of application in the coupling facility to SNPS recovery pending. If the application reopens its ACB on the same VTAM, processing would again be identical to the processing of VTAM for SNPS applications, with the addition of changing the coupling facility state back to enabled.

Forced Takeover OPEN:

A second variation of MNPS takeover OPEN does not require that the current active application image enter the SNPS recovery pending state before allowing another node in the sysplex to attempt acquisition of the application. The takeover node can signal to the current owning node that the application wants to move in a much wider range of states (enabled, takeover in progress, and the suspect state being the additional acceptable states).

When the current owning VTAM receives a forced takeover request, it will first initiate a persistent CLOSE ACB for the application, which generate a steady state image of the application and its sessions for the takeover node. After the steady state image of the application is completed and the application has successfully closed, the owning node signals the takeover VTAM, much as is done in the planned takeover processing, that ownership has been be transferred. If the ownership is transferred, the former owning VTAM initiates a non-persistent CLOSE ACB, as planned takeover, to clean up the local image of the application. Recovery processing at the takeover VTAM is identical, after ownership has been successfully transferred.

An application must indicate that it wants to participate in forced takeover processing, and the takeover OPEN ACB must be specifically marked as being a forced takeover OPEN ACB. An OPEN ACB is denoted as being capable of initiating an MNPS forced takeover OPEN ACB by specifying (PARMS=(FORCETKO=YES)) on the ACB macroinstruction. The application indicates a willingness to be a recipient of an MNPS forced takeover by specifying PARMS=(FORCETKO=ALL) or PARMS=(FORCETKO=MULTI) on the SETLOGON OPTCD=PERSIST macroinstruction.

Guideline: After MNPS recovery processing, even though persistence is enabled by default for the application, you should issue SETLOGON OPTCD=PERSIST

with PARMS=(FORCETKO=ALL) or PARMS=(FORCETKO=MULTI) specified to ensure that the application indicates support of forced takeover processing while on this node.

Tip: An application that accepts MNPS forced takeover requests must consider carefully the PSTIMER setting chosen, if one is used at all. A persistent CLOSE ACB for the application is driven by VTAM as part of the forced takeover logic, and after that has completed successfully a non-persistent CLOSE ACB is driven. The application PSTIMER interval value should be at least long enough to allow the internal VTAM processing to occur between the completion of the persistent CLOSE and the start of the non-persistent CLOSE. An interval of 30 seconds or more is necessary for this situation.

Restoring the sessions pending recovery

For information pertaining to how an application identifies and restores sessions pending recovery, see "Restoring sessions pending recovery" on page 137. For multinode persistent sessions, any sessions that involve partners on the same VTAM that recovers a persistence-enabled application are not maintained. Also note that the CID of a session recovered after a node failure is not the same as the CID of the session before the failure.

If the application is going to just clear, (for example, reset by way of SESSIONC CONTROL=CLEAR and SESSIONC CONTROL=SDT), the session after doing a OPNDST RESTORE, rather than resuming the session traffic transparently, then consider setting NIBTRNCK when establishing the session with the OPNDST (ACQUIRE or ACCEPT) or OPNSEC. This will reduce the overhead required to write the session state date to the coupling facility during session traffic.

Closing an application program

A VTAM application program closes itself by issuing a CLOSE macroinstruction that specifies the program's ACB. The CLOSE macroinstruction is used in the same way as the OPEN macroinstruction. Normally, the CLOSE macroinstruction is issued in the communication part of the VTAM application program and cannot be issued from an exit routine. The CLOSE request informs VTAM to terminate the SSCP-LU session with the application program and to mark the application program as no longer active in the VTAM network. For example:

```
CLOSPRG     CLOSE         ACB1
```

When a program closes, VTAM terminates all sessions and posts any outstanding operations as complete. In addition, the application program can neither issue the SENDCMD or RCVCMD macroinstructions nor access the vector lists that the ACB points to.

For additional information on the use of CLOSE by a program operator application (POA), refer to Appendix L, "Program operator coding requirements," on page 875.

For information on the use of CLOSE with persistent LU-LU session support, see "Using persistent LU-LU session support" on page 66.

The application program can learn that it should close its ACB in any of the following ways:

- The application program can determine for itself that it should close (perhaps by determining the time of day).

- The application program can receive a special text or data request, either from an LU or from the VTAM operator, indicating that the application program should close operations.
- The TPEND exit routine of the program can be entered, either because the VTAM operator has issued a HALT or VARY NET,INACT command or some abnormal event has caused VTAM to end, or because another instance of the application is being opened and this application needs to end as part of forced takeover processing.

Program initiates closing

For a normal end to operations, the application program can send a final request on all sessions. For an error or special condition, it can send the final request as well as store information about the nature of the error.

A VTAM application program closes itself by issuing a CLOSE macroinstruction that specifies its ACB. The CLOSE request notifies VTAM to terminate the SSCP-LU session with the application program and to mark the program as no longer active in the VTAM network. When VTAM processes a CLOSE ACB, and that application is not enabled for persistence, it terminates all of the sessions related to that ACB. If the application is enabled for persistence, then the sessions are maintained for the duration of the time specified by the persistent sessions timer (PSTIMER on the SETLOGON).

Note: For the extended recovery facility (XRF) sessions, during a normal end to operations:

- If the primary application program issues CLOSE to terminate its sessions, it also closes the backup extended recovery facility (XRF) sessions.
- If the primary application program issues CLSDST, it terminates an individual backup XRF session.
- If the backup application program issues CLOSE, it terminates the backup sessions and closes the backup application program.

Program receives a closedown message

The application program can close as the result of a special request from some element in the network. This occurs if closing the application program depends on a situation remote from the host processor (and the VTAM operator cannot be informed about the situation). For example, suppose a terminal operator at an LU in Chicago knows that Chicago is always the last user of an application program that is in the host processor in New York. When all terminal operators in Chicago have finished using the application program, a terminal operator in Chicago sends a special request to the VTAM application program in New York, telling it to close its operations. The VTAM application program then closes in an orderly fashion, notifying the VTAM operator at the host processor.

TPEND exit routine is entered

The TPEND exit routine is entered when the VTAM operator deactivates the application program or issues a HALT command or when, because of an internal error or problem, VTAM stops or abnormally ends.

In addition, for SNPS processing, VTAM schedules the TPEND exit in response to an OPEN ACB from an alternate application that wants to take over sessions from

an application that has enabled the persistent LU-LU session support. The TPEND exit routine may also be scheduled if an active application program with the same name has enabled persistence on another VTAM when this VTAM connects to the multinode persistent sessions coupling facility structure. Specifically, this occurs when a persistence-enabled application program is open and VTAM is not connected to the coupling facility. Later, when VTAM connects to the coupling facility, it detects an open application program with the same name that is already enabled for persistence.

VTAM might also drive the TPEND exit upon receipt of an MNPS forced takeover request from another node in the sysplex. The requested CLOSE ACB in this situation is treated as a persistent CLOSE ACB, allowing this VTAM to create a stable image of the application session and underlying HPR pipe data for the recovering node to use for rebuilding the sessions and pipes remotely.

VTAM can schedule the TPEND exit in response to three kinds of HALT commands:

- A standard HALT command, which contains neither the QUICK nor the CANCEL operand
- A HALT NET,QUICK command, which initiates a *quick closedown*
- A HALT NET,CANCEL command, which initiates a *cancel closedown*.

A VTAM operator can choose to simultaneously end all application programs running under VTAM. To do this, the operator issues a standard HALT command or a HALT NET,QUICK command. Neither command, however, is completed (that is, VTAM is not halted) until all application programs have closed their ACBs. The following sections describe the specific actions taken by the TPEND exit routine in response to the different HALT commands.

When the TPEND exit routine is entered, register 1 contains the address of a 2-word parameter list in which:

- Word 1 of this parameter list contains the address of the ACB of the application program being shut down.
- Word 2 contains one of the following codes that indicates the reason for entry to the exit routine:

  0 The VTAM operator issued a standard HALT command.

  4 The VTAM operator deactivated the application program or issued a HALT NET,QUICK command, or VTAM is halting itself in an orderly fashion because of an internal problem.

  8 The VTAM operator issued a HALT NET,CANCEL command, or VTAM is being abnormally terminated.

  12 An alternate application has issued an OPEN ACB for the same application network name that this application has opened. The new application image is either located on this VTAM, or is on a different VTAM in the sysplex and has notified this VTAM of the request using an MNPS forced takeover signal. Because the alternate application has taken over, the original application must issue CLOSE ACB.

  See “TPEND exit routine” on page 263 for complete descriptions of the reason codes that may be present in word 2 of the parameter list.

For codes 0 and 4, the TPEND exit routine should take action as indicated in the following sections. For codes 8 and 12, the exit routine should immediately return control to the mainline program, where a CLOSE macroinstruction should be issued.

If an application program does not have a TPEND exit routine or if that exit routine cannot be scheduled, the application program is abnormally terminated.

For more information on the TPEND exit routine, see its description in Chapter 7, "Using exit routines," on page 219.

Action for a standard HALT command

The VTAM application program responds to a notification of a HALT command (HALT NET without the QUICK or CANCEL operand) as a request from VTAM to close the application's operations. Additionally, the standard HALT command indicates that VTAM waits for the application program to close operations in an orderly manner. VTAM does not complete the standard HALT processing until all application programs have issued a CLOSE macroinstruction.

While VTAM is halting (because of a standard HALT command, a HALT command with the QUICK or CANCEL operand, or an abend), it stops new sessions from being established. During HALT processing, the application program need not issue a SETLOGON OPTCD=HOLD macroinstruction to prevent new sessions from being established.

When VTAM receives a standard HALT command, it prevents any new application programs from associating themselves with VTAM (by opening their ACBs) and prevents application programs from establishing new sessions. VTAM allows the application programs to continue communications on active sessions. For each application program, VTAM schedules the TPEND exit routine (if the program has one) and passes code 0 in the parameter list.

Under these conditions, the application program does not have to immediately close its ACB. The TPEND exit routine can inform other parts of the application program that the standard HALT command has been issued. It can do this by posting an ECB or by setting a switch that is checked by other parts of the application program. The application program can continue communications but should end them as soon as it can. It should then issue a CLOSE macroinstruction.

Note: The CLOSE macroinstruction cannot be issued in an exit routine; it must be issued in the mainline program.

If the application program has no TPEND exit routine and the VTAM operator issues a standard HALT, the application program has no immediate way of knowing that the HALT command has been issued. The application program continues communicating with LUs until the VTAM operator cancels the application program or until VTAM terminates (because the VTAM operator has entered a HALT NET,QUICK command).

Action for HALT NET,QUICK command or for VTAM-initiated HALT

An application also enters its TPEND exit routine when the VTAM operator deactivates the application program or issues a HALT NET,QUICK command or

when VTAM enters halt-quick processing because of an internal error. In these cases, VTAM closes down the network rapidly.

After it receives a HALT NET,QUICK command or after it enters halt-quick processing, VTAM does not allow any new application programs to associate themselves with VTAM (by opening their ACBs), nor does it allow application programs to establish new sessions. For application programs already in session with LUs, VTAM does not complete any new RPL-based communication requests. Any pending communication request is marked complete, with (RTNCD,FDB2)=(X'10',X'03') to indicate that the operation was canceled because of a quick closedown.

When an application program determines that a quick closedown is in progress, it should close its ACB as soon as possible. The TPEND exit routine learns of the quick closedown by finding code 4 in the parameter list when it is entered. That exit routine should do a minimum of closedown processing and return control to VTAM as soon as possible so that the mainline program can issue the CLOSE macroinstruction.

The user should be aware that, after the TPEND exit routine returns control to VTAM, the halt-quick situation does not prevent VTAM from scheduling the application program's other exit routines (such as the LOSTERM exit routine). Because of that, the TPEND exit routine should set a quick-halt-in-progress switch, which the application program should test at the beginning of each exit routine. When the switch is on, each exit routine should immediately return control to VTAM. The TPEND exit routine should also set a switch or post an ECB to signal the mainline program to close the ACB as soon as possible.

Action for HALT NET,CANCEL command or abnormal termination

VTAM's receipt of a HALT NET,CANCEL command or a VTAM abnormal termination also causes entry to the TPEND exit routine. For either event, VTAM interrupts any RPL-based operation and does not complete it (that is, the RPL is not marked as complete and no ECB is posted or RPL exit routine scheduled). VTAM rejects any RPL-based VTAM macroinstruction with (RTNCD,FDB2)=(X'10',X'03') or (X'14,'X'10'). Only CLOSE can be issued. Therefore, when the TPEND exit routine detects code 8 in the parameter list it receives, the exit routine should set a switch or post an ECB to inform the mainline program that it should immediately issue the CLOSE macroinstruction. The exit routine should then return control to VTAM so that control can be given to the mainline program.

Action for a takeover OPEN

An application program that is enabled for persistence enters its TPEND exit routine under the following conditions:

- For single-node persistent sessions, an alternate application program that is capable of persistence issues OPEN ACB to takeover the sessions from the original application.
- For multinode persistent sessions, an application program with the same network name has enabled persistence on another VTAM when this VTAM connects to the multinode persistent sessions coupling facility structure. Specifically, this occurs when a persistence-enabled application program is open and VTAM is not connected to the coupling facility. Later, when VTAM connects to the coupling facility, it detects an open application program with the same name that is already enabled for persistence.

- For multinode persistent sessions, an additional reason for the TPEND exit to be driven is the receipt of a forced takeover request from another node in the sysplex. Specifically, a copy of this same application has issued OPEN ACB on another node, and has specified FORCETKO=YES on the ACB macroinstruction. This setting indicates that the new copy of the application has reason to acquire ownership of the existing sessions, so this image of the application must be cleaned up before the new image can successfully acquire the sessions.

  Tip: An application can prevent this final form of TPEND exit usage by specifying PARMS=(FORCETKO=NONE) or PARMS=(FORCETKO=SINGLE) on its SETLOGON OPTCD=PERSIST macroinstruction.

In all cases, VTAM closes the application program by scheduling the TPEND exit routine with reason code 12.

For all TPEND conditions, VTAM does not allow the application program to respond to any operator commands, nor does it allow the application program to establish any new sessions. The original application program must issue CLOSE. Any RPL macroinstructions issued by the original application program will fail. Any pending or future communication request is marked with (RTNCD,FDB2)=(X'10',X'0D') to indicate that VTAM is inactive for that ACB. VTAM places any sessions in the recovery pending state and holds the allocated resources for future availability during session recovery.

Opening and closing an application program as a generic resource

The generic resource function enables multiple application programs that provide the same function to be known and accessed by a single generic name. LUs can initiate sessions using the generic name; VTAM determines which application program is used to establish the session. The LU knows the application program by its generic name only. This function enables VTAM to provide workload balancing by distributing incoming session initiations among a number of identical application programs, instead of overwhelming one application program.

An application program that is acting as a generic resource can have sessions with many LUs. One LU can have multiple sessions with a generic resource name; however, to maintain parallel session integrity, these sessions must be established with the same application.

VTAM maintains an awareness of the identity of the LU and the identity of the application that is acting as the generic resource for this session. This ensures that all subsequent session initiations from an LU that is in session with a generic resource resolve to the same application program.

Identifying an application program as a generic resource member

An application program must “associate” itself with a generic resource name before establishing LU-LU sessions. This association is accomplished when an application program identifies itself as a generic resource member by issuing SETLOGON OPTCD=GNAMEADD after OPEN ACB, but before SETLOGON OPTCD=START.

An application program can terminate the association with a generic resource name by issuing SETLOGON OPTCD=GNAMEDEL. Termination of the association

can occur even after the application program establishes LU-LU sessions. GNAMEDEL indicates to VTAM to stop using this generic resource member when distributing new LU-LU session initiations.

An application that is enabled for persistence in a sysplex that supports multinode persistent sessions will have its association with its generic resource name terminated during recovery of that application from a node failure. In this case, the application must issue the SETLOGON OPTCD=GNAMEADD macroinstruction to be associated with a generic resource name.

Initiating a session using a generic resource name

When an LU initiates a session with a generic resource, VTAM determines whether that LU is currently in session with a generic resource. If it is, VTAM ensures that subsequent sessions resolve to the same application program. If the LU is not in session with a generic resource, VTAM resolves the generic resource to the next available application program. VTAM then tracks the affinity created between the LU and the application program. VTAM maintains the affinity to ensure that all session requests from an LU to a generic resource name resolve to the same application.

Initiating a session using the name of an application that is a member of a generic resource

If an LU requests sessions using the application network name instead of using the generic resource name, VTAM establishes the session. However, if the LU later uses the generic name, session initiation resolves to *another* application known by that generic name.

Closing an application program that is a member of a generic resource

Before an application (that is a member of a generic resource) issues CLOSE ACB, all LU sessions *should* be terminated. The application program issues CHANGE OPTCD=ENDAFFIN, which allows VTAM to end the affinity between the application program and the LU. After all sessions with LUs have been terminated, and the generic resource application program has terminated affinities with all LUs, the application can issue CLOSE ACB. If the application issues CLOSE ACB without terminating the LU-to-application affinities, the affinities may persist.

VTAM also provides the CHANGE macro with OPTCD=ENDAFFNF (End Affinity Force). This option allows the affinity with a specific LU to be terminated immediately, even if sessions are still active.

Ownership of affinities between LUs and application programs

An LU-to-application program affinity is controlled by either VTAM or the application program. An application program using generic resources can only use the CHANGE macroinstruction, to terminate the affinity between an LU and itself, if the application is the owner of the affinity. The application program controls this affinity if any of the following are true:

- LUAFFIN=NOTAPPL was not specified during session establishment and the session uses LU 6.2 sync point services.

- LUAFFIN=NOTAPPL was not specified during session establishment and the session uses LU 6.2 limited resource support.
- LUAFFIN=NOTAPPL was not specified during session establishment and the session uses LU 6.1 protocols.
- LUAFFIN=NOTAPPL was not specified during session establishment and the application program specified SETLOGON OPTCD=GNAMEADD with AFFIN=APPL specified on the NIB.
- LUAFFIN=APPL was specified during session establishment.

VTAM controls the affinities in all other situations.

See "NIB—Create a node initialization block" on page 433 for more information.

Terminating affinities that are application-owned

A generic resource application, acting as the PLU or the SLU, terminates the affinity with an LU by issuing the CHANGE OPTCD=ENDAFFIN macroinstruction. However, termination of the affinity cannot occur until all sessions between the LU and the application program are terminated. If CHANGE OPTCD=ENDAFFIN is issued and there is at least one active session with that LU, the CHANGE is rejected.

A generic resource application, acting as the PLU or the SLU, can also force the termination of the affinity with an LU by issuing the CHANGE OPTCD=ENDAFFNF macroinstruction. When using the force option (OPTCD=ENDAFFNF), the affinity will immediately be terminated without regard to sessions.

If there is a new session pending, VTAM is given ownership of the affinity. If the pending session fails, VTAM terminates the affinity between the LU and the application program. If the pending session becomes active, the application is notified and control of the affinity is determined by the factors listed in "Ownership of affinities between LUs and application programs" on page 79.

Terminating VTAM-owned affinities

VTAM owns and controls all affinities that are not controlled by a generic resource application. In all cases, VTAM owns and controls this affinity before an application is notified that a session is being established. For example, if an LU 6.1 session fails prior to the application program being notified of that session, VTAM, as owner, terminates the affinity between the LU and the application program.

After the last session between the LU and an application of the generic resource ends, VTAM terminates the affinity between the LU and the application.

In a sysplex that supports multinode persistent sessions, affinities are maintained for a generic resource application that has enabled persistence, even after the VTAM (or operating system) that owns the application fails. This is true as long as the application remains in recovery pending state, which typically is the duration of the persistent sessions timer (PSTIMER). The application's partner LUs still maintain their affinities with the application, even though that application is not available for new sessions and is not processing any existing session traffic.

Chapter 5. Establishing and terminating sessions with logical units

Before two LUs can communicate, VTAM must establish a session between them. In any session between logical units (LUs), one LU, the primary logical unit (PLU), acts as the primary end of the session (also called the primary half-session); and the other LU, the secondary logical unit (SLU), acts as the secondary end of the session (also called the secondary half-session). For some LU types, the PLU has more control over communications than does the SLU. "Sessions" on page 3 shows an example of the SNA session-initiation procedure.

The type of macroinstruction issued by the initiator of a session determines which LU is to be the PLU and which LU is to be the SLU in that session. If the application program issues a REQSESS macroinstruction to initiate the session, that application program becomes the SLU in the established session. If an application program issues a SIMLOGON macroinstruction, it becomes the PLU in the session. The PLU actually establishes the session by sending a BIND request to the SLU.

The system services control point (SSCP) assists the LUs in establishing and terminating sessions. Although only one SSCP (a single-domain network) is discussed in the following sections, the session-establishment and termination procedures involved for a multiple-domain network are very similar, with the SSCPs cooperating to give the appearance of a single SSCP to the LUs involved.

Defining LUs

Before you can establish a session, both LUs must be active, connected, enabled, and available.

An LU is active after a VARY ACT command applied to it completes successfully. The command can be for the LU specifically, or for a set of resources including the LU. The VTAM operator can explicitly issue the command, or the command can be issued implicitly within the list of resources activated upon initialization of VTAM. Refer to z/OS Communications Server: SNA Operation for further information about the VARY ACT command.

An LU is connected when a physical path exists from the node containing the SSCP to the node containing the LU, and a session between the SSCP and the LU is established on this path (not applicable for independent LUs).

LUs on nonswitched lines are always connected if they are active. LUs on switched lines are connected only when a link is established over the switched network. For LUs belonging to physical units (PUs) that can only dial in to the network, dial-in must occur before a session can be established. If dial-out capability is supported, VTAM dials out to the PU if the PU is not connected and if a session is being initiated with an LU belonging to the PU. Channel-attached peripheral PUs exhibit some of the characteristics of switched PUs and can be connected or disconnected while they are active.

LUs can indicate whether they are willing to have sessions. An enabled LU is willing to have sessions. A disabled LU does not currently want a session, but it might in the future. Therefore, if the session-initiation request specifies that the

© Copyright IBM Corp. 2000, 2013

session can be queued until the LU is enabled, it is appropriate to queue the session. An inhibited LU does not want a session in the foreseeable future, and no session should be queued for it.

Separate enabled states exist for LUs acting as PLUs and as SLUs. For example, an application program LU that has opened its ACB with MACRF=LOGON, but has not yet issued SETLOGON OPTCD=START, can act as a PLU, but cannot act as an SLU. While in this state, the application program must issue OPNDST OPTCD=ACQUIRE to establish any sessions. When SETLOGON OPTCD=START is issued, the application program can act as both a PLU and an SLU. An application program LU that has opened its ACB with MACRF=NLOGON is enabled as a PLU and is inhibited as an SLU. It cannot act as the SLU. Table 6 summarizes the interaction between SETLOGON and the ACB's MACRF operand.

An LU is available if it is active, connected, enabled, and not at its session limit. The session limit is the maximum number of concurrent LU-LU sessions that the LU can support. For LUs that support LU 6.2 protocols, session limits can be dynamically defined. For dependent LUs, the session limit is one. VTAM application program LUs and independent LUs can support an unlimited number of sessions.

Note: These notes relate to the following table. SETLOGON OPTCD=STOP does not cause the SSCP or VTAM, on behalf of the LU issuing SETLOGON OPTCD=STOP, to take any action to prevent session initiation. It does cause the SSCP to set an indicator that can be queried by another application issuing INQUIRE OPTCD=APPSTAT.

Table 6. Interaction between SETLOGON and the ACB MACRF operand

Application program's ability to act:	**MACRF=NLOGON**	**MACRF=LOGON**	**MACRF=LOGON**	**MACRF=LOGON**
	(SETLOGON OPTCD=START, QUIESCE, or STOP, if issued, is rejected with (RTNCD,FDB2)= (X'14', X'61')).	(Before SETLOGON OPTCD=START)	(After SETLOGON OPTCD=START, but before SETLOGON OPTCD= QUIESCE)	(After SETLOGON OPTCD= QUIESCE)
As an SLU	***Inhibited***			
		Disabled	***Enabled***	***Inhibited***
	SSCP rejects Initiate requests (sense code= X'083A0000') for sessions in which this LU acts as an SLU.	If queuing is requested in Initiates, SSCP queues requests for sessions in which this LU acts as an SLU; otherwise, it rejects the Initiates (with sense code=X'083A0002').	SCIP exit routine scheduled with BIND for the requested session.	SSCP rejects Initiate request (sense code= X'0801000F') for sessions in which this LU acts as an SLU.

Table 6. Interaction between SETLOGON and the ACB MACRF operand (continued)

Application program's ability to act:	MACRF=NLOGON	MACRF=LOGON	MACRF=LOGON	MACRF=LOGON
As a PLU	*Enabled*	*Enabled*	*Enabled*	*Inhibited*
	VTAM rejects all CINITs (except those immediately completing OPNDST OPTCD= ACQUIRE) (sense code= X'08010000') by VTAM for the LU.	CINIT either immediately completes OPNDST OPTCD= ACQUIRE or ACCEPT, or is queued at LU to await OPNDST OPTCD= ACCEPT.	CINIT does one of the following: 1. Completes OPNDST OPTCD= (ACQUIRE, ACCEPT) 2. Schedules the LOGON exit routine 3. If there is no LOGON exit routine, is queued at LU to await OPNDST OPTCD= ACCEPT.	SSCP rejects Initiate requests (sense code = X'0801000A') for sessions in which this LU acts as a PLU.

Stages of session establishment

Creating a session between two LUs (an LU-LU session) is a three-stage process. The first stage begins when an LU requests that a session be established. This request is in the form of an SNA Initiate request sent to the SSCP. The SSCP determines whether the LUs are available for sessions. If both LUs are available, a pending active session is created between them. If both LUs are active and connected, and at least one of them is not available (not enabled or at its session limit), and the session-initiation request indicates the session can be delayed until the LUs are available, then a queued session is created. When both LUs are available, the oldest queued session between them becomes a pending active session. Normally, sessions are queued in the order in which the Initiate requests are received. However, the Initiate request can indicate that the session should take priority over other queued sessions, and be put at the head of the queue. The queued session represents a request to establish a session. Queued sessions are not counted against the session limits for the LUs, so there can be many queued sessions waiting for a particular LU to become available.

The second stage begins when the pending active session is created (that is, both LUs are now available for the requested session). The SSCP sends a Control Initiate (CINIT) request to the LU that acts as the PLU in the session being established. The CINIT request is a request sent to the PLU to establish a session by sending a BIND request to the SLU. Pending active sessions are counted against the session limits for the LUs. Therefore, the number of active and pending active sessions for each LU cannot exceed the LU's session limit.

The third stage begins when the BIND request is sent from the PLU to the SLU. The BIND carries information concerning session protocols. The SLU examines the session parameter and if it is acceptable (or in the case of a negotiable BIND, if the SLU wishes to return a set of modified parameters), the SLU sends a positive response to the BIND. When the PLU receives the positive response, the session is considered to be active.

Note: Use the SETLOGON HOLD/START macroinstruction to control the number of session requests currently being processed. Refer to the SETLOGON macroinstruction, "SETLOGON—Modify an application program's capability to establish sessions" on page 543, for more information on SETLOGON HOLD/START.

Stages of session termination

Unlike session initiation, session termination can occur with or without SSCP assistance.

The SSCP assists directly when it is sent a Terminate request (which ends the session). The originator of the request can be either of the two LUs involved in the session or a third party (for example, the VTAM operator). The three types of Terminate requests are orderly, forced, and cleanup. Following are the descriptions of these types:

- **Terminate Orderly** causes a Control Terminate (CTERM) Orderly request to be sent from the SSCP to the PLU of the affected session. The session is not disrupted, and communication can continue. The PLU decides when, if ever, to end the session. It can start an application-program-dependent takedown procedure to end the session in an orderly manner, or it can ignore the request.

  Note: Terminate Orderly does not apply if the PLU is an independent LU.
- **Terminate Forced** causes a CTERM Forced request to be sent from the SSCP to the PLU of the affected session. In this case, UNBIND is sent automatically from the PLU and immediately terminates the session. Further session communication is impossible.
- **Terminate Cleanup** causes Cleanup Session (CLEANUP) requests to be sent from the SSCP to the PLU and SLU of the affected session. These requests inform the LUs that the session has been terminated and that further communication is impossible. When an LU receives a CLEANUP, it automatically sends an UNBIND to the other LU in the session. This ensures that the other LU is notified that the session has been ended (possibly because of a network outage) even if CLEANUP cannot be sent to it from the SSCP. The cleanup processing for a peripheral LU differs slightly from that done for a subarea LU, but the result in both cases is that the half-session at the LU (including any boundary function support of that LU) is reset and that UNBIND is sent to the session partner (the other LU in the session) to cause its half-session to be reset. The SSCP is notified when each half-session is reset so that the SSCP might reduce the session count for each LU.

A session can also terminate without the direct assistance of the SSCP. Either LU can send an UNBIND at any time to the other LU in the session. Each LU notifies the SSCP when its half-session is reset so that the LU's session count can be reduced.

In addition, certain network outages or VTAM operator actions (such as deactivating a link) can cause a session to be terminated. In this case, VTAM notifies the LU of the loss of the session. The LU then informs the SSCP so that the

LU's session count can be reduced. See “Session outage notification” on page 110 for a detailed discussion of session outage termination.

Sources of SNA Initiate and Terminate requests

Session-initiation requests can come from any of the following sources:

Device-type LU Initiate

Some device-type LUs can send formatted (bit-encoded) Initiate requests to VTAM as a result of some action, such as pressing a key on a terminal. Other device-type LUs allow terminal operators to enter a character-coded LOGON that is translated into a formatted Initiate request by the SSCP. Thus, the device-type LU Initiate permits a terminal or its operator to initiate a session; the application program need not have previously known of the device-type LU's existence.

Application program LU Initiate

A VTAM application program can initiate a session in which it acts as the PLU by issuing an OPNDST OPTCD=ACQUIRE or SIMLOGON macroinstruction. This permits an application program to initiate a session and is useful, for example, if a device-type LU (such as an unattended printer) cannot itself initiate the session, or if the time when a session is needed is known only by the application program. This requires that the application program have previous knowledge of the LU's existence. An application program can also initiate a session (for example, an LU-LU session between programs) in which it acts as the SLU by issuing a REQSESS macroinstruction. (REQSESS cannot be used when the requested PLU is an independent LU.)

Third party Initiate

A session can be initiated by an LU that is not one of the session partners, or by the SSCP. In VTAM there are two ways this can happen:

- CLSDST OPTCD=PASS. An application program acting as a PLU in a session can “pass” its session partner (the SLU) to another LU which acts as the PLU in a new session with that session partner. This permits the SLU to be passed from application program to application program in a sequence strictly controlled by the application programs involved. The types of session initiation previously discussed do not allow this; instead, for them, sessions are established in the order in which the corresponding Initiate requests are received by the SSCP.
- Network Definition or VTAM Operator Command. The VTAM operator can cause a session to be established with a particular application program (called the controlling application program) by issuing a VARY LOGON command for a device-type LU. This means that whenever the device-type LU has no LU-LU session, or for an independent LU that has no LU-LU session with the controlling application program but can have other LU-LU sessions, a session is automatically initiated between the controlling application program and the device-type LU; this automatic session initiation is not done if the last session that the device-type LU had is with the controlling application program and the session ended normally, for example, with CLSDST. A similar session is established by VARY ACT if the device-type LU is defined to have a controlling application program (by the LOGAPPL keyword on the LU definition statement). These techniques allow a session to be initiated without requiring either the application program or the device-type LU to do the session initiation. To each of the LUs, it appears almost as if the other LU had initiated the session.

Refer to z/OS Communications Server: SNA Operation for more information on the VARY LOGON and VARY ACT commands.

Application programs that support LU 6.2 protocols can request automatic activation of a specified number of sessions used for LU 6.2 conversations. Refer to z/OS Communications Server: SNA Programmer's LU 6.2 Guide for more information on the definition of LU 6.2 application programs.

Session-termination requests can come from the following sources:

Device-type LU Terminate

Some device-type LUs can send formatted Terminate requests to VTAM as a result of some action, such as pressing a key on a terminal. Other device-type LUs allow terminal operators to enter a character-coded LOGOFF that is translated into a formatted Terminate request by the SSCP. Terminate Orderly, Forced, or Cleanup can be used depending on what the specific LU supports.

Application program LU Terminate

VTAM application programs that are acting as PLUs can terminate the session by issuing the CLSDST macroinstruction to send an UNBIND request to the SLU.

Programs that are acting as SLUs can terminate the session by issuing the TERMSESS OPTCD=UNBIND to send an UNBIND request to the PLU. (If you use TERMSESS OPTCD=UNBIND, you must specify the CID in either the ARG or NIB operand. If you use the NIB operand, you cannot use NAME.)

SLU application programs can issue a TERMSESS macroinstruction to send a Terminate request (Orderly or Forced) to the SSCP. If an application program closes its ACB while it still has sessions, those sessions are terminated by sending a Terminate request (Cleanup) to the SSCP.

Third party Terminate

Something other than one of the session partners, such as the following, can terminate a session:

- Network Outage. If a break occurs in the communication path for the session, the session terminates.
- VARY INACT. If one of the LUs in a session is deactivated by the VTAM operator, the session terminates. Terminate Forced is used for immediate deactivation (VARY INACT,I) and Cleanup is used for forced deactivation (VARY INACT,F).
- VARY TERM. The VARY TERM command sends a Terminate request to the SSCP. The TYPE operand on the VARY TERM command determines which type of Terminate request is sent. Terminate Orderly is sent for TYPE=COND, Terminate Forced is sent for TYPE=UNCOND, and Terminate Cleanup is sent for TYPE=FORCE.
- Request Discontact. A PU can request that it be discontacted by the network. The PU can specify how existing sessions are terminated prior to the discontact.

For further details about the way a session can be terminated, see “Session outage notification” on page 110. “NSPE request” on page 105 and “Notify request” on page 106 discuss two requests that an application program can receive if that application program initiates a session that is then terminated before it is fully established.

Macroinstructions related to session establishment and termination

VTAM provides macroinstructions to establish and terminate sessions under specific circumstances. These include:

- SIMLOGON initiates sessions in which the application program acts as the PLU.
- OPNDST establishes a session between the application program and an LU.
- CLSDST terminates sessions in which the application program is acting as the PLU.
- REQSESS initiates a session in which the application program acts as the SLU.
- OPNSEC establishes a session in response to a BIND request from a PLU.
- SESSIONC with CONTROL=BIND is used by a SLU to reject a BIND request.
- TERMSESS terminates sessions in which the application program is acting as the SLU.

For more information on each macroinstruction, see Chapter 13, "Conventions and descriptions of VTAM macroinstructions," on page 371.

SIMLOGON macroinstruction

The application program that is functioning as a PLU uses the SIMLOGON macroinstruction to initiate sessions. SIMLOGON is also used to supply dial parameters when an application functioning as a PLU initiates a session with an LU associated with a PU that is coded in a switched major node.

SIMLOGON processing sends an Initiate request to the SSCP. After successfully receiving the Initiate request, the SSCP sends a CINIT request to the application program. VTAM can then schedule the LOGON exit routine. See "LOGON exit routine" on page 101 for a description of how the application program receives the CINIT request.

An application program must be authorized to issue the SIMLOGON macroinstruction (AUTH=ACQ on the APPL definition statement).

The SIMLOGON macroinstruction uses the RPL to pass information to VTAM. The RPL points to a list of NIBs which contain the names of the LUs with which sessions should be initiated. The RPL indicates whether sessions should be initiated for as many NIBs in the list as possible, or just one of them. The RPL also indicates whether the session-initiation request can be queued.

The RPLAREA field can point to a user area containing up to 255 bytes of user data to be sent with the session-initiation request. The AREALEN field in the RPL must contain the length of the data. This user data is passed to the PLU in the CINIT request.

The NIBs, pointed to in the RPL, must be in contiguous storage with LISTEND=NO specified in all the NIBs except the last one, which must specify LISTEND=YES. The NAME field of each NIB contains the name of the LU with which to initiate a session.

In addition, if network-qualified names are being used, the NETID field of each NIB can contain the name of the network where the LU is located. The LOGMODE field of each NIB contains the logon mode name of a session parameter set and class of service suggested for use in establishing the session. If the application

program is supplying the dial parameters, the NIB contains the address of the application program's dial parameter list. See “Establishing parameters for sessions” on page 122 for more information about session communication.

If the USERFLD in the NIB is not 0, it is a correlator for the Initiate procedure, and is passed to the LOGON exit routine with a CINIT request if the procedure is successful. If the procedure fails, the USERFLD in the NIB is passed to the NSEXIT exit routine with a Notify request. If the USERFLD is 0, a Network Services Procedure Error (NSPE) request is passed to the NSEXIT routine if the procedure fails. The correlator permits the application program to determine if a CINIT or Notify is caused by SIMLOGON and, if so, which one. If the application program and the SLU support parallel sessions, multiple concurrent SIMLOGONs can be outstanding for the same SLU. In that case, a unique correlator for each SIMLOGON would be needed, if the application program wanted to match the CINIT request or Notify request with the SIMLOGON that caused it. That is because the PLU and the SLU names passed to the LOGON exit routine do not uniquely identify the session-initiation request if the same PLU and SLU are involved in multiple CINIT requests.

Note: When using OPTCD=ASY during session initiation, an application program task can be suspended unless provided with an RPL exit routine. See “Initializing a session” on page 44 for information about using SIMLOGON OPTCD=ASY during session initiation.

Initiating sessions with all LUs in a list

When OPTCD=CONALL is specified, SIMLOGON attempts to initiate sessions with every LU in the list. SIMLOGON is considered successful if at least one session is initiated. If an LU appears several times in the list and both the application program and the LU support parallel sessions, SIMLOGON attempts to initiate parallel sessions between the application program and the LU. OPTCD=CONALL is posted complete only after an Initiate has been sent and a response (positive or negative) has been received for every NIB in the list.

Initiating a session with the first LU in a list

When OPTCD=CONANY is specified, SIMLOGON initiates a session with each LU in the list, in turn, until the SSCP indicates that the Initiate is successful. Thus, SIMLOGON creates one queued or pending active session with the first LU in the list for which that is possible. SIMLOGON is considered successful if a session is initiated. OPTCD=CONANY is posted complete as soon as a positive response is received for an Initiate or when the end of the NIB list has been reached.

Queuing Initiates

If OPTCD=NQ is specified, each Initiate request sent to the SSCP indicates sessions cannot be queued, and the Initiate succeeds only if the LU is immediately available. If OPTCD=Q is specified, then each Initiate request sent to the SSCP indicates a session can be queued if the LU is disabled or at its session limit.

When OPTCD=Q, OPTCD=QALL or QSESSLIM or QNOTENAB determines how and when requests are queued. If the Initiate request is queued and the LU is active, connected, and not inhibited for sessions as an SLU, the Initiate succeeds. If the LU is disabled as an SLU or is at its session limit, a session is queued until the LU is available. If the LU is available, a pending active session is created immediately. If OPTCD=Q is specified, an additional OPTCD is available to qualify the conditions for which the Initiate is queued within VTAM. If OPTCD=QALL is

specified, VTAM indicates in the Initiate that the SSCP can queue the request if the LU is at session limit or is not enabled. If, however, queuing is desired in only one of these cases, OPTCD=QSESSLIM (LU at session limit) or OPTCD=QNOTENAB (LU is not enabled) can be coded. In each case, the Initiate request sent to the SSCP indicates the appropriate queuing conditions.

Notification to the session partner of a request for a session

SIMLOGON causes VTAM to notify the session partner of an SLU that is at its session limit that another PLU wants to establish a session with the SLU. To obtain this function, issue OPTCD=(Q,RELRQ). If the SLU is available, the session is initiated. If the SLU is at its session limit, the session is queued to wait for SLU availability; and the current session partner of the SLU, if it is a VTAM application program, is notified in its RELREQ exit routine that a session is queued for the SLU. The current session partner can terminate the session to make the SLU available, or it can ignore the RELREQ request (RELREQ does not apply if the PLU is an independent LU). For further information on the RELREQ exit routine, see "RELREQ exit routine" on page 253.

OPNDST macroinstruction

The OPNDST macroinstruction requests VTAM to establish a session between the application program and an LU, with the application program acting as the PLU. No matter how a session is initiated, OPNDST is used to accept the resulting pending active session in which the application program acts as the PLU. Additionally, OPNDST can be used to initiate a session with an LU and to accept the resulting pending active session in one operation. This is known as acquiring a session. Finally, the OPNDST macroinstruction is used to supply dial parameters when an application program functioning as a PLU initiates a session with an LU associated with a PU that is coded in a switched major node.

The NIB used for OPNDST specifies options regarding the processing of the session, such as whether a RESP exit routine should be scheduled when a response is received. The USERFLD in the NIB contains a 4-byte value to be associated with the session. This value is returned to the application program, in the USER field of the RPL used for a SEND, RECEIVE, RESETSR, or SESSIONC macroinstruction, when the associated communication operation is complete. This value might be a pointer to a user control block representing the session. Refer to "Node initialization block (NIB)" on page 115 for more information about what the NIB contains.

Accepting and establishing a single pending active session

OPNDST OPTCD=ACCEPT is used to accept a single pending active session in which the application program acts as the PLU, and thus establishes an active session without regard to the source of the pending active session. The application program is notified of a pending active session in its LOGON exit routine, if available, or by the completion of a queued OPTCD=ACCEPT that was waiting for a pending active session to accept. If the LOGON exit routine is available, the CINIT is included with the input parameters.

The RPL and the NIB identify which pending active session to accept. If OPTCD=ANY is specified in the RPL, the oldest pending active session is accepted. If OPTCD=SPEC is specified, the NIB identifies the pending active session. If the application program supports parallel sessions (PARSESS=YES is coded on the APPL definition statement), and the NIBCID field is not 0, the pending active

session identified by the CID is accepted. (The CID can be obtained from the LOGON exit parameter list.) If the NIBCID field is 0, or the application program does not support parallel sessions, the oldest pending active session between the application program and the LU named in the NIBSYM field is accepted.

If network-qualified names are used, the NETID field in the NIB (NIBNET) is used with NIBSYM to locate a network-qualified LU. If the SIMLOGON specifies a network-qualified name, the corresponding OPNDST OPTCD=ACCEPT should also specify a network-qualified name.

If the specified pending active session does not exist, action is dependent on the RPL queuing option. If OPTCD=NQ is specified, the OPNDST is rejected with (RTNCD,FDB2)=(X'00',X'09'). If OPTCD=Q is specified, OPNDST processing is suspended until a pending active session that can be accepted is created. If OPTCD=ANY is specified in the RPL, OPNDST waits for the first pending active session to be created. If OPTCD=SPEC is specified in the RPL, OPNDST waits for a pending active session with the LU identified in the NIBSYM field. In this case, the setting of the NIBCID field is ignored. Because the CID is assigned arbitrarily, it is not necessary to wait for a pending active session with a particular CID.

Once the application program identifies the pending active session to be accepted, a BIND request is sent from the application program to the SLU to establish the session. The BIND request carries the session parameters used to establish the protocols to be used for the session. See "BIND request" on page 91 for more information. If the SLU wants to go into session, it sends a positive response to the BIND; the session is established and OPNDST is posted complete. If the session cannot be established (for example, the SLU rejects the BIND request, or the path between the application program and the SLU is lost), OPNDST is posted complete indicating this exception condition with, for example, (RTNCD,FDB2)=(X'10',X'01').

Acquiring sessions

Initiating, accepting, and establishing an active session:
About this task

OPNDST OPTCD=ACQUIRE is used to initiate sessions with a set of LUs and then to accept the resulting pending active sessions and establish active sessions. This is equivalent to issuing a SIMLOGON macroinstruction and then accepting the resulting pending active sessions with OPNDST OPTCD=ACCEPT, except that the two operations are combined into one. The LOGON exit routine is not used for OPNDST OPTCD=ACQUIRE.

Use of OPNDST OPTCD=ACQUIRE requires authorization for the issuing application program (AUTH=ACQ on the APPL definition statement).

The NIBSYM field identifies the LU with which a session is desired.

The NIBNET field identifies the network that contains the LU if PARMS=(NQNAMES=YES) is specified at OPEN. OPNDST OPTCD=ACQUIRE supports lists of NIBs in contiguous storage with the LISTEND=NO option specified in all except the last NIB, which has LISTEND=YES. The RPL points to the first NIB in the list. The action taken with an NIB list depends on whether CONALL or CONANY is specified. If only one NIB is in the list, CONALL and CONANY have the same meaning.

Initiating, accepting, and establishing active sessions with all LUs in a list:

About this task

When OPNDST OPTCD=(ACQUIRE,CONALL) is specified, VTAM attempts to establish sessions with every LU in the list. OPNDST is posted complete only when an attempt (successful or unsuccessful) has been made to establish a session with every LU in the list. OPNDST is considered successful if at least one session is established. The same LU name can appear several times in the list, in which case an attempt is made to establish parallel sessions between the application program and the LU. This does not succeed unless both the application program and the LU support parallel sessions.

Initiating, accepting, and establishing an active session with the first LU in a list:
About this task

When OPNDST OPTCD=(ACQUIRE,CONANY) is specified, VTAM establishes a session with the first available LU in the list. OPNDST is successful and the OPNDST is posted complete if a session is established. If no session can be established with any LU in the list, OPNDST is posted complete, indicating this exception condition.

Note: When using OPNDST OPTCD=ACQUIRE during session initiation, an application program task could be suspended unless provided with an RPL exit routine. See "Initializing a session" on page 44 for information about OPNDST OPTCD=ACQUIRE during session initiation.

Restoring sessions

Restoring sessions pending recovery:
About this task

OPNDST OPTCD=RESTORE is used to restore sessions pending recovery. See "Restoring sessions pending recovery" on page 137 for more information.

Extended recovery facility session requests

The VTAM application program must specify, during OPNDST processing, whether the session request is for a primary extended recovery facility (XRF) or backup XRF session. This is accomplished by using the correct parameters on the OPNDST macroinstruction command, setting the appropriate BIND indicator specifying "control vectors included", and completing the XRF session-activation control vector using the NIB BNDAREA field when OPNDST is issued.

The XRF control vector must be appended to the end of the user data field in the BIND. The control vector indicator is the BINCTLV bit in the BINCMNP2 field.

For more information concerning the structure of the XRF session-activation control vector, see "XRF session activation control vector" on page 845.

BIND request

A BIND request is sent from the application program (which acts as the PLU) to the SLU to establish a session. The BIND includes the session parameters which define the protocols to be used on the session. The session parameters can be suggested by the initiator of the session (code BNDAREA=0 and LOGMODE=0 in the NIB to get the session parameters), another set of session parameters identified by a logon mode name (code BNDAREA=0 and LOGMODE=*logon mode name* [or

=C' ' for the default] in the NIB), or a set built by the application program (code BNDAREA= address of the session parameter). Refer to “Establishing parameters for sessions” on page 122 for more information about session communication.

You can use the OPNDST macroinstruction to pass a BIND image to VTAM. If the BIND contains user-data-structured subfields and you need to specify the network qualified PLU name, you can obtain the network ID from the network-name vector. This vector is part of the Resource-Identification Vector List, which is returned by VTAM after the PLU's ACB has successfully opened.

The BIND request can be negotiable or non-negotiable. A negotiable BIND permits the SLU the option of modifying the session parameters if they are unsuitable. If the BIND is non-negotiable and the session parameters are unacceptable to the SLU, the SLU must reject the BIND request to prevent the session from being established. To specify whether a BIND or BIND response is to be negotiable or non-negotiable, the application program must use the PROC=NEGBIND or PROC=NNEGBIND parameters in the NIB.

To send a negotiable BIND, specify PROC=NEGBIND in the NIB. In this case, the RPLAAREA field should point to an area into which the BIND response can be placed, and AAREALN must be the length of this area. Only one NIB in an NIB list can specify negotiable BIND, as there is only one alternate area field in which to put the BIND response. If OPNDST is successful, VTAM places the BIND response into the location pointed to by AAREA and sets the RPL's ARECLEN field to the actual length of the response. The response is truncated if ARECLEN is greater than AAREALN. The PLU application program should examine any changes made to the session parameters by the SLU. If they are acceptable, communication can begin. If not, the application program must issue CLSDST to terminate the session. The SLU can send a non-negotiable BIND response to a negotiable BIND request. This implies that no changes are desired by the SLU (the SLU has found the parameters in the BIND to be acceptable and agrees to establish the session). The PLU can examine the BIND type field in the BIND response to determine whether the response is negotiable or non-negotiable.

To send a non-negotiable BIND, specify PROC=NNEGBIND in the NIB. If the SLU sends a negotiable BIND response to a non-negotiable BIND request, VTAM automatically terminates the session and OPNDST fails with (RTNCD,FDB2)=(X'10',X'01') and sense code = X'084E0000'. This same return code and sense code are set for any other VTAM-detected errors in a received BIND response (for example, a transmission services profile that is not valid).

CLSDST macroinstruction

The CLSDST macroinstruction is used to terminate sessions in which the application program is acting as the PLU. CLSDST sends UNBIND requests from the PLU to the SLU to terminate sessions for which BIND has been sent (active and pending active sessions), and rejects any CINIT requests for which OPNDST OPTCD=ACCEPT has not yet been issued. CLSDST will also terminate a queued session by sending a Terminate. Additionally, CLSDST can be used to initiate the next session for the SLU.

Scope of CLSDST

The scope of a CLSDST operation is the set of sessions to be terminated. The CLSDST macroinstruction uses the RPL, and optionally, an NIB to identify the scope of the CLSDST. If no NIB is supplied, then the RPLARG field contains a CID which uniquely identifies the session to be terminated. If there is an NIB, then the

NIBCID field is checked. The NIBCID field is used only if PARSESS=YES is coded on the APPL definition statement. If the NIBCID field is not 0, it uniquely identifies the session to be terminated. (The CID for a pending active session can be obtained from the LOGON exit parameter list.) If the NIBCID field is 0, then the scope of CLSDST is all the sessions between the application program and the SLU whose name is in the NIBSYM field is specified on the ACB macroinstruction.

If the NIBCID field is 0, the scope of CLSDST is all the sessions between the application program and the network-qualified name of the SLU and the NIBNET field, if PARMS=(NQNAMES=YES) is specified on the ACB macroinstruction. If PARMS=(NQNAMES=NO) is specified on the ACB macroinstruction, or if PARMS=(NQNAMES) is not specified, all sessions to the SLU or SLUs specified in the NIBSYM are to be terminated. In other words, if this application program is in session with multiple LUs with the same name in separate networks, all sessions are terminated. However, LU 6.2 sessions maintained by VTAM LU 6.2 support (APPC=YES on the APPL definition statement) are unaffected by this macroinstruction.

CLSDST OPTCD=RELEASE

For OPTCD=RELEASE, VTAM sends an UNBIND request on each session for which BIND has been sent and rejects any outstanding CINIT requests that have been received. If this causes the SLU's session count to drop below its session limit, VTAM automatically turns the oldest queued session for the SLU, if any, into a pending active session. If there are no queued sessions or if the LU is independent, and if the SLU has a controlling application program (established with VARY LOGON or the LOGAPPL operand on the LU definition statement), which is not the application program issuing CLSDST, a session is initiated between the controlling application program and the SLU.

Sense codes on negative response to CINIT:

When VTAM processes a OPTCD=RELEASE for a pending active session (BIND not yet sent), it rejects the associated CINIT with a negative response. The application program can indicate the sense value to use with the negative response to CINIT by specifying OPTCD=SENSE and putting application-program-specified sense values into the SSENSEO, SSENSMO, and USENSEO fields of the RPL. Otherwise, VTAM rejects the CINIT with a sense code of X'08010000'. Only a nonzero sense code is allowed. If you specify OPTCD=SENSE and a sense code of X'00000000', CLSDST is rejected with (RTNCD,FDB2)=(X'14',X'50').

Session outage notification (SON) codes on UNBIND:

When the application program issues OPTCD=RELEASE to terminate an active session or pending active session (BIND sent, BIND response not received), VTAM sends an UNBIND to the SLU, terminating the session. The UNBIND type code (as specified by the SONCODE parameter) indicates the cause of the session outage. The application program can set the UNBIND SON code by specifying OPTCD=SONCODE and PARMS=(SONCODE=code) on CLSDST. The application program can also include sense information on the UNBIND by specifying PARMS=(SONCODE=X'FE') and putting the sense values into the SSENSEO, SSENSMO, and USENSEO fields of the RPL.

CLSDST OPTCD=PASS

CLSDST OPTCD=PASS is used to initiate a session between the SLU and a new PLU before terminating the sessions between the application program and the

SLU. Use of OPTCD=PASS requires authorization for the issuing application program (AUTH=PASS on the APPL definition statement).

When OPTCD=PASS is issued, the RPLAAREA field must point to the name of the PLU with which the SLU is to have a session. The RPL's AREA field can point to up to 255 bytes of user data to be sent with the session-initiation request. The RPL's AREALEN field must contain the length of the data. This user data is passed to the new PLU in the CINIT request. If the RPL points to an NIB, the LOGMODE field in the NIB specifies a logon mode name which is used to select a set of session parameters and a class of service for the new session. The USERFLD field in the NIB, if not 0, is used as a correlator for the session-initiation request. This correlator appears in the NSEXIT exit routine parameter list, if it is scheduled with a Notify request, to indicate that the initiate procedure failed or completed successfully. If the USERFLD field in the NIB is 0, then the NSEXIT routine is scheduled with a Network Services Procedure Error (NSPE) request if the procedure fails.

The name of the PLU in RPLAAREA can be either network-qualified or not, if PARMS=(NQNAMES=YES) on the ACB macroinstruction. If the name is network-qualified:

- AREALEN must be greater than 16
- RPLAAREA points to the 8-byte network identifier (padded with blanks, if necessary) followed by the 8-byte name of the LU (padded with blanks, if necessary).

OPTCD=PASS first initiates a session between the session partner in the sessions being terminated, and the new PLU. The initiate operation specifies priority queuing, so that if the SLU is at its session limit, the new session is placed at the front of the queue. Then, the sessions within the scope of CLSDST are terminated. The type of UNBIND sent by OPTCD=PASS is an UNBIND with an UNBIND SON code of BIND forthcoming. This type of UNBIND indicates that another PLU is expected to BIND a session and, therefore, the LU should not go into a mode that would reject the new session. (An example of such a mode is the IBM 3790 Communication System offline operation mode, called local mode.)

If the SLU is at its session limit when OPTCD=PASS is issued, the queued session between the new PLU and the SLU becomes a pending active session after any of the SLU's sessions are terminated, and session establishment continues. The effect for dependent LUs is that the application program determines which session the SLU has next, even if other PLUs have been waiting longer for a session with that SLU.

The CLSDST macroinstruction can specify PARMS=(THRDPTY=NOTIFY) to indicate that the application program wants to be notified when the target session is established (that is, when a positive response is received to the BIND for that session). If this parameter is specified, and the session is established, the application program receives a Notify request in its NSEXIT routine. If the session setup fails, the application program receives an NSPE or Notify request in its NSEXIT routine, regardless of the setting of this parameter. See “NSEXIT exit routine” on page 244 for more information about the Notify request.

CLSDST OPTCD=TERMQ

CLSDST OPTCD=TERMQ is used to terminate a queued session in which this application is acting as the PLU. For example, if SIMLOGON OPTCD=Q is issued and the session is queued (CINIT not yet sent), OPTCD=TERMQ could be used to terminate this session.

When OPTCD=TERMQ is used, the session partner name must be specified via the NIB (NIBSYM, and possibly, NIBNET). All sessions that are queued between the PLU APPL and the specified SLU will be terminated. VTAM will send a TERMINATE RU for this case.

REQSESS macroinstruction

The REQSESS macroinstruction initiates a session in which the application program acts as the SLU. (If the PLU is an independent LU, REQSESS cannot be used.) REQSESS sends an Initiate request to the SSCP which, in turn, sends a CINIT request to the desired PLU. If the PLU accepts the session and sends a BIND request, the application program's SCIP exit routine is scheduled with the BIND. The application program can then examine the BIND area and decide whether to establish a session. Before an application program can issue the REQSESS macroinstruction, it must have issued SETLOGON OPTCD=START, or REQSESS fails with (RTNCD,FDB2)=(X'10',X'02').

The REQSESS macroinstruction uses an RPL and an NIB to pass information. The RPL's AREA and RECLEN fields specify up to 255 bytes of user data which are passed in the Initiate request. The data is available to the PLU in the CINIT request. You must specify OPTCD=NQ in the RPL because queuing is not supported for REQSESS. The RPL's NIB field points to the NIB. You should specify LISTEND=YES in the NIB because REQSESS does not support NIB lists. The NIBSYM field contains the name of the PLU with which the application program wants a session. The LOGMODE field in the NIB contains the name of a set of session parameters suggested for use in establishing the session; the logon mode name also implies a particular class of service. This logon mode name applies to the logon mode table associated with the application program issuing the REQSESS macroinstruction. See “Establishing parameters for sessions” on page 122 for more information about the REQSESS macroinstruction in relation to session parameters.

If the USERFLD field in the NIB is not 0, it is a correlator for the Initiate procedure, and is passed to the SCIP exit routine when BIND is received. If the procedure fails, the USERFLD field in the NIB is passed to the NSEXIT routine with a Notify request. If the USERFLD field is 0, then a Network Services Procedure Error (NSPE) request is passed to the NSEXIT routine if the procedure fails. The correlator permits the application program to determine whether a BIND or Notify request is caused by a REQSESS and, if so, which REQSESS. If the application program and the PLU both support parallel sessions, multiple concurrent REQSESS requests can be outstanding for the same PLU. In that case, a unique correlator for each REQSESS is needed if the application program wants to match the BIND in the SCIP exit routine or the Notify request in the NSEXIT exit routine with the REQSESS that caused it. This is because the PLU and the SLU names passed to the SCIP and NSEXIT exit routines do not uniquely identify the session-initiation request.

In order for a nonzero user field to be used, both the PLU and its SSCP must support the user-request correlation facility; if they do not, REQSESS is rejected with (RTNCD,FDB2)=(X'10',X'12') and sense code=X'10030000'. REQSESS can be reissued if the NIB's USERFLD is zero; however, this means that an NSPE rather than a Notify request is received if an error occurs, and that the user correlator passed in the SCIP exit routine (if the BIND request is received) is zero.

OPNSEC macroinstruction

The OPNSEC macroinstruction is issued by a VTAM application program that has received a BIND request from a PLU and wishes to respond positively to the BIND and establish the session. The application program is the SLU in the session. The application program receives the BIND in its SCIP exit routine. See "Exit routines related to session establishment and termination" on page 100 for more information about the SCIP exit routine.

OPNSEC uses the RPL and the NIB to identify the BIND for the session to be established, and to specify options for the session. The RPL's NIB field points to the NIB to be used. If the application program supports parallel sessions (PARSESS=YES is coded on the APPL definition statement) and the NIBCID field is not zero, the CID uniquely identifies the BIND to be accepted. (The CID can be obtained from the SCIP exit parameter list.) If the NIBCID field is zero, or if the application program does not support parallel sessions, the NIBSYM field must contain the name of the PLU from which a BIND has been received and that is to be accepted. If more than one BIND has been received from the named PLU, the oldest is accepted.

The name supplied can be network-qualified if PARMS=(NQNAMES=YES) is specified on the ACB macroinstruction. If a BIND cannot be found to accept, the OPNSEC is rejected with (RTNCD,FDB2)=(X'10',X'00').

It is possible for the pending active session created by the BIND to end before the application program issues the OPNSEC macroinstruction. For example, if the PLU is a VTAM application program, it might issue OPNDST, which would cause the BIND to be sent, and then issue CLSDST before the OPNDST completed. This would cause an UNBIND to be sent, which could purge the BIND. If this happens, OPNSEC is posted with (RTNCD,FDB2)=(X'10',X'12'), and sense code=X'08060000' in the SSENSEI, SSENSMI, and USENSEI fields. This is not an error on the part of the application program, but is a situation the application program should be prepared to handle.

The PLU name in the BIND request presented to this application program can be the network name of the PLU or an uninterpreted name as follows:

- If the PLU is a VTAM application program whose APPL definition statement includes either a network name or an ACBNAME name, and if this program issues CLSDST OPTCD=PASS to initiate the session, the network name is used in the BIND.
- If SIMLOGON, OPNDST OPTCD=ACQUIRE, or automatic logon is used to initiate the session, the network name is used in the BIND.
- If the SLU initiated the session (for example, by issuing REQSESS, or LOGON), the uninterpreted name can be used in the BIND. The same name that is used in the original session-initiation request from the SLU is returned in the BIND.

When the BIND flows in the network, the name fields (NS_PLU and NS_SLU) can be network-qualified. As the BIND is presented to the application, this name, if it were network-qualified, is changed from network-qualified to an 8-byte name. The 8-byte name can differ from the LU portion of the network-qualified name. If it does, name translation has occurred. For details of the names available to the application, see Table 34 on page 240 and Table 45 on page 262.

Use of the network name, as opposed to the ACBNAME, in the BIND permits the SLU to unambiguously relate a BIND to a particular PLU. The network name can

also be used by the SLU to reinitiate a session (for example, after a session outage), even if the original session is initiated by the PLU or a third party.

The NIB used for OPNSEC specifies options regarding the processing of the session, such as whether a RESP exit routine should be scheduled when a response is received. The USERFLD in the NIB contains a 4-byte value to be associated with the session. This value is returned to the application program in the USER field of the RPL used for a SEND, RECEIVE, RESETSR, or SESSIONC macroinstruction, when the associated communication operation is complete. This value might be a pointer to a user control block representing the session.

When OPNSEC successfully completes, VTAM indicates in the NIB the type of encryption (selective or required) that is specified by the application program. Refer to "Node initialization block (NIB)" on page 115 for more information about the contents of the NIB.

BIND response

The BIND request from the PLU includes session parameters that specify the protocols to be used on the session. The BIND can be negotiable or non-negotiable. If it is non-negotiable, the SLU application program must accept the BIND as it is, or reject it. It cannot modify the session parameters. If the BIND is negotiable, the application program can modify the session parameters and return them to the PLU in a negotiable BIND response. To send a negotiable BIND response, the application program must issue OPNSEC with PROC=NEGBIND specified in the NIB, with the BNDAREA field in the NIB pointing to the session parameters and user data to send to the PLU. The ISTDBIND DSECT can be used to map this area. If no changes to the BIND are required, then PROC=NNEGBIND, or BNDAREA=0 can be specified in the NIB to send a non-negotiable response to the BIND, indicating to the PLU that the original BIND session parameters are acceptable. The application program cannot send a negotiable BIND response to a non-negotiable BIND request. If PROC=NEGBIND is specified on the OPNSEC macroinstruction for a non-negotiable BIND request, the OPNSEC macroinstruction is rejected with (RTNCD,FDB2)=(X'14',X'74'); a BIND request rejected response with sense code=X'08010000' is sent to the PLU. See Table 7 for a description of OPNSEC macroinstruction PROC options used to send the BIND response. See Appendix F, "Specifying a session parameter," on page 793, to determine which session parameters can be modified in a negotiable BIND response.

Table 7. OPNSEC macroinstruction PROC options used to send BIND response

BIND received	OPNSEC PROC option set by SLU application program	Action taken
Non-negotiable	NNEGBIND	Valid. A non-negotiable BIND response is sent.
Non-negotiable	NEGBIND	Not valid. OPNSEC is rejected with (RTNCD,FDB2)=(X'14',X'74'). The BIND is rejected with sense code=X'08010000'.
Negotiable	NNEGBIND	A non-negotiable response is sent. BNDAREA is ignored. Values from the BIND request are used for the session.
Negotiable	NEGBIND	Negotiable response is sent. New parameters in the BNDAREA are used if BNDAREA is supplied. If BNDAREA is not supplied, a non-negotiable response is sent. If session parameters that are not valid are supplied, OPNSEC is rejected with (RTNCD,FDB2)=(X'14',X'75'), and a request rejected response is sent to the BIND (sense code=X'08010000').

SESSIONC macroinstruction with CONTROL=BIND

The SESSIONC macroinstruction with CONTROL=BIND is used to reject a BIND request from a PLU if the application program does not wish to establish the session. For example, if the BIND request is non-negotiable and the session parameters are unacceptable, the application program would have to reject the BIND.

The SESSIONC macroinstruction uses the RPL and, optionally, an NIB to identify the BIND to be rejected. If the RPL contains a CID, the CID uniquely identifies the BIND to be rejected. (The CID can be obtained from the SCIP exit routine parameter list when the SCIP exit is scheduled with the BIND request.) If the RPL points to an NIB and the application program is capable of parallel sessions (PARSESS=YES is coded on the APPL definition statement) and the NIBCID field is not zero, then the CID identifies the BIND to be rejected. Otherwise, the NIBSYM field contains the name of the PLU from which the BIND came.

The name supplied can be network-qualified by using NIBNET if PARMS=(NQNAMES=YES) is specified on the ACB macroinstruction. If there is more than one BIND outstanding from the named PLU, the oldest is rejected.

If the queued BIND to be rejected cannot be found (for example, if the PLU sent UNBIND that purged the BIND), SESSIONC is posted with (RTNCD,FDB2)=(X'14',X'13') if a CID is specified, or with (RTNCD,FDB2)=(X'14',X'60') if only a name is specified.

The PLU name in the BIND request presented to the application program can be either the network name of the PLU or an uninterpreted name as follows:

- If the PLU is a VTAM application program whose APPL definition statement includes either a network name or an ACBNAME name, and if this program issues CLSDST OPTCD=PASS to initiate the session, the network name is used in the BIND.
- If SIMLOGON, OPNDST OPTCD=ACQUIRE, or automatic logon is used to initiate the session, the network name is used in the BIND.
- If the SLU initiated the session (for example, by issuing REQSESS, INIT-SELF, or LOGON), the uninterpreted name is used in the BIND. The same name that is used in the original session-initiation request from the SLU is returned in the BIND.

When the BIND flows in the network, the name fields (NS_PLU and NS_SLU) can be network-qualified. As the BIND is presented to the application, this name, if it were network qualified, is changed from network-qualified to an 8-byte name. The 8-byte name can differ from the LU portion of the network-qualified name. If it does, name translation has occurred. For details of the names available to the application, see Table 34 on page 240 and Table 45 on page 262.

Use of the network name, as opposed to the ACBNAME, in the BIND permits the SLU to unambiguously relate a BIND to a particular PLU. The network name can also be used by the SLU to reinitiate a session (for example, after a session outage), even if the original session is initiated by the PLU or a third party.

The SSENSEO, SSENSMO, and USENSEO fields of the RPL specify the sense data to be returned in the BIND response when the request is rejected.

TERMSESS macroinstruction

The TERMSESS macroinstruction terminates sessions in which the application program acts as the SLU.

The RPL OPTCD operand specifies the type of session termination desired by the application program:

- OPTCD=COND causes a Terminate Orderly request to be sent to the SSCP.
- OPTCD=UNCOND causes a Terminate Forced request to be sent to the SSCP.
- OPTCD=UNBIND causes an UNBIND to be sent to the PLU.
- OPTCD=TERMQ causes a Terminate Forced request to be sent to the SSCP for a queued or pending active session.

When it receives an UNBIND in its SCIP exit routine or a CLEANUP in its NSEXIT exit routine, VTAM notifies the application program that the session ends. Otherwise, if you specify OPTCD=UNBIND, VTAM directly ends the session by sending an UNBIND to the PLU.

The scope of a TERMSESS is the set of sessions terminated. The TERMSESS macroinstruction uses the RPL, and optionally, an NIB to identify the scope of the TERMSESS. If no NIB is supplied, then the RPLARG field contains a CID which uniquely identifies the session to be terminated. If there is an NIB, then the NIBCID field is checked. The NIBCID field is used only if PARSESS=YES is coded on the APPL definition statement. If the NIBCID field is not 0, it uniquely identifies the session to be terminated. If the NIBCID field is 0, the scope of the TERMSESS is all the sessions in which the application program issuing TERMSESS is the SLU, and the network-qualified name of the PLU named in NIBSYM and the NIBNET field if PARMS=(NQNAMES=YES) is specified on the ACB macroinstruction. If PARMS=(NQNAMES=NO) is specified on the ACB macroinstruction or if PARMS=(NQNAMES) is not specified, all sessions to the PLU or PLUs specified in NIBSYM are to be terminated. In other words, if this application program is in session with multiple LUs with the same name in separate networks, all sessions are terminated.

Note:

1. OPTCD=COND has no effect if the PLU is an independent LU.
2. Under some network outage conditions, the SSCP cannot cause the sending of a CTERM Forced to the PLU; under these conditions, the SSCP treats the Terminate Forced as if it had been Terminate Cleanup. See “Stages of session termination” on page 84 for a description of the actions that result from these types of terminations.
3. LU 6.2 sessions maintained by VTAM LU 6.2 support (APPC=YES on the APPL definition statement) are unaffected by this macroinstruction.

TERMSESS OPTCD=UNBIND

TERMSESS OPTCD=UNBIND causes an UNBIND to flow to the PLU. This UNBIND can contain sense information and an UNBIND SON (UNBIND type) code specified by the SLU. When the SLU application program issues TERMSESS OPTCD=UNBIND to terminate an active session or pending active session (BIND received, BIND response not sent), VTAM sends an UNBIND to the PLU identified by the CID in the RPLARG or NIBCID field. The application program can set the UNBIND SON code by specifying OPTCD=SONCODE and PARMS=(SONCODE=*code*) in the RPL. The application program can also include

sense information on the UNBIND by specifying PARMS=(SONCODE=X'FE') and putting application-specified sense values into the SSENSEO, SSENSMO, and USENSEO fields of the RPL.

VTAM performs the same functions for ending sessions as it does when the PLU sends UNBIND. That is, session services are informed of session-ended, or UNBIND failed, and so forth. When VTAM receives the UNBIND response from the PLU (whether positive or negative), it completes the TERMSESS processing and posts the request complete.

Refer to "TERMSESS—Request session termination, application program is SLU" on page 560 for details on how to specify sense code fields.

TERMSESS OPTCD=TERMQ

TERMSESS OPTCD=TERMQ is used to terminate a queued session or pending active session in which the application program is acting as the SLU. For example, if REQSESS has been issued and completed successfully, but the subsequent BIND has not been received, OPTCD=TERMQ could be used to terminate this session. When this OPTCD code is used a name must be provided via the NIB. All sessions that match this name pair and are queued or pending active will be terminated. Active sessions are not affected.

Note:

1. The SLU application program that issues TERMSESS OPTCD=UNBIND must process the RTNCD, FDB2, and possibly sense information in the RPL to determine if it can discard its representation of the session, or perform some other process (retry the TERMSESS, ABEND, or issues other appropriate macroinstruction).
2. If the SLU application program issues TERMSESS in lieu of SESSIONC to reject a pending active session (BIND received, BIND response not sent), then OPTCD=UNBIND must be used.
3. If you are using OPTCD=UNBIND with the TERMSESS macroinstruction, you must specify the CID address in either the ARG or NIB operand. The NIBCID field is only used if PARSESS=YES is coded on the APPL definition statement. If you use the NIB operand, NAME cannot be used.
4. The PLU can examine the SON code and the SENSE information on the UNBIND only if it has an SCIP exit and if SONSCIP=YES is specified on its APPL definition statement.
5. The SCIP exit routine is strongly recommended.

Exit routines related to session establishment and termination

Several exit routines notify a VTAM application program of requests that affect sessions. This section describes the following four exit routines that are designed for common use and are included in an application program:

- When a CINIT request is received, VTAM schedules the LOGON exit routine to request that the application program establish a session in which it acts as the PLU.
- When session-control requests, including BIND and UNBIND requests, are received, VTAM schedules the SCIP exit routine.
- When certain network-services requests are received from the SSCP, VTAM schedules the NSEXIT routine. These requests can indicate the state of a session or of a session-initiation or termination procedure.

- When a CTERM (or similar) request is received, VTAM schedules the LOSTERM exit routine to request termination of a session in which the application program acts as the PLU.

The following sections describe the purpose of these exit routines and the information available to them. For further information about how some of these exit routines are used for session outage notification, see "Session outage notification" on page 110. For a summary of exit routines involved in session outage or disruptions, see Table 10 on page 108. For a description of the interfaces to the exit routines and of other general considerations that apply to exit routines, as well as alternative exit routines, see Chapter 7, "Using exit routines," on page 219.

LOGON exit routine

VTAM schedules the LOGON exit routine to inform the application program that it should establish a session in which it acts as the PLU. VTAM schedules the LOGON exit when a CINIT request is received unless:

- The ACB is opened with MACRF=NLOGON.
- The ACB is opened with MACRF=LOGON, but SETLOGON OPTCD=START has not been issued (CINIT is queued).
- The CINIT results from the application program issuing OPNDST OPTCD=ACQUIRE.
- The application program has a matching outstanding OPNDST OPTCD=ACCEPT to accept that CINIT.
- A SETLOGON OPTCD=HOLD request holds the exit (CINIT is queued).

If a CINIT has been queued, the LOGON exit routine is scheduled when SETLOGON OPTCD=START is issued.

The LOGON exit parameter list provides the name of the SLU and the CID of the established session. These items can be used to identify the CINIT request to be accepted with OPNDST OPTCD=ACCEPT or rejected with CLSDST.

The exit parameter list provides the user-request correlator. Unless the application program initiates the session with the SIMLOGON macroinstruction, this field is 0. If the application program uses the SIMLOGON macroinstruction, the field contains the contents of the NIB's USERFLD field at the time SIMLOGON is issued. The field also provides a means of determining whether SIMLOGON caused the CINIT request and, if so, which one. A read-only RPL is also provided to the exit routine. The AREA field in this RPL points to a read-only copy of the CINIT request. The RPL RECLEN field contains the length of the CINIT request. Refer to *SNA Formats* for a description of the CINIT request.

Because different network components manage the network addresses and the addresses can vary, the application program should ignore any network addresses that appear in the CINIT request unit.

Certain information can appear in control vectors appended to the CINIT RU.

Vector	Description
X'0D'	Class-of-service and virtual route list
X'0E'	Network-name control vector or network-qualified name of the PLU if present

X'0E' Network-name control vector or network-qualified name of the SLU if present

X'15' Network-qualified address pair control vector

X'2C' COS and TPF control vector

X'2D' Mode-name control vector

X'2F' Model-terminal-information control vector

X'59' Session authorization data control vector

X'5F' Extended fully qualified PCID control vector

X'60' Fully qualified PCID control vector

X'64' TCP information control vector

X'66' Data compression control vector

For a detailed description of these control vectors, refer to *SNA Formats*.

The LOGON exit routine can include a procedure to obtain a copy of the information in the CINIT request. This information can be interrogated for session parameters and also for the logon mode name and the class-of-service (COS) name. For example, the logon mode name must be saved by the application program for use with a subsequent CLSDST OPTCD=PASS macroinstruction if the same COS is to be used in the second session. A procedure for obtaining the session parameters is demonstrated in Example 1 of "Examples of how an application program processes session parameters" on page 127. Example 2 of the same section demonstrates a procedure for obtaining the logon mode name.

SCIP exit routine

The SCIP exit routine is scheduled whenever a session-control request is received for the application program. In this section, we shall consider only BIND, which is a request to establish a session, and UNBIND, which is a request to terminate a session. Other session-control requests are discussed in "SCIP exit routine" on page 256.

Receiving a BIND request

When a BIND request is received, VTAM schedules the SCIP exit routine unless the SCIP exit routine is being held by SETLOGON OPTCD=HOLD. If SCIP exits are being held, an exit for the BIND request is scheduled after the next SETLOGON OPTCD=START.

The exit parameter list provides the CID of the session to be established. This CID can be used to identify the BIND that is to be accepted by OPNSEC or rejected by SESSIONC CONTROL=BIND. The BIND request that is passed to the SCIP exit routine contains the PLU name. For information about uninterpreted PLU names, see "OPNDST macroinstruction" on page 89 and "SESSIONC macroinstruction with CONTROL=BIND" on page 98.

The exit parameter list also provides a user-request correlator. If the application program initiates the session with the REQSESS macroinstruction, the correlator field has the same value as the USERFLD field in the NIB at the time REQSESS is

issued. Otherwise, the field is 0. The field provides a means for the application program to determine if the BIND is caused by a REQSESS macroinstruction and if so, which one.

A read-only RPL is also provided to the exit routine. The AREA field in this RPL points to a read-only copy of the BIND request. The RPL's RECLEN field contains the length of the request. For a description of the BIND request unit, refer to *SNA Formats*. BIND session parameters are also discussed in Appendix F, "Specifying a session parameter," on page 793.

Certain information can appear in control vectors appended to the BIND RU.

Vector	Description
X'0E'	Network-name control vector or network-qualified name of the PLU
X'0E'	Network-name control vector or network-qualified name of the SLU
X'27'	XRF session-activation control vector
X'2B'	Route selection-control vector
X'2C'	COS and TPF control vector
X'2D'	Mode-name control vector
X'5F'	Extended fully-qualified PCID control vector
X'60'	Fully-qualified PCID control vector
X'66'	Data compression control vector

Receiving an UNBIND request

When an UNBIND request is received by an SLU application program or is received by a PLU application program for which SONSCIP=YES is coded on the APPL definition statement, the SCIP exit routine is scheduled. The CID of the terminated session is provided in the exit parameter list. Because VTAM automatically sends a positive response to the UNBIND, the application program needs only to clean up its representation of the session; no CLSDST or TERMSESS should be issued.

The exit parameter list provides the USERFLD field specified in the NIB when the session is established by OPNDST or OPNSEC. The USERFLD can be used by the application program to identify the session. A read-only RPL is also provided to the exit routine. The AREA field in the RPL points to a read-only copy of the UNBIND request. The RPL's RECLEN field contains the length of the request. The UNBIND SON code in the request indicates the reason for the session termination and can be used by the application program to determine what recovery action to take. UNBIND parameters are discussed in the section titled "Session outage notification (SON) codes on UNBIND" on page 93.

An SLU application program must have an SCIP exit routine. If a PLU application program does not have an SCIP exit routine or if SONSCIP=NO is coded on the APPL definition statement, the NSEXIT exit routine (if it exists) is scheduled with CLEANUP when UNBIND is received by the PLU application program; if the NSEXIT exit routine does not exist either, the LOSTERM exit routine is scheduled. See Table 8 on page 105 for information on exit routine scheduling.

Certain information can appear in control vectors appended to the UNBIND RU.

Vector	Description

X'35' Extended-sense data control vector

X'60' Fully-qualified PCID control vector

NSEXIT exit routine

The NSEXIT exit routine is scheduled when certain network services requests are received from the SSCP. The requests that can currently be received are: Clean Up Session, Network Services Procedure Error, and Notify.

The NSEXIT parameter list includes a pointer to a read-only RPL. The RPL's AREA field points to a read-only copy of the request that is received. The RPL's RECLEN field contains the length of the request. For a description of the request unit formats, refer to "NSEXIT exit routine" on page 244.

CLEANUP request

The Clean Up Session (CLEANUP) request is sent from the SSCP when a session has been terminated (perhaps because of a network outage). Any outstanding RPLs will still be completed by VTAM. The parameter list includes the CID for the session and the USERFLD field from the NIB used to establish the session. A CLSDST or TERMSESS macroinstruction should not be used after receipt of CLEANUP for the session because the session is gone; if issued, they fail.

CLEANUP can be received by both the PLU and the SLU. If the CLEANUP request is received by a PLU application program and the application program does not have an NSEXIT exit routine, the LOSTERM exit routine (if it exists) is scheduled. See Table 8 on page 105 for information about session outage notification. Application programs that intend to act as SLUs in any sessions must have an NSEXIT exit routine, or they might not be notified of the loss of those sessions.

If UNBIND is received by a PLU application program which does not have an SCIP exit routine or for which SONSCIP=NO is coded on the APPL definition statement, the NSEXIT exit routine (if it exists) is scheduled with CLEANUP. See Table 8 on page 105 for information about session outage notification.

There are several instances where a lost session is reported to the application in the LOSTERM Exit with a reason code of 12 or 20. In the past, if the LOSTERM Exit had already been scheduled with a reason code of 32, this more recent indication of 12 or 32 would be discarded.

If the LOSTERM Exit has already been scheduled with a reason code of 32, this more recent indication of 12 or 20 is promoted to a CLEANUP. Therefore, the NSEXIT, if available, is scheduled with a CLEANUP RU. If the NSEXIT is not available, the indication is discarded and whatever caused the attempt to notify the LOSTERM Exit and then the NSEXIT is discarded.

Table 8. Session outage notification summary

Application program is	Receives	Exit routine scheduling priority (read left to right)		
		SCIP	**NSEXIT**	**LOSTERM**
Primary Logical Unit	CTERM Orderly	Not applicable	Not applicable	Reason code 32
	CTERM Forced	Not applicable	Not applicable	Reason code 12 or 20
	CLEANUP	Not applicable	CLEANUP	Reason code 12 or 24, followed by 16 or, if APPL definition statement includes LOSTERM=SECOND | IMMED, reason code 48
	UNBIND	UNBIND	CLEANUP	Reason code 12 or 24, followed by 16
	Notify or NSPE	Not applicable	Notify or NSPE	No exit routine scheduled
Secondary Logical Unit	CLEANUP	Not applicable	CLEANUP	No exit routine scheduled
	UNBIND	UNBIND	Not applicable	Not applicable
	Notify or NSPE	Not applicable	Notify or NSPE	No exit routine scheduled

Note:

1. The SCIP exit routine (if available) schedules for a PLU (for UNBIND) only if you code SONSCIP=YES on the APPL definition statement. For SONSCIP=NO, UNBIND is handled as if the SCIP exit routine were not available. (We strongly recommend the SCIP exit routine for support of independent LUs.) An application program acting as an SLU must have an SCIP exit routine.
2. If the LOSTERM exit routine is not available, and is the last choice in the scheduling priority, an exit routine is not scheduled.

NSPE request

The Network Services Procedure Error (NSPE) request informs the application program of an error in an Initiate or Terminate procedure started by the application program. This can happen after a successful SIMLOGON, REQSESS, TERMSESS, or CLSDST OPTCD=PASS macroinstruction executes. These macroinstructions post complete after the Initiate or Terminate procedure successfully starts; however, the procedure can fail later. The NSPE request contains the names of the LUs involved in the session-initiation or termination request. If the USERFLD field in the NIB used for the macroinstruction is not 0, then NSPE is not sent. In that case, the USERFLD field is a correlator for the procedure, and a Notify request is sent instead.

NSPE can also occur for OPNDST OPTCD=ACQUIRE. (Notify is not possible.) NSPE occurs if a session terminates after the SSCP responds to the Initiate from OPNDST OPTCD=ACQUIRE, but before it is notified by the PLU that the session is established. In this case, the OPNDST operation is normally posted complete with error feedback information. In some rare outage situations, however, OPNDST can be posted complete with (RTNCD,FDB2)=(X'00',X'00'), and NSPE can be received as well as session outage notification as described in "Session outage notification" on page 110. The receipt of NSPE can occur before or after the

OPNDST posting. Because either OPNDST is posted complete with error feedback information or session outage notification occurs, the NSPE request can be ignored.

Notify request

The Notify request using control vector hex 3 informs the application program of the completion of an Initiate procedure started by the application program through a SIMLOGON, REQSESS, or CLSDST OPTCD=PASS macroinstruction. When a third application program requires notification of the Initiate procedure being complete, use the CLSDST OPTCD=PASS macroinstruction and specify PARMS=(THRDPTY=NOTIFY). The Notify request, rather than NSPE, is also used for procedure errors if the USERFLD field in the NIB for the macroinstruction is not 0. In that case, the USERFLD is a correlator for the procedure that enables the application program to determine which procedure failed. The exit parameter list contains this 4-byte correlator.

LOSTERM exit routine

The LOSTERM exit routine is scheduled to inform the application program that one of the following events has occurred for a session with this application program:

- Request/response units have been discarded because the application program would not accept them and there is no buffer space to queue them in VTAM; session data recovery or session termination is required. This is the only reason that LOSTERM can be scheduled for an SLU application program.
- A CTERM Conditional has been received by a PLU application program, perhaps because the other end of the session issued TERMSESS OPTCD=COND; the application program determines the appropriate response or action.
- A CTERM Forced has been received by a PLU application program whose LOSTERM Exit has not already been scheduled for a CTERM conditional, perhaps because the other end of the session issued TERMSESS OPTCD=UNCOND; CLSDST is required for this session.
- A Test Request message has been received from a BSC 3270 terminal; CLSDST is required for this session.
- A session outage has occurred that normally would have been reported through the NSEXIT or SCIP exit routines, but these exit routines are not available. See Table 8 on page 105 for a summary of session outage notification. CLSDST might be required for this session.
- A request to suspend or resume a synchronous cross-memory SRB-mode macroinstruction request has been unsuccessful. The LOSTERM exit is used to notify the application program because normal request completion notification cannot be performed.

See “Stages of session termination” on page 84 for further information about these exits.

The CID of the affected session is provided in the LOSTERM exit routine parameter list. The contents of the USERFLD field from the NIB used when the session is established are also provided. The exit parameter list contains a code indicating the reason the LOSTERM exit routine is scheduled.

Summary tables of exit routines

The following tables summarize the exit routines that are discussed in the preceding sections.

Table 9. Summary of exit routines involved in session initiation

Action or event causing session-initiation request	Exit routine [1] for event that occurs...	
	...in application program named PLUAPPL	**...in application program named SLUAPPL**
LU sends Initiate request or character-coded logon requesting session with PLUAPPL.	LOGON exit routine is scheduled with CINIT.	(Not applicable)
SLUAPPL issues REQSESS requesting session with PLUAPPL.	LOGON exit routine is scheduled with CINIT.	If PLUAPPL accepts CINIT by issuing OPNDST OPTCD=ACCEPT, BIND is received in SCIP exit routine. If, after SSCP responds positively to Initiate resulting from REQSESS, the session cannot be established (for example, PLUAPPL rejects the CINIT), NSEXIT exit routine is scheduled with NSPE or Notify RU.
PLUAPPL issues OPNDST OPTCD=ACCEPT for session with SLUAPPL.	(Not applicable)	BIND received in SCIP exit routine.
PLUAPPL issues OPNDST OPTCD=ACQUIRE for session with SLUAPPL.	(Not applicable)	BIND received in SCIP exit routine.
PLUAPPL issues SIMLOGON for session with SLUAPPL.	LOGON exit routine is scheduled with CINIT. If, after the SSCP responds positively to Initiate resulting from SIMLOGON, the session cannot be established, NSEXIT exit routine is scheduled with NSPE or Notify RU.	If PLUAPPL accepts CINIT by issuing OPNDST OPTCD=ACCEPT, BIND is received in SCIP exit routine.
PLUAPPL is the controlling application program for a device-type LU (LOGAPPL=PLUAPPL in the LU's definition statement), and another PLU releases the LU for which there are dependent queued sessions.	LOGON exit routine is scheduled with CINIT.	(Not applicable)
VTAM operator issues VARY LOGON command to make PLUAPPL the controlling application program for a dependent LU and the LU is now available.	LOGON exit routine is scheduled with CINIT.	(Not applicable)
Application program named PASSER issues CLSDST OPTCD=PASS to pass SLUAPPL to PLUAPPL.	LOGON exit routine is scheduled with CINIT. (If the requested session cannot be established, PASSER's exit routine is scheduled with an NSPE or Notify RU.)	The SCIP exit routine is entered twice; once by the UNBIND received from PASSER, again by the BIND received from PLUAPPL.
Application program named GETTER issues SIMLOGON OPTCD=RELRQ requesting PLUAPPL to release an LU with which it is currently in session.	RELREQ exit routine is scheduled.	(Not applicable)

Table 9. Summary of exit routines involved in session initiation (continued)

Action or event causing session-initiation request	Exit routine [1] for event that occurs...	
	...in application program named PLUAPPL	**...in application program named SLUAPPL**

Note:

1. If the program does not have the exit routine, no notification occurs, or the specified function is not supported.

Table 10. Summary of exit routines involved in session outages or disruption

Action or event causing session outage [1] or session disruption	Method of notification	
	...for application program acting as PLU [2]	**...for application program acting as SLU [2]**
Session outage occurs (for example, because of a link failure, a virtual route deactivation, an RU received exceeding maximum RU size, an NCP failure, or a switched line disconnection).	If SONSCIP=YES is coded on the PLU application program's APPL definition statement, UNBIND is received in the SCIP exit routine. If SONSCIP=NO and the application program has an NSEXIT exit routine, that exit routine is scheduled with CLEANUP. Otherwise, the LOSTERM exit routine is scheduled.	SCIP exit routine is scheduled with UNBIND or, depending on the type of outage, NSEXIT exit routine can be scheduled with CLEANUP.
VTAM operator issues a VARY NET,INACT,I (immediate) command to deactivate an LU in the same domain or to deactivate a resource representing an LU in another domain.	LOSTERM exit routine is scheduled.	UNBIND is received in the SCIP exit routine. If the normal communication paths are not available, NSEXIT exit routine is scheduled with CLEANUP.
VTAM operator issues a VARY NET,TERM,TYPE=UNCOND command to terminate one or more sessions.	LOSTERM exit routine is scheduled.	UNBIND is received in the SCIP exit routine. If normal communication paths are not available, NSEXIT exit routine is scheduled with CLEANUP.
VTAM operator issues a VARY NET,INACT,F or R command to deactivate an LU in the same domain or to deactivate a resource representing an LU in another domain.	If the application program has an NSEXIT exit routine, that exit routine is scheduled with CLEANUP. Otherwise, the LOSTERM exit routine is scheduled. [3]	NSEXIT exit routine is scheduled with CLEANUP. For some race (contention) conditions, SCIP exit routine can be scheduled with UNBIND instead.
VTAM operator issues a VARY NET,TERM,TYPE=FORCE command to terminate one or more sessions.	If the application program has an NSEXIT exit routine, that exit routine is scheduled with CLEANUP. Otherwise, the LOSTERM exit routine is scheduled.	NSEXIT exit routine is scheduled with CLEANUP. For some race (contention) conditions, SCIP exit routine can be scheduled with UNBIND instead.
VTAM operator issues a VARY NET,TERM,TYPE=COND command to terminate one or more sessions.	LOSTERM exit routine is scheduled.	UNBIND is received in SCIP exit routine if PLU terminates session.
Session-type error detected (RUs discarded because of lack of buffer space).	LOSTERM exit routine is scheduled.	LOSTERM exit routine is scheduled.

Table 10. Summary of exit routines involved in session outages or disruption (continued)

Action or event causing session outage [1] or session disruption	Method of notification	
	...for application program acting as PLU [2]	**...for application program acting as SLU [2]**
One of the following situations occurs: • VTAM is being terminated. • VTAM recognizes an internal error. • VTAM receives a HALT command or VARY NET,INACT command for the application program from the operator. • An alternate application has issued an OPEN ACB to take over sessions from an application program that has enabled persistence.	TPEND exit routine is scheduled.	TPEND exit routine is scheduled.

Note:

1. A session outage is any action or event that causes a path between a PLU and an SLU to break or that causes loss of one of the participants in the session.
2. If the program does not have an applicable exit routine, notification does not occur. Certain error conditions are indicated only through a LOSTERM exit routine.
3. Specify either exit routine, or both. If you do not specify either exit routine, the VARY INACT command hangs.

Table 11. Summary of exit routines involved in session termination by session participant

Action or event causing session termination	Method of notification	
	...for application program acting as PLU [1]	**...for application program acting as SLU [1]**
Device-type LU requests session termination (forced or orderly).	LOSTERM exit routine is scheduled. [2]	(Not applicable)
Application program issues TERMSESS OPTCD=UNBIND.	SCIP exit routine with UNBIND. [2] [3]	(Not applicable)
Application program issues TERMSESS OPTCD=COND or UCOND.	LOSTERM exit routine is scheduled.	UNBIND received in SCIP exit routine.
Application program issues CLSDST.	(Not applicable)	UNBIND received in SCIP exit routine.
Application program acting as SLU issues CLOSE or device-type LU requests session termination (CLEANUP).	SCIP exit routine with UNBIND, NSEXIT exit routine with CLEANUP, or LOSTERM exit routine is scheduled.	(Not applicable)
Application program acting as PLU issues CLOSE.	(Not applicable)	SCIP exit routine with UNBIND or NSEXIT exit routine with CLEANUP is scheduled.

Note:

1. If the program does not have the required exit routine, notification does not occur.
2. For forced termination, if the normal communication paths are unavailable, the SCIP exit routine with UNBIND or NSEXIT exit routine with CLEANUP is scheduled.
3. For the PLU's SCIP to be driven with an UNBIND, you must code SONSCIP=YES on the APPL definition statement.

Session outage notification

"Stages of session termination" on page 84 gives an overview of certain events that can lead to session termination. Usually, an application program includes SCIP, NSEXIT, and LOSTERM exit routines. These routines are scheduled to notify the application program of various types of session outages and related events. "Exit routines related to session establishment and termination" on page 100 describes these exit routines. Table 12 on page 111 provides a more detailed list of session outage notification (SON) reasons.

If the relevant exit routines are not coded in the application program, the program might not be informed of certain key events in a timely manner (if at all). Thus, the exit routines should be coded.

Note: Normally, when an outage occurs for a session, outstanding RPL-based operations for the session are posted complete with appropriate feedback information; this is not always possible if the outage is because of a VTAM abnormal termination. For a given session outage, the order in which the RPL exit routine and the SCIP, NSEXIT, or LOSTERM exit routines are scheduled is unpredictable.

The exit routine chosen depends upon:

- The type of session outage event
- Whether the application program is the PLU or the SLU in the session
- The availability of the various exit routines

Note: An exit routine is available if it is specified in an EXLST for the application program. In some cases, if the exit routine normally scheduled for an event is not available, an alternate exit routine schedules. See Table 8 on page 105 for information about session outage notification and Table 12 on page 111 for a more detailed list of session outage notification reason codes. For information on exit routines and their availability, see "SCIP exit routine" on page 102, "NSEXIT exit routine" on page 104, and "LOSTERM exit routine" on page 106.

It is possible for one session outage to generate multiple session outage notification signals to an LU (along different paths through the network) for the same disrupted session. For example, CLEANUP and UNBIND can both be sent to an LU.

Queuing a request for a session with an SLU

When a session is requested with a particular SLU, that LU might not be available to act as the SLU in the requested session. This might be because the LU is either at its session limit, or is not currently enabled to act as an SLU.

Table 6 on page 82 shows how an SLU application program can change from the disabled state to the enabled state by using SETLOGON OPTCD=START. Some terminals also have the ability to switch back and forth between enabled and disabled states. For example, for certain IBM 3274 control units, if a printer or display is powered off, the control unit notifies the SSCP that the terminal is disabled for sessions. If the terminal is then powered on, the control unit notifies the SSCP that the terminal is enabled for sessions. Thus, the SSCP is able to keep track of the session capability of these LUs.

If an application program requests a session with such an LU by using SIMLOGON OPTCD=Q and the LU is available, the session is established at that time. However, if the LU is currently disabled for sessions as an SLU, or if the LU is at its session limit, then SIMLOGON is posted complete, but the session is not established until sometime after the LU becomes available (for example, by powering on).

The LU can change from an enabled to a disabled state (for example, by powering off) just before the BIND reaches the LU. In this case, VTAM rejects the BIND request. The SNA sense code in the response can be used to determine the appropriate recovery procedure. This sense code is made available in the SSENSEI and SSENSMI fields of the RPL that issued OPNDST OPTCD=ACCEPT.

Sense code X'08450000' indicates that the LU notifies the SSCP when it is again enabled. For this sense code, SIMLOGON OPTCD=Q can be reissued immediately and the SSCP queues the request for the session until the LU becomes available.

Sense code X'080A0000' indicates that the LU does not notify the SSCP when it becomes available for a session. This code is used if either the LU or its SSCP does not support the required notification facility. In this case, the application program should not immediately reissue SIMLOGON OPTCD=Q because another BIND rejection results. Manual intervention is required. For example, when the terminal operator powers on the terminal, the operator can notify the host operator, who in turn can send a message to the program to issue SIMLOGON at that time.

Table 12. Session outage notification UNBIND type codes and reason codes

Session outage notification signals

SCIP UNBIND RU with type code

NSEXIT
CLEANUP RU (CU)

LOSTERM
reason code

(Type and reason codes shown in decimal)

"ns" means "exit routine not scheduled"

Session outage notification reasons	If LU is PLU			If LU is SLU		
	SCIP	**NSEXIT**	**LOSTERM**	**SCIP**	**NSEXIT**	**LOSTERM**
UNBIND type=01 received by PLU • UNBIND 01 sent by SLU.	01	CU	24, then 16	ns	ns	ns
UNBIND type=01 received by SLU • CLSDST OPTCD=RELEASE by PLU • See "CTERM Forced received by PLU" in the following rows • CLOSE by PLU.	ns	ns	ns	ns	01	ns
UNBIND type=02 received by SLU • CLSDST OPTCD=PASS by PLU.	ns	ns	ns	ns	02	ns
UNBIND type=12 received by this LU • Unrecoverable failure of other LU.	12	CU	12	ns	12	ns

Table 12. Session outage notification UNBIND type codes and reason codes (continued)

Session outage notification signals

SCIP UNBIND RU with type code

NSEXIT
CLEANUP RU (CU)

LOSTERM
reason code

(Type and reason codes shown in decimal)

"ns" means "exit routine not scheduled"

Session outage notification reasons	If LU is PLU			If LU is SLU		
	SCIP	**NSEXIT**	**LOSTERM**	**SCIP**	**NSEXIT**	**LOSTERM**
UNBIND type=TC received by this LU for any UNBIND type not listed above, including (but not restricted to): • TC=07 Virtual route inoperative • TC=08 Route extension inoperative • TC=09 Hierarchical reset because of SSCP session activation • TC=10 SSCP session deactivated or failed • TC=11 Virtual route deactivated • TC=14 Recoverable failure of other LU • TC=15 Cleanup done by other LU • TC=254 session protocol is not valid or user-supplied sense code.	TC	CU	24, then 16	ns	TC	ns
CTERM Forced received by PLU (PLU sends UNBIND type=01 to SLU. No action is taken if PLU is an independent LU.) • VARY INACT,I of either LUs, CDRSC, CDRM, or associated component • HALT NET,QUICK • REQDISCONT Immediate from PU • Cross-Domain Takedown (Forced).	ns	ns	12	ns	ns	ns
CTERM Forced received by PLU (PLU sends UNBIND type=01 to SLU. No action is taken if PLU is an independent LU.) • Terminate Forced by SLU • LOGOFF UNCOND by SLU • TERMSESS UNCOND by SLU • VARY TERM,UNCOND of session.	ns	ns	20	ns	ns	ns

Table 12. Session outage notification UNBIND type codes and reason codes (continued)

Session outage notification signals

SCIP UNBIND RU with type code

NSEXIT
CLEANUP RU (CU)

LOSTERM
reason code

(Type and reason codes shown in decimal)

"ns" means "exit routine not scheduled"

Session outage notification reasons	If LU is PLU			If LU is SLU		
	SCIP	**NSEXIT**	**LOSTERM**	**SCIP**	**NSEXIT**	**LOSTERM**
CLEANUP received by this LU (UNBIND type=0C sent to other LU) • VARY INACT,F or R of LU, CDRSC, CDRM, or associated component • VARY TERM,FORCE of session • Cross-Domain Takedown Cleanup • CLOSE or ABEND by other LU • RU received larger than maximum size specified in BIND • Route extension inoperative	ns	CU	24 then 16 or 48	ns	CU	ns
CLEANUP received by this LU (UNBIND type=01 sent to other LU) • Terminates Cleanup by this or other LU (includes Terminate Forced changed to Terminate Cleanup by the SSCP).	ns	CU	12	ns	CU	ns
Global failure of this LU (UNBIND type=01 sent to other LU) (TPEND exit routine with reason code 8) • HALT NET,CANCEL • ABEND of application program • VTAM failure	ns	ns	ns	ns	ns	ns
Failure of this LU for this session (UNBIND type=14 sent to other LU)	14	CU	24, then 16	14	ns	ns
Buffer overflow at this LU (Does not cause session termination)	ns	ns	36	ns	ns	36
Test Request message received by PLU (UNBIND type=01 sent to the SLU) • From BSC 3270 attached to NCP	ns	ns	12	ns	ns	ns

Table 12. Session outage notification UNBIND type codes and reason codes (continued)

Session outage notification signals

SCIP UNBIND RU with type code

NSEXIT
CLEANUP RU (CU)

LOSTERM
reason code

(Type and reason codes shown in decimal)

"ns" means "exit routine not scheduled"

Session outage notification reasons	If LU is PLU			If LU is SLU		
	SCIP	**NSEXIT**	**LOSTERM**	**SCIP**	**NSEXIT**	**LOSTERM**
CTERM Orderly received by PLU (No action taken if PLU is an independent LU) (Does not cause session termination) • VARY TERM,COND of session • Terminate Orderly by SLU • TERMSESS COND by SLU • LOGOFF COND by SLU • Cross-Domain Takedown Orderly.	ns	ns	32	ns	ns	ns

Control blocks used for session establishment and termination

The two control blocks used for session establishment and termination are the request parameter list (RPL) and the node initialization block (NIB). The application-supplied operands for dial connections function provides a third control block, the application-supplied dial-parameter control block (ASDP), that is used only during session initiation.

Request parameter list (RPL)

The request parameter list, built with the RPL or the GENCB macroinstruction, contains information that describes a request for VTAM services. After the request has been completed and the event has been posted, the RPL can be used for another request.

When used for session establishment or termination, an RPL contains information that describes the session and how to establish or terminate it. A sample RPL which could be used with an OPNDST macroinstruction follows:

```
RPL1     RPL   AM=VTAM,ACB=ACB1,OPTCD=(ACCEPT,SPEC,ASY),              C
               NIB=NIB1,ECB=ECB1
```

where:

- RPL1 is the label for the macroinstruction and serves as the name of the RPL.
- AM=VTAM specifies the access method.
- ACB=ACB1 specifies that the request is being issued by the application program identified by the ACB labeled ACB1.

- OPTCD=(ACCEPT,SPEC,ASY), when used with an OPNDST macroinstruction, specifies that asynchronous processing is to be used to accept a CINIT for a session with the LU identified in the NIB.
- NIB=NIB1 specifies the address of the NIB containing the name of the SLU.
- ECB=ECB1 specifies that ECB1 is posted when the request defined by the RPL completes.

Node initialization block (NIB)

A node initialization block (NIB) describes a session that is to be established or terminated. After a session is established, VTAM supplies the communication identifier (CID), session, which is VTAM's means of identifying the session. VTAM also supplies a limited set of “device characteristics” for the LU.

An NIB can contain:

- The symbolic name of the LU
- The network identifier of the LU
- The generic resource name
- The communication identifier (CID) of the target session
- The processing options to be used when the application program communicates on the session being established
- The user data to be associated with the session
- The logon mode name
- The Start Data Traffic indication
- The address of a BIND area in which the application program can construct a set of session parameters
- The level of cryptography (required or selective) for the session, if cryptography is supported
- An indication of whether the application program specified required or selective encryption
- The address of the application-supplied dial parameter list
- An indication of the end user's national language.
- An indication not to do some MNPS data tracking. This is to reduce the performance impact of writing to the coupling facility for each send and receive on MNPS sessions.

The symbolic name of an LU is assigned at network definition. It is the name in the name field of the definition statement (for example, LU statement, APPL statement, LOCAL statement) used to describe the LU to VTAM. This symbolic name is generally used only when the application program establishes a session with an LU. After a session is established, the application program uses the CID to communicate on the session. For initiating a session, the symbolic name is placed in the NIB before the session-initiation request (OPNDST, SIMLOGON, or REQSESS) is issued. For accepting a CINIT, a symbolic name is coded only if you are accepting a session with a specific LU. For accepting a CINIT for a session with any LU, a symbolic name is not specified in the NIB; when the session is established, VTAM puts the symbolic name of the LU in the NIB.

The CID is a 32-bit number assigned by VTAM when a session is established. It is used to identify that session. The application program subsequently places the CID in an RPL or NIB control block to specify a particular session to VTAM. The

application program should not be written to be dependent on the internal structure of the CID, which should be treated as an unstructured 32-bit number.

The processing options (PROC) determine which characteristics VTAM assigns to the session (for example, whether certain input on the session causes VTAM to schedule a DFASY or an RESP exit routine). See Chapter 6, "Communicating with logical units," on page 151, for a description of the DFASY or the RESP exit routines.

The user data is a 4-byte field (USERFLD) that permits some relevant data to be associated with the session-initiation request or the session itself.

When a nonzero USERFLD is specified on certain requests that initiate a session (SIMLOGON, REQSESS, or CLSDST OPTCD=PASS), it becomes a correlator for the session-initiation request. This enables the application program to recognize events associated with a specific initiate procedure. Therefore, if a session-initiation fails after the request is accepted, a Notify request that includes the correlator from the NIB's USERFLD is presented to the application program's NSEXIT routine, to allow the application program to determine which session-initiation procedure failed. If the user field is 0, an NSPE request (rather than a Notify request) is presented to the NSEXIT routine. The correlator is also passed on to the requests to establish a session. Thus, when the LOGON exit routine is scheduled, as a result of a SIMLOGON macroinstruction, the CINIT request presented in the exit routine includes the correlator from the NIB used with SIMLOGON. When the SCIP exit routine is scheduled with a BIND request resulting from a REQSESS macroinstruction, the BIND request includes the correlator from the NIB used with REQSESS. The correlator is arbitrarily assigned by the application program and is meaningful only to the application program. Therefore, it is passed only to the application program that created it. For example, if application program A issues a REQSESS macroinstruction for a session with application program B and specifies a correlator, that correlator would not be passed to application program B's LOGON exit routine. However, when program B establishes the session by issuing OPNDST OPTCD=ACCEPT, the resulting BIND request (received in an SCIP exit routine in application program A) includes that correlator, thus enabling program A to recognize that the BIND resulted from issuing the REQSESS macroinstruction.

On a session-establishment request, (OPNDST OPTCD=ACQUIRE, OPNDST OPTCD=ACCEPT, or OPNSEC), the USERFLD is associated with the session. This USERFLD value is placed in the USER field of the RPL upon completion of each communication request (SEND, RECEIVE, RESETSR, and SESSIONC) on the session. For OPNDST OPTCD=ACQUIRE, the USERFLD supplies only the user data to be associated with the session being established, and is not used as a correlator for the session initiation request generated by OPNDST OPTCD=ACQUIRE.

The logon mode name is the name of an entry in a logon mode table. The entry contains the session parameters to be passed to the PLU in the CINIT. The logon mode name also indirectly specifies the class of service to be used for the session.

The BNDAREA operand can be specified in the NIB macroinstruction to give the location of the BIND area where an application program at the primary end of a session can predefine or dynamically construct a set of session parameters to be used for the session. When the BNDAREA operand contains an address, the LOGMODE operand (*logon mode name*) is ignored. (The ISTDBIND DSECT can be used to set up session parameters in the BIND area.) Similarly, BNDAREA can be

used to designate an area in which a secondary application program can build the session parameters for a negotiable BIND response.

The Start Data Traffic indicator (the SDT operand) specifies whether the application program acting as the PLU issues the Start Data Traffic request (SDT=APPL) or whether VTAM should issue that request automatically as part of OPNDST processing (SDT=SYSTEM). The transmission services profile in the session parameters indicates whether the Start Data Traffic (SDT) request is used in the session. When you indicate use of the SDT request in the transmission services profile, you must send the request:

- During initial session-establishment processing
- After you send a Clear request
- After sequence number resynchronization to inform the SLU that the flow of requests and responses can begin.

For many application programs, it is convenient and adequate to let VTAM issue the Start Data Traffic request during initial OPNDST processing (that is, allow SDT=SYSTEM to take effect by default when defining the NIB). However, the PLU must still send an SDT request after a Clear request is sent.

For an application program acting as the SLU, the SDT indication is used to specify whether the application program wants VTAM to respond automatically to SDT, or to allow the application program to respond. If SDT=APPL is specified, the application program must respond to SDT requests received in the SCIP exit routine by issuing a SESSIONC macroinstruction with CONTROL=SDT and STYPE=RESP.

The level of cryptography for a session can be required or selective. In a required cryptographic session, all data requests are enciphered before they are sent. In a selective cryptographic session, data requests are enciphered based on the setting of the RPLCRYPT bit in the RPL, which can be set by the RPL's CRYPT keyword. Even though no cryptography is specified, either end of the session can still require cryptography because of system definition requirements. In this case, enciphering occurs transparently to the application program. See “How VTAM determines the level of cryptography for a cryptographic session” on page 133 for further information on the levels of cryptography, and “Sending and receiving enciphered data requests” on page 218 for level of cryptography information that applies to MVS-only systems.

The Programmed Cryptographic Facility (PCF) and the Cryptographic Unit Support Program (CUSP) do not support network-qualified names. If the application program is using cryptographic facilities, the 8-byte LU name of the partner using cryptography must be unique in all interconnected networks. VTAM uses 8-byte names on the interface to the PCF and CUSP.

The application-supplied operands for dial connections function enables an application to supply certain parameters that are coded on the PATH definition statement and PU definition statement. For more information refer to “Application-supplied dial parameter control block (ASDP)” on page 119.

The NIB also contains a coded value that specifies the end user's national language. The application program receives this value from the device characteristics area of the NIB. The hex code can be specified on the LANG parameter in the mode table entry (MODEENT macroinstruction) that is associated with an LU.

The end user can override this code by specifying the language code on the LANG or LANGTAB parameters on a USS command. The hex code that is available in DEVLANG is determined when the LU-LU session is established and cannot change during the session.

Table 13 shows the language codes that can be specified using the LANG or LANGTAB parameters and the corresponding hex values that are provided in the NIB. The application program must determine whether an LU can display the selected language of an end user. Sending double-byte character sets (DBCS) or other invalid characters to a terminal that does not support the character set can result in errors. Many IBM 3270 type terminals can be queried for character sets and DBCS support. The query bit, DEVQUERY, is determined by the LANG parameter of the MODEENT macroinstruction. Refer to the z/OS Communications Server: SNA Resource Definition Reference for more information.

Table 13. Language code settings (MVS only)

Hex code	Language code	Language name	Original name	Principal country
X'02'	ARA	Arabic [1]	Arabi	Arab Countries
X'03'	CHT	Traditional Chinese	Zhongwen	R.O.C.
X'04'	CHS	Simplified Chinese		P.R.C.
X'05'	DAN	Danish	Dansk	Denmark
X'06'	DEU	German	Deutsch	Germany
X'07'	DES	Swiss German	Schweizer-Deutsch	Switzerland
X'08'	ELL	Greek	Ellinika	Greece
X'09'	ENG	UK English	English	United Kingdom
X'00'		US English (default)		United States
X'01'	ENU	US English (specified)		United States
X'0A'	ESP	Spanish	Espanol	Spain
X'0B'	FIN	Finnish	Suomi	Finland
X'0C'	FRA	French	Francais	France
X'0D'	FRB	Belgian French		Belgium
X'0E'	FRC	Canadian French		Canada
X'0F'	FRS	Swiss French	Suisse-Francais	Switzerland
X'10'	HEB	Hebrew [1]	Ivrith	Israel
X'12'	ISL	Icelandic	Islensk	Iceland
X'13'	ITA	Italian	Italiano	Italy
X'14'	ITS	Swiss Italian	Italiano svizzero	Switzerland
X'11'	JPN	Japanese	Nihongo	Japan
X'15'	KOR	Korean	Choson-o; Hanguk-o	Korea
X'16'	NLD	Dutch	Nederlands	Netherlands
X'17'	NLB	Belgian Dutch		Belgium
X'18'	NOR	Norwegian	Norsk	Norway
X'19'	PTG	Portuguese	Portugues	Portugal
X'1A'	PTB	Brazil Portuguese		Brazil

Table 13. Language code settings (MVS only) (continued)

Hex code	Language code	Language name	Original name	Principal country
X'1B'	RMS	Rhaeto-Romanic	Romontsch	Switzerland
X'1C'	RUS	Russian	Russkij	USSR
X'1D'	SVE	Swedish	Svenska	Sweden
X'1E'	THA	Thai	Thai	Thailand
X'1F'	TRK	Turkish	Turkce	Turkey
X'3F'			Unknown language code[2]	

Note:

1. Language is written from right to left and is not supported by MVS Message Service.
2. Language specified on LANG or in the second parameter of LANGTAB is not found in this table. You can use this value to support a language not currently in the table.

OPNDST OPTCD=ACQUIRE and SIMLOGON allow lists of NIBs to be used. If OPTCD=CONANY is specified on the request, OPNDST or SIMLOGON is done for each NIB in the list until the operation is successful. In this case, one session at the most is established or initiated. If OPTCD=CONALL is specified, the operation is performed, in turn, for each NIB in the list, and continues until the end of the list is reached. The request is considered successful if at least one session is established or initiated.

To define an NIB list, you use the NIB or GENCB macroinstructions or an IBM-provided DSECT to define a series of contiguous NIBs. You can have any number of NIBs, but the last NIB in the list must have the LISTEND indicator (LISTEND=YES in the NIB macroinstruction). Other NIBs in the list must have LISTEND=NO.

The coding might look like this:

```
RPL1     RPL   AM=VTAM,ACB=ACB1,NIB=NIB1,OPTCD=ACQUIRE
NIB1     NIB   NAME=LU1,LISTEND=NO
NIB2     NIB   NAME=LU2,LISTEND=NO
NIB3     NIB   NAME=LU3,LISTEND=YES
```

If all the NIBs in a program are in one list, you might want to specify a working subset of the list for one operation. To do this, code the RPL to point to any one NIB in the list. The subset includes all NIBs from (and including) the NIB pointed to by the RPL through (and including) the next NIB in which the LISTEND indicator is set (LISTEND=YES).

This list form can be used only in the SIMLOGON, OPNDST OPTCD=ACQUIRE, and OPNDST OPTCD=RESTORE macroinstructions. The LISTEND indicator (LISTEND=YES) should be set in the NIB used with any other macroinstruction.

Application-supplied dial parameter control block (ASDP)

The ASDP, used only during session initiation, enables an application to provide dial parameters and other signal information. The application can indicate that VTAM temporarily replace the dial number, direct call line name, and DLCADDR values that are coded on the PATH definition statement. Additionally, if the ASDP's

no NODEID check flag is set to 0, the application can temporarily replace the following values that are coded on the PU definition statement in a switched major node:

- IDBLK and IDNUM
- CPNAME
- All of the preceding values

To retain network security, two operands, one on the APPL definition statement and one on the PU definition statement, restrict the application program's authority to provide the dial parameters.

Operand
Definition

AUTH=ASDP | NOASDP
This operand, coded on the APPL definition statement, specifies whether the application program has the authority to provide dial parameters during session initiation.

Use of the application-supplied operands for dial connections function requires authorization for the issuing application (AUTH=ASDP on the APPL definition statement).

ASDP=YES | NO
This operand, coded on the PU definition statement in a switched major node, specifies whether an application program can supply dial parameters for that PU.

Both operands must enable authority before the application program can supply the dial parameters for a specific PU. For example, if an authorized application program (AUTH=ASDP) supplies parameters for an unauthorized switched PU (ASDP=NO), session initiation fails.

The dial parameter list contains a header block followed by a maximum of six unique subfields. The header block contains a length field and a flag byte. The length field defines the length of the parameter list, including the length field, flag byte, and subfields. The subfields include:

Dial number
This subfield can contain up to a 32-byte (in EBCDIC) dial number that replaces the dial number that is coded on the PATH definition statement of a PU in a switched major node.

Direct call line name
This subfield can have a maximum of 8 bytes (in EBCDIC). The subfield contains a direct call line name that replaces the direct call line name that is coded on the PATH definition statement of a PU in a switched major node.

IDBLK and IDNUM
This 4-byte subfield provides an IDBLK and IDNUM that are compared to the IDBLK and IDNUM that are returned by the connected PU.

DLCADDR
This ASDP subfield contains up to 250 bytes of expanded signal information in the form of one or more DLCADDR data strings. Each DLCADDR data string contains:

- A 1-byte subfield_id that specifies a value in the range 1–96

- A 1-byte length field that indicates the amount of signal information contained in the DLCADDR data string
- DLCADDR signal information in hexadecimal format.

This information can replace the DLCADDR information that is coded on the PATH definition statement in a switched major node.

CPNAME
This subfield contains a CPNAME that is compared to the CPNAME that is returned by the attached PU. The subfield contains 1–17 bytes of data consisting of an optional 1–8 byte network qualifier concatenated with a period to a 1–8 byte CPNAME.

Connection name
This subfield contains a GRPNM that allows the application program to override the GRPNM operand value coded on the PATH definition statement in a switched major node.

The supplied dial parameters can have varied effects. The following conditions apply to the no NODEID check flag, IDBLK, IDNUM, and CPNAME values:

- If the no NODEID check flag is set to 1, VTAM establishes the LU session without checking the IDBLK, IDNUM, or CPNAME of the attached PU.
- If the no NODEID check flag is set to 0 and the application does not supply an IDBLK, IDNUM or CPNAME, VTAM performs current NODEID checking. Refer to z/OS Communications Server: SNA Resource Definition Reference for information pertaining to the current NODEID checking procedures.
- If the application supplies the IDBLK and IDNUM, or CPNAME or both, the dialed PU must return at least one value that matches the supplied values. If no value matches, session initiation fails.

The coded values on the PATH definition statement limit the types of dial parameters that can be provided by the application. Be aware of the following:

- If both the application and the PATH definition statement supply a dial number, DLCADDR value, or a direct call line name, VTAM dials the one that is specified by the application.
- If the application specifies a dial number or DLCADDR, but not a direct call line name, VTAM restricts session initiation to PATH definition statements coded with dial numbers or DLCADDRs.
- If the application specifies a direct call line name, but not a dial number or a DLCADDR, VTAM restricts session initiation to PATH definition statements coded with direct call line names.
- If the application specifies all three values (dial number, direct call line name, and DLCADDR), VTAM attempts session initiation using any available PATH statement.
- If the application does not supply a dial number, DLCADDR, or a direct call line name, VTAM uses the PATH definition statement values.

The following conditions apply when an application program initiates subsequent sessions.

- If the existing connection is initiated without using the application-supplied operands for dial connections function, the application cannot supply dial parameters for any additional session requests.
- If the existing connection is initiated with the application-supplied operands for dial connections function, the application can either supply no dial parameters

for additional session requests or supply the same dial parameters as supplied for the original request. If the application provides a type or value that differs, session initiation fails.

Be aware of the following:

- If the application specifies an application-supplied operand when initiating a session with a nonswitched device, the session fails.
- During dial processing, VTAM uses the supplied dial number, DLCADDR, or direct call line name as though it has been coded directly on the PATH definition statement.

VTAM uses the SIMLOGON, OPNDST OPTCD=ACQUIRE and NIB macroinstructions to locate and retrieve the dial parameters. The SIMLOGON and OPNDST OPTCD=ACQUIRE macroinstructions provide the specific NIB that contains the address of the application's dial parameter list. The NIB macroinstruction contains a keyword, ASDPAREA, that identifies the application's dial parameter list. ASDPAREA sets the NIBASDP indicator and the NIBASDPA field. NIBASDP is set to 1 when the application is providing the dial parameters. NIBASDPA defines the address of the application's dial parameter list.

Establishing parameters for sessions

As part of the session-establishment process, the PLU and the SLU must agree on the communication rules to be followed during the session. These communication rules, which are governed by session parameters, enable each end of the session to know what the other end of the session does and does not do in different communication situations.

Session parameters are indicators and numeric values that specify such things as "the PLU requires responses to requests" or "the SLU does not send end brackets if brackets are used." See "What is communicated: Requests and responses" on page 151, for a description of session parameters, and Appendix F, "Specifying a session parameter," on page 793, for an example of session parameter fields.

General pattern of agreement on session parameters

A session-establishment request reaches the PLU in the form of a CINIT request. A set of session parameters is included in the CINIT. These parameters are available for inspection by the PLU when it processes the CINIT.

During processing of the CINIT, the PLU can decide to use the session parameters suggested by the initiator of the session, or the PLU can choose a different set of parameters. In either case, when the application program issues an OPNDST macroinstruction to accept the session, it must designate a set of session parameters for the session. The session parameters are sent as part of the BIND request, which is created by VTAM as a result of the OPNDST macroinstruction. (Whether the application program uses the suggested parameters or different ones can be determined by user conventions. For example, the application program might always use the session parameters that accompany a CINIT or might always disregard the suggested parameters and select session parameters on the basis of some criteria chosen by the user.)

When the BIND request reaches the SLU, the SLU can examine the session parameters in the request. If the BIND is non-negotiable, the SLU must either

accept or reject the whole set of parameters. It cannot accept some and reject others. The SLU accepts the session parameters by sending a positive response to the BIND. If it rejects the parameters, the session is not established.

When the BIND is negotiable, the SLU can modify the parameters, and return the modified parameters in a positive response to the BIND. These parameters are then examined by the PLU, and data traffic within the session commences if the PLU accepts the changes made by the SLU. If the PLU does not accept the parameters, it must terminate the session, for example by issuing CLSDST.

Defining and naming a set of session parameters (logon mode and class of service)

In many cases, it is convenient to predefine a set of session parameters that could be sent in the BIND. When a set is defined, a name is associated with the set. That name is known as the logon mode name. The *logon mode name* is used in Initiate requests, and in the LOGMODE operand of certain macroinstructions and VTAM operator commands to identify the set of session parameters.

Additionally, in VTAM, a *logon mode name* implies a particular class of service used for the session. Class of service is a set of characteristics used to construct a route between session partners.

Several session parameter sets, each with its own logon mode name, can group into a table known as a logon mode table. There is a logon mode table, a default logon mode name, and an associated default set of session parameters associated with each LU that can act as the secondary end of a session,

Instead of using a predefined session parameter set, an application program can build a set of parameters at the time it is needed. The set of parameters is built in an area of the application program known as a BIND area, whose address is placed in the NIB used for the OPNDST or OPNSEC macroinstructions.

Refer to the z/OS Communications Server: SNA Network Implementation Guide for information about logon mode tables, class of service, and other aspects of the session parameter definition process.

How logon mode names and session parameters are used

A logon mode name is used to suggest a particular set of session parameters and class of service (COS) that is used between session partners. The logon mode name can be specified by one of the session partners or the VTAM operator at different points in the session-initiation and session-establishment process. Table 14 shows how a logon mode name can be specified.

Table 14. How to specify a logon mode name

Sent by	Negotiation	Specified in
LU	Suggested	Initiate Request
SLU APPL	Suggested	LOGMODE field of the NIB during REQSESS
PLU APPL	Suggested	LOGMODE field of the NIB during SIMLOGON
	Set	With the OPNDST macroinstruction
VTAM Operator	Suggested	LOGMODE operand of the VARY LOGON command

The following steps describe how a logon mode name is used:

1. The logon mode name is supplied by a session partner or operator as part of an Initiate request.
2. The VTAM that owns the SLU maps the logon mode name in the logon mode table and translates it into session parameters and a COS name.
3. If necessary, the logon mode name is passed from the domain in which the Initiate request originates to the domain of the VTAM that owns the SLU and is then translated in that domain. If no logon mode name is supplied as part of the Initiate request, the appropriate default session parameters and COS name are determined in the VTAM that owns the SLU.

   Note: If the logon mode name supplied as part of the intitate request cannot be translated in the VTAM that owns the SLU, VTAM might use the default logon mode entry ISTCOSDF. For additional information regarding the ISTCOSDF start option, refer to z/OS Communications Server: SNA Resource Definition Reference.

   For the OPNDST macroinstruction, the COS is not specified in this field (because it is associated with a queued CINIT that cannot be overridden) unless it is an OPNDST OPTCD=ACQUIRE operation.
4. The session parameters and COS name are then passed to the VTAM that owns the PLU. VTAM uses the COS name to determine a set of COS parameters.
5. The session parameters and COS parameters are sent to the PLU in the CINIT.

NIB LOGMODE and BNDAREA operands

The NIB LOGMODE and BNDAREA operands are used with a number of macroinstructions to specify, directly or indirectly, a set of session parameters. The following text and Table 15 through Table 19 on page 125 show how these operands are used for each macroinstruction.

Table 15. Determining session parameters for an INQUIRE macroinstruction

NIB specifies LOGMODE=	NIB specifies BNDAREA=	INQUIRE OPTCD=SESSPARM issued for an LU in the same domain	A different domain
0	Reserved	The session parameters associated with a queued CINIT are placed in the RPL's AREA field. See the INQUIRE macroinstruction (OPTCD=SESSPARM) for a description of when queued CINIT is used.	Same as same domain
C' '	Reserved	The default session parameters associated with the LU are placed in the RPL's AREA field.	Option not valid
Logon mode name	Reserved	The session parameters associated with the specified logon mode name are placed in the RPL's AREA field.	Option not valid

Table 16. Determining session parameters for an OPNDST OPTCD=ACCEPT macroinstruction

NIB specifies LOGMODE=	NIB specifies BNDAREA=	OPNDST OPTCD=ACCEPT issued for an LU in the same domain	A different domain
0	0	The session parameters associated with a queued CINIT are used. See the OPNDST OPTCD=ACCEPT macroinstruction for a description of which queued CINIT is used.	Same as same domain
C' '	0	The default session parameters associated with the LU are used.	Option not valid
Logon mode name	0	The session parameters associated with the specified logon mode name are used.	Option not valid
Reserved	ISTDBIND area address	The session parameters associated with the specified BIND area are used.	Same as same domain

Note: In all uses of OPNDST OPTCD=ACCEPT, the class-of-service parameters specified in the queued CINIT are used.

Table 17. Determining session parameters for an OPNDST OPTCD=ACQUIRE macroinstruction

NIB specifies LOGMODE=	NIB specifies BNDAREA=	OPNDST OPTCD=ACQUIRE issued for an LU in the same domain or in a different domain
0	0	The default session parameters and class-of-service name associated with the LU are used.
C' '	0	Same as specifying LOGMODE=0 and BNDAREA=0.
Logon mode name	0	The session parameters and class-of-service name associated with the specified logon mode name are used.
Any of the preceding	ISTDBIND area address	The session parameters associated with the specified BIND area are used. The class-of-service parameters used are those associated with the default logon mode name.

Table 18. Determining session parameters for a SIMLOGON or CLSDST OPTCD=PASS macroinstruction

NIB specifies LOGMODE=	NIB specifies BNDAREA=	SIMLOGON or CLSDST OPTCD=PASS issued for an LU in the same domain or in a different domain
0	Reserved	The default session parameters associated with the LU are used.
C' '	Reserved	Same as specifying LOGMODE=0 and BNDAREA=0.
Logon mode name	Reserved	The session parameters associated with the specified logon mode name are used.

Table 19. Determining session parameters for a REQSESS macroinstruction

NIB specifies LOGMODE=	NIB specifies BNDAREA=	REQSESS issued for an application program in the same domain or in a different domain
0	Reserved	The default session parameters associated with the application program issuing the REQSESS macroinstruction are used.[1]
C' '	Reserved	Same as specifying LOGMODE=0 and BNDAREA=0.
Logon mode name	Reserved	The session parameters associated with the specified logon mode are used.

Note:

1. The logon mode name and session parameters are associated with the application program that issued the REQSESS macroinstruction (the SLU) and not the LU named in the NIB (the PLU).

When used by the INQUIRE OPTCD=SESSPARM macroinstruction:

- LOGMODE=0 indicates that VTAM is to take the session parameters associated with a queued CINIT (including any user data field, if present) and place them in the field pointed to by the AREA field of the RPL. The AREALEN field of the RPL must specify the length of AREA. If CINIT is not queued, (RTNCD,FDB2)=(X'14',X'4C') is set.
- LOGMODE=C' ' indicates to VTAM that the default session parameters for the LU are to be returned in the field pointed to by the AREA field. The AREALEN field of the RPL must specify the length of AREA. If a match is not found, (RTNCD,FDB2)=(X'14',X'4B') is set.
- LOGMODE=*logon mode name* indicates to VTAM the logon mode name with which it is to search the logon mode table defined for the LU named in the NIB. If a match is found, the session parameters are returned in the field pointed by the AREA field of the RPL. The AREALEN field of the RPL must specify the length of AREA. If a match is not found, (RTNCD,FDB2)=(X'14',X'4B') is set.

Note: For details on the use of the NIBSYM and NIBCID fields, see the OPTCD=SESSPARM parameter in the INQUIRE macroinstruction description in "INQUIRE—Obtain logical unit information or application program status" on page 412.

When used by the OPNDST OPTCD=ACCEPT macroinstruction:

- LOGMODE=0 and BNDAREA=0 indicate that VTAM is to take the session parameters from a queued CINIT and construct a BIND request that is sent to the LU. If CINIT is not queued, (RTNCD,FDB2)=(X'14',X'4C') is set.
- LOGMODE=C' ' and BNDAREA=0 indicate that VTAM is to take the default session parameters for the LU named in the NIB and construct a BIND request that is sent to the LU. If CINIT is not queued, (RTNCD,FDB2)=(X'14',X'4C') is set.
- LOGMODE=*logon mode name* and BNDAREA=0 indicate to VTAM the logon mode name with which it is to search the logon mode table defined for the LU named in the NIB. The LU must be in the same domain as the application program that issued the OPNDST macroinstruction. If a match is found, the session parameters associated with that logon mode name are used to construct a BIND request that is sent to the LU. If a match is not found, or if the LU is in another domain, (RTNCD,FDB2)=(X'14',X'4B') is set, or if CINIT is not queued, (RTNCD,FDB2)=(X'14',X'4C') is set.
- LOGMODE=0 or C' ' or *logon mode name* and BNDAREA=*BIND area address* indicate that the session parameters in the specified BIND area are to be used to construct the BIND request. If CINIT is not queued, (RTNCD,FDB2)=(X'14',X'4C') is set.
- In all cases, the class-of-service parameters used are those specified in the queued CINIT.

When used by the OPNDST OPTCD=ACQUIRE macroinstruction:

- LOGMODE=0 or C' ' and BNDAREA=0 indicate to VTAM that default session parameters and the associated class-of-service parameters (defined for the LU named in the NIB) are to be used to construct the BIND, which is sent to the LU.
- LOGMODE=*logon mode name* and BNDAREA=0 indicate to VTAM the logon mode name with which it is to search the logon mode table defined for the LU named in the NIB. If a match is found, the session parameters and

class-of-service parameters associated with that logon mode name are used to construct a BIND, which is sent to the LU. If a match is not found, (RTNCD,FDB2)=(X'14',X'4B') is set.

- LOGMODE=0 or C' ' or *logon mode name* and BNDAREA=*BIND area address* indicate that the session parameters in the specified BIND area are to be used to construct the BIND. The class-of-service parameters used are those associated with the default logon mode name.

When used by the SIMLOGON or CLSDST OPTCD=PASS macroinstruction:

- LOGMODE=0 or C' ' indicates to VTAM that the default session parameters and associated class-of-service parameters for the LU are to be used as part of the CINIT that eventually results from the request. If a match is not found, (RTNCD,FDB2)=(X'14',X'4B') is set.
- LOGMODE=*logon mode name* indicates to VTAM the logon mode name with which it is to search the logon mode table for the LU named in the NIB. If a match is found, the session parameters and class-of-service parameters associated with that logon mode name are to be used as part of the resulting CINIT. If a match is not found, (RTNCD,FDB2)=(X'14',X'4B') is set.

When used by a REQSESS macroinstruction:

- LOGMODE=0 or C' ' indicates to VTAM that the default session parameters and associated class-of-service parameters for the application program issuing REQSESS are to be used as a part of the CINIT that eventually results from the REQSESS macroinstruction. If a match is not found, (RTNCD,FDB2)=(X'14',X'4B') is set.
- LOGMODE=*logon mode name* indicates to VTAM the logon mode name with which it is to search the logon mode table for the application program that issued the REQSESS macroinstruction. If a match is found, the session parameters and class-of-service parameters associated with that logon mode name are to be used as part of the resulting CINIT. If a match is not found, (RTNCD,FDB2)=(X'14',X'4B') is set.

Examples of how an application program processes session parameters

The following examples show how an application program processes session parameters.

Example 1: Using session parameters associated with CINIT

In this example, the application receives a CINIT for a session with an LU named LU1. VTAM then schedules the LOGON exit routine. The exit routine coding might be as follows:

```
INQ1     INQUIRE RPL=RPL1,OPTCD=SESSPARM
                (Test for (RTNCD,FDB2)=
                (00,05) in the RPL.)
                (Load value in RECLEN field of the RPL into
                register 7 and check that it does not
                exceed the length of AREA1.)
                .
                .
                .
         MODCB AM=VTAM,RPL=RPL1,AREALEN=(7)
                (Test return codes from execution of the MODCB macro.)
                .
                .
```

```
                .
INQ2     INQUIRE RPL=RPL1,OPTCD=SESSPARM
                .
                .
                .
                (Check session parameters placed in AREA1 and
                determine that they are appropriate.)
                .
                .
                .
         OPNDST RPL=RPL1,OPTCD=ACCEPT
                .
                .
                .
RPL1     RPL    AM=VTAM,NIB=NIB1,AREA=AREA1,AREALEN=0
NIB1     NIB    NAME=LU1,LOGMODE=0,BNDAREA=0
AREA1    DS     XL100
```

The INQUIRE macroinstruction at INQ1 attempts to get the session parameters (from the CINIT) with the RPL's AREALEN field set to 0. The macroinstruction fails with (RTNCD,FDB2)=(X'00',X'05') (insufficient length). Upon return from the macroinstruction, however, the RPL's RECLEN field contains the number of bytes needed for the session parameters (and any user data). The required length is loaded into register 7, and the MODCB macroinstruction is issued to put the value into the AREALEN field of the RPL. Then, at INQ2, the INQUIRE macroinstruction is issued again, causing VTAM to put the session parameters into AREA1. (The session parameters are those that are received with the CINIT.) The logic of the LOGON exit routine checks the session parameters and determines that they are appropriate for the LU and for the type of session that the application program has with that LU. Therefore, the application program issues the OPNDST OPTCD=ACCEPT macroinstruction to generate the BIND request, using the NIB whose BNDAREA and LOGMODE fields are set to 0. Because the LOGMODE and BNDAREA fields are set to 0, VTAM uses the session parameters associated with the CINIT to build the BIND that is sent to the LU. (An area of 100 bytes is shown in the example for the amount of storage reserved for AREA1; that might not be the correct value for your application program. The value should be equal to the maximum size of the session parameters plus any user data from the Initiate.)

Example 2: Locating the logon mode name in the CINIT RU

In this example, the application program transfers the LU-LU session to another application program in the same host by issuing CLSDST OPTCD=PASS. To keep the same class of service as specified for this session, the logon mode name is obtained from the CINIT RU and specified later with the CLSDST OPTCD=PASS.

This is an example of code that accesses the COS/LOGMODE/LIST vector containing the logon mode name:

```
*********************************************************************
*     EXAMPLE:  GET LOGON MODE NAME FROM CINIT RU (FOR CLSDST PASS)   *
*                                                                     *
*     REGISTER 1:  ADDRESS OF LOGON EXIT INPUT PARAMETER LIST         *
*     REGISTERS 2, 3, 4:  WORK REGISTERS                              *
*                                                                     *
*     LOCAL VARIABLES:  CINITVEC, CINITEND, LOGMODE                   *
*********************************************************************
*
*   GET CINIT ADDRESS
*
         L     REG04,16(,REG01)       GET ADDRESS OF READ ONLY RPL
         L     REG03,RPLAREA(,REG04)  GET ADDRESS OF CINIT
*
```

```
*   GET ADDRESS OF END OF CINIT
*
         LR    REG02,REG03            GET CINIT ADDRESS
         AL    REG02,RPLRLEN(,REG04)  ADD LENGTH OF CINIT
         ST    REG02,CINITEND         SAVE ADDRESS OF CINIT END
*
*   GET ADDRESS OF SLU NAME FIELDS
*
         LA    REG04,12(,REG03)       GET ADDRESS OF BIND IMAGE
         SLR   REG02,REG02            CLEAR FOR LENGTH
         ICM   REG02,3,10(REG03)      GET LENGTH OF BIND IMAGE
         ALR   REG04,REG02            GET ADDRESS OF SLU NAME FIELDS
*
*   REGISTER 4 NOW HAS THE ADDRESS OF THE SLU NAME TYPE
*
*   GET ADDRESS OF RESERVED FIELD (FORMERLY REQUESTOR ID)
*
         LA    REG03,2(,REG04)        GET ADDRESS OF SLU NAME
         SLR   REG02,REG02            CLEAR FOR LENGTH
         IC    REG02,1(,REG04)        GET LENGTH OF SLU NAME
         ALR   REG03,REG02            GET ADDRESS OF NEXT FIELD
*                                     (FORMERLY REQUESTOR ID, NOT
*                                      USED BY VTAM, NOW RESERVED)
*
*   REGISTER 3 NOW HAS ADDRESS OF RESERVED FIELD (FORMERLY REQUESTOR ID)
*
*   GET ADDRESS OF RESERVED FIELD (FORMERLY PASSWORD)
*
RESERV1  LA    REG04,1(,REG03)        GET ADDRESS OF OLD REQ ID
         SLR   REG02,REG02            CLEAR FOR LENGTH
         IC    REG02,0(,REG03)        GET LENGTH OF DATA
         ALR   REG04,REG02            GET ADDRESS OF NEXT FIELD
*                                     (FORMERLY PASSWORD, NOT USED
*                                      BY VTAM, NOW RESERVED)
*
*   REGISTER 4 NOW HAS ADDRESS OF RESERVED FIELD (FORMERLY PASSWORD)
*
*   GET ADDRESS OF USER AREA
*
RESERV2  LA    REG03,1(,REG04)        GET ADDRESS OF OLD PASSWORD
         SLR   REG02,REG02            CLEAR FOR LENGTH
         IC    REG02,0(,REG04)        GET LENGTH OF DATA
         ALR   REG03,REG02            GET ADDRESS OF USER AREA
*
*   REGISTER 3 NOW HAS ADDRESS OF USER AREA
*
*   GET ADDRESS OF DEVICE INFORMATION (NOT USED BY APPLICATIONS)
*
         LA    REG04,1(,REG03)        GET ADDRESS OF USER AREA
         SLR   REG02,REG02            CLEAR FOR LENGTH
         IC    REG02,0(,REG03)        GET LENGTH OF USER AREA
         ALR   REG04,REG02            COMPUTE ADDRESS OF DEVICE INFO
*
*   REGISTER 4 NOW HAS ADDRESS OF DEVICE INFORMATION
*
*   GET ADDRESS OF KEY (NOT USED BY APPLICATIONS)
*
         LA    REG03,2(,REG04)        GET ADDRESS OF DEVICE INFO
         ICM   REG04,12,0(REG04)      GET LENGTH OF DEVICE INFO
         SRA   REG04,16               SHIFT RIGHT
         ALR   REG03,REG04            COMPUTE ADDRESS OF KEY
*
*   REGISTER 3 NOW HAS ADDRESS OF KEY
*
*   GET ADDRESS OF CONTROL VECTORS
*
         LA    REG04,1(,REG03)        GET ADDRESS OF KEY
```

```
         SLR   REG02,REG02            CLEAR FOR LENGTH
         IC    REG02,0(,REG03)        GET LENGTH OF KEY
         ALR   REG04,REG02            COMPUTE ADDRESS OF VECTORS
*
*   REGISTER 4 NOW HAS ADDRESS OF VECTOR (IF PRESENT)
*
*   IS VECTOR WITHIN CINIT (IS THERE A VECTOR)?
*
         C     REG04,CINITEND         COMPARE TO CINIT END ADDRESS
         BNL   NOVECTOR               BRANCH IF NOT IN CINIT
*
*   LOOP TO LOOK FOR COS/LOGMODE/LIST VECTOR
*
CHECKVEC CLI   0(REG04),X'0D'         COS/LOGMODE/LIST VECTOR?
*                                     (REG 4 HAS ADDRESS OF VECTOR)
         BNE   NEXTVECT               NO, TRY NEXT VECTOR
*
*   COS/LOGMODE/LIST VECTOR FOUND
*   STORE DATA NEEDED
*
         MVC   LOGMODE,2(REG04)       SAVE 8-BYTE LOGON MODE NAME
         B     VECFOUND               CONTINUE PROCESSING (OUT OF LOOP)
*
NEXTVECT LA    REG03,2(,REG04)        GET ADDRESS OF VECTOR DATA
*                                     (REG 4 HAS ADDRESS OF VECTOR)
         SLR   REG02,REG02            CLEAR FOR LENGTH
         IC    REG02,1(,REG04)        GET LENGTH OF VECTOR DATA
         ALR   REG03,REG02            COMPUTE ADDRESS OF NEXT VECTOR
         LR    REG04,REG05            SAVE ADDRESS OF NEXT VECTOR
*
*   CHECK FOR END OF CINIT
*
ENDCHECK C     REG04,CINITEND         COMPARE TO END ADDRESS
         BL    CHECKVEC               VECTOR WITHIN CINIT, PROCESS IT
*
NOVECTOR .... CONTINUE PROCESSING
*
VECFOUND .... CONTINUE PROCESSING
*
         .
         .
CINITEND DS    A
LOGMODE  DS    CL8
```

Beginning with the BIND image at byte 12 of the CINIT RU, the fields and vectors are variable in length. See *SNA Formats* for a description of the CINIT RU. Each variable length field contains a 1- or 2-byte length field, followed by data. If a particular data item is not present, its length field contains zeros. To compute the address of the next variable length field (or vector), add the length of the data item to its address.

The coded example permits for multiple vectors, does not require a particular order of the vectors, and does not assume that a particular vector is present. At labels RESERV1 and RESERV2, variable length fields that are not used by VTAM are bypassed in the same way that the other fields are processed, by adding the length to the address of the data. In this way, the code is not dependent on these fields being reserved.

Example 3: Building and using session parameters in a BIND area

In this example, the application program initiates a session with an LU named LU2 in the same domain. A logon mode table is defined and was identified by the

system programmer in the MODETAB operand of the LU definition statement for LU2. The application program wants to get the default-session parameters from the logon mode table, modify them, and then send the modified parameters to the LU in the BIND when it acquires the LU. Negotiable BIND is used. The coding could look like this:

```
                .
                .
                .
         INQUIRE RPL=RPL2,OPTCD=SESSPARM
                .
                .
                .
                (Test and modify the session parameters in SPAREA2.)
                .
                .
                .
         MODCB AM=VTAM,NIB=NIB2,BNDAREA=SPAREA2
         OPNDST RPL=RPL2,OPTCD=ACQUIRE,
               AAREA=NBNDAREA,AAREALN=L'NBNDAREA
                .
                .
                .
RPL2     RPL   AM=VTAM,NIB=NIB2,AREA=SPAREA2,AREALEN=L'SPAREA2
NIB2     NIB   NAME=LU2,LOGMODE=C' ',PROC=NEGBIND
SPAREA2  DS    XL(BINUSE-ISTDBIND)
NBNDAREA DS    XL256
```

Because the NIB's LOGMODE field contains blanks, the INQUIRE macroinstruction causes the default session parameters from the logon mode table to be moved into SPAREA2. The application program then modifies the session parameters to fit the way it wants to communicate with LU2. The MODCB macroinstruction puts the address of SPAREA2 into the NIB's BNDAREA field. When the OPNDST macroinstruction is executed, the modified session parameters are transmitted to LU2 in the BIND request. The response to the negotiable BIND is returned in NBNDAREA. Example 4 shows how the response might have been sent.

Example 4: Responding to a negotiable BIND request

In this example, the application program, called LU2 and defined in "Example 3: Building and using session parameters in a BIND area" on page 130 receives a negotiable BIND request in its SCIP exit routine. It moves the CID obtained from the SCIP-exit-parameter list to the CID field of NIBR and moves the session parameters of the received BIND to AREAR. It can then modify the session parameters to make them suitable for the negotiable BIND response. The response is sent using the OPNSEC macroinstruction.

```
         OPNSEC RPL=RPLR
                .
                .
                .
RPLR     RPL   AM=VTAM,NIB=NIBR
*
NIBR     NIB   PROC=NEGBIND,BNDAREA=AREAR
*
AREAR    DS    XL256               BIND RESPONSE AREA
```

Establishing cryptographic sessions

VTAM supports single-domain and cross-domain cryptographic sessions.

Establishing single-domain cryptographic sessions

About this task

Before a cryptographic session can be established, VTAM must recognize a request for a cryptographic session, determine whether both ends of the session are capable of cryptography, verify that the levels of cryptographic sessions specified for both ends of the session are compatible, and get a cryptographic session key.

Compatible cryptographic levels are not necessarily the same type. For example, if one partner LU specifies selective encryption and the second LU specifies required encryption, the established session uses required encryption. VTAM rejects the cryptographic session request if one end of the session is not capable of cryptography and the other end of the session requires cryptography.

After determining that both ends of the session are capable of cryptography, VTAM issues a request to the cryptographic service to get a session cryptography key. The cryptographic service could be one of the following:

- IBM Programmed Cryptographic Facility (PCF)
- IBM Cryptographic Unit Support (CUSP)
- IBM Integrated Cryptographic Service Facility (ICSF/MVS)

In this request, VTAM specifies the name of the SLU. This name is not network-qualified.

If there is no SLU key for the SLU in the cryptographic key data set (CKDS) and if the session is to be selective or required, VTAM rejects the session-initiation request.

If there is an SLU key, VTAM gets a session-cryptography key enciphered under the SLU key and gets another copy enciphered under the host master key. VTAM saves the latter key. Then it puts the former key in the BIND request, and sends the BIND request to the SLU, which stores the session-cryptography key. Then the SLU generates an 8-byte random bit string (called the initial chaining value), saves it, enciphers it (under the session-cryptography key), puts it in the BIND response, and transmits the response to the PLU.

When VTAM receives the BIND response, it uses the session-cryptography key to decipher the initial chaining value and saves this deciphered value. To verify that both ends of the session are using the same session-cryptography key and initial chaining value, VTAM inverts the first 4 bytes of the initial chaining value, enciphers the value (under the session-cryptography key), and returns it to the SLU in a Cryptography Verification (CRV) request.

The SLU deciphers the value of the CRV request (using the session-cryptography key), inverts the first 4 bytes, and compares this value with the initial chaining value that it saved earlier. If the values are the same, both ends of the session are confirmed to be using the same session-cryptography key and initial chaining value, so the SLU sends a positive response to the CRV request. If the two values do not match, it sends a negative response to the CRV request.

When the PLU receives a positive response to its CRV request, normal VTAM session-establishment processing continues. If VTAM receives a negative response to its CRV request, it sends an UNBIND request to the SLU to terminate the session. At the PLU, the OPNDST macroinstruction fails with (RTNCD,FDB2)=(X'10',X'01').

The level of cryptography cannot be set by the application program in the negotiable BIND response; it must be specified in the NIB, using the ENCR operand of the NIB macroinstruction.

Establishing cross-domain cryptographic sessions

About this task

To establish an LU-LU cross-domain cryptographic session, VTAM must first establish a session between the SSCPs in each domain. When VTAM receives a request to establish a cryptographic session with a resource in another domain (at the secondary end of the requested LU-LU session), VTAM obtains a session-cryptography key. One copy of this key is enciphered under the cross-domain key of the SSCP of the primary end of the requested LU-LU session, and one copy is enciphered under the SLU key. VTAM then puts the latter copy in the BIND image in the CDCINIT request, the former copy in the CDCINIT request after the BIND image, and sends the CDCINIT request to the SSCP of the primary end of the requested session.

The VTAM at the primary end of the requested session processes the CDCINIT request by using the cross-domain key of the other SSCP to translate the session-cryptography key that is enciphered under the cross-domain key, so that it is enciphered under the host master key in its domain. VTAM then schedules the application program's LOGON exit routine; the application program then issues an OPNDST macroinstruction. VTAM processes the OPNDST macroinstruction by moving the session-cryptography key (enciphered under the SLU key) into a BIND request and by saving the session-cryptography key (enciphered under the host master key). The BIND request is then sent either to the host processor at the SLU (if the secondary end is an application program) or directly to the cryptographic device (if the secondary end is a cryptographic device). If the SLU is a cryptographic device, the cryptographic session-establishment processing is the same from this point as that for a single-domain cryptographic session. See “Establishing single-domain cryptographic sessions” on page 132 for information about single-domain sessions.

If the SLU is an application program, VTAM at the SLU takes the session-cryptography key sent in the BIND request, translates it using the application program's SLU key so that it is enciphered under the local host master key, and saves it. Then VTAM generates an initial chaining value, saves a copy of it, enciphers it (under the session-cryptography key), and passes it to the primary end of the session in the BIND response. From this point, the cryptographic session-establishment processing is the same as that for a single-domain cryptographic session. See “Establishing single-domain cryptographic sessions” on page 132 for information about single-domain sessions.

The level of cryptography cannot be set by the application program in the negotiable BIND response; it must be specified in the NIB, using the ENCR operand of the NIB macroinstruction.

How VTAM determines the level of cryptography for a cryptographic session

For an OPNDST request, VTAM determines the level of cryptography to be used in a cryptographic session by examining:

- The cryptographic requirements of the primary and secondary ends of the session as established at VTAM definition or by the VTAM MODIFY operator command
- The logon mode table entry
- The NIB value for the PLU

Table 20 shows the combination of values and the levels of sessions established. Table 21 on page 135 shows how one part of the cryptographic requirement is determined using both the logon mode table entry and the higher cryptographic level specified in the system definition for either end of the session.

For an OPNSEC request, VTAM determines the level of cryptography to be used in a cryptographic session by examining:

- The cryptographic requirements of the SLU as established at VTAM definition or by the VTAM MODIFY operator command
- The BIND request operands
- The NIB value for the SLU

Table 22 on page 136 shows the combination of values and the levels of sessions established.

For information pertaining to LU 6.2 sessions, refer to the z/OS Communications Server: SNA Programmer's LU 6.2 Guide.

Table 20. Level of cryptography for OPNDST requests

Primary end of the session, from VTAM definition or VTAM operator command (See note)	**Cryptographic requirement for the SLU**	**NIB value for the primary end of the session**	**Level of the cryptographic session requested in BIND**
Required	Required	Required Selective None	A required session is established.
	Selective	Required Selective None	
	None, but capable of cryptography	Required Selective None	
	None, and not capable of cryptography	Required Selective None	The request for session establishment fails.

Table 20. Level of cryptography for OPNDST requests (continued)

Primary end of the session, from VTAM definition or VTAM operator command (See note)	Cryptographic requirement for the SLU	NIB value for the primary end of the session	Level of the cryptographic session requested in BIND
Selective	Required	Required Selective None	A required session is established.
	Selective	Required	A required session is established.
		Selective None	A selective session is established.
	None, but capable of cryptography	Required	A required session is established.
		Selective None	A selective session is established.
	None, and not capable of cryptography	Required Selective None	The request for session establishment fails.
Optional or no specification	Required	Required Selective None	A required session is established.
	Selective	Required	A required session is established.
		Selective None	A selective session is established.
	None, but capable of cryptography	Required	A required session is established.
		Selective	A selective session is established.
		None	A session is established without encryption.
	None, and not capable of cryptography	Required Selective	The request for session establishment fails.
		None	A session is established without encryption.

Note: The cryptographic requirements specified on the VTAM definition statement and VTAM operator command for the PLU are compared. The higher of the two cryptographic levels is used.

Table 21. Establishing cryptographic requirements using logon mode entry and definition for secondary end of session

System definition (See note)	Logon mode table entry	Resulting cryptographic requirement
Required	Required	Required
	Selective	
	None	

Table 21. Establishing cryptographic requirements using logon mode entry and definition for secondary end of session (continued)

System definition (See note)	Logon mode table entry	Resulting cryptographic requirement
Selective	Required	Required
	Selective	Selective
	None	
Optional (but capable of cryptography)	Required	Required
	Selective	Selective
	None	None
None (not capable of cryptography)	Required	The request for session establishment fails.
	Selective	
	None	None

Note: The cryptographic requirements specified on the VTAM definition statement and VTAM operator command for the SLU are compared. The higher of the two cryptographic levels is used.

Table 22. Level of cryptography for OPNSEC requests

Secondary end of the session, from VTAM definition or command	BIND command operands	NIB value for the secondary end of the session	Level of the cryptographic session in the BIND response
Required	Required	Required Selective None	A required session is established.
	Selective	Required Selective None	The request for session establishment fails.
	None	Required Selective None	
Selective	Required	Required Selective None	A required session is established.
	Selective	Required	For non-negotiable BIND: the request for session establishment fails. For negotiable BIND: a required session is established.
		Selective None	A selective session is established.
	None	Required Selective None	The request for session establishment fails.

Table 22. Level of cryptography for OPNSEC requests (continued)

Secondary end of the session, from VTAM definition or command	BIND command operands	NIB value for the secondary end of the session	Level of the cryptographic session in the BIND response
Optional	Required	Required Selective None	A required session is established.
	Selective	Required	For non-negotiable BIND: the request for session establishment fails. For negotiable BIND: a required session is established.
		Selective None	A selective session is established.
	None	Required Selective	The request for session establishment fails.
		None	A clear session is established.

Restoring sessions pending recovery

Persistent LU-LU session support permits an application program to restore sessions following a failure and recovery. Single-node persistent sessions can be used to restore sessions disrupted by an application failure. Multinode persistent sessions can be used to restore sessions disrupted by a node failure, such as a failure in the hardware, operating system, or VTAM. MNPS can also be used to move applications to other VTAMs without the benefit of a node failure. See "Using persistent LU-LU session support" on page 66 for more information about the difference between single-node and multinode persistent sessions and how an application enables persistence.

An application enabled for single-node persistence can also use this function to allow an alternate application program to take over sessions from an original.

When the application restarts or an alternate application takes over, VTAM avoids session re-establishment flows by allowing the application program to restore any sessions that are pending recovery.

The recovering application can use INQUIRE OPTCD=PERSESS to identify the sessions pending recovery and must use OPNDST OPTCD=RESTORE to restore the sessions pending recovery. The application also has the option of terminating individual sessions.

Data tracking

When VTAM opens an ACB that is capable of persistence, regardless of whether persistence is enabled, VTAM begins tracking data about the RUs that flow across each session. This data tracking permits the recovering application to resynchronize the restored sessions.

The recovering application's ability to restore the sessions pending recovery and resynchronize RU flow depends on complete data tracking information. By

providing information about the most recent RUs that are processed, VTAM enables the application to determine which RUs were not processed and to resynchronize the sessions appropriately. VTAM tracks data for all applications that are capable of persistence. VTAM tracks data through the life of the session, including the time when the session is pending recovery. In a sysplex that supports multinode persistent sessions, this data is maintained in the multinode persistent sessions coupling facility structure for access by other VTAM end nodes in the sysplex.

When an application begins restoring the sessions, VTAM reconnects the sessions and sends information to the recovering application about the last RUs that were received on the sessions. The information is returned in the OPNDST OPTCD=RESTORE command and is in the format of the session state control vector (control vector hex 29). A copy of the BIND (as mapped by ISTDBIND) is also provided. See Table 110 on page 747 for more information pertaining to control vector hex 29 and Table 123 on page 819 for more information pertaining to the BIND. A 4-byte pointer in the NIB (NIBRPARM) indicates where the recovery information is stored.

VTAM tracks additional information for recovery of LU 6.1 and LU 6.2 sessions supported through the record application program interface (RAPI) as shown in Table 23.

Table 23. Information tracked for recovery of LU 6.1 and LU 6.2 sessions

INFORMATION	LU 6.1	LU 6.2
BID	X	X
BIS	X	X
FMH-5		X
MODENAME		X
Session Instance Identifier		X
Session Qualifier Pair	X	

BID data

VTAM tracks the sequence number, the request header, and the first 5 bytes of the last BID or BID response to flow on LU 6.1 and LU 6.2 sessions. The BID data is cleared following the transmission of a non-BID FMH-5 from the contention loser to the contention winner. This data is returned to the application program following an OPNDST RESTORE. The application program can then derive the BID status from this data in conjunction with the control vector hex 29 data.

BIS data

VTAM tracks the sequence number, the request header, and the first 5 bytes of the last BIS request to flow from the PLU to the SLU on an LU 6.1 or LU 6.2 session. VTAM also copies this data for the last BIS request to flow from the SLU to the PLU. This data is returned to the application program following an OPNDST RESTORE.

FMH-5

VTAM tracks the sequence number, the request header, and the first 16 bytes of the last FMH-5 request unit that flowed from the PLU to the SLU or the SLU to the PLU on an LU 6.2 session. This data is returned to the application program following an OPNDST RESTORE.

MODENAME

VTAM tracks the MODENAME from the user data field of an LU 6.2 BIND or BIND response and returns it to the application program following an INQUIRE PERSESS.

Session instance identifier

VTAM tracks the session instance identifier from the user data field of an LU 6.2 BIND response and returns it to the application program following an INQUIRE PERSESS.

Session qualifier pair

VTAM tracks the primary and secondary resource qualifiers from the user data field of an LU 6.1 BIND and BIND response and returns them to the application program following an INQUIRE PERSESS.

Restoring sessions

A recovering application program can use the INQUIRE macroinstruction to identify all sessions that are pending recovery. The application program handles each session on an individual basis. The OPNDST macroinstruction enables the application program to restore a session or sessions. It can be used for both PLU and SLU sessions. CLSDST permits an application program, acting as a PLU, to terminate a session pending recovery. A recovering application program that is acting as an SLU uses the TERMSESS macroinstruction to terminate a session pending recovery.

INQUIRE OPTCD=PERSESS builds an NIB or an NIB list that describes any sessions pending recovery. Following an INQUIRE OPTCD=PERSESS, VTAM indicates in the NIB whether the application is a PLU or an SLU.

Note: It is recommended that the application program use INQUIRE OPTCD=PERSESS to ensure that it recovers all existing sessions.

The INQUIRE coding might look like:

```
INQUIRE   RPL=rpl address,                  X
          OPTCD=PERSESS,                    X
          AREA=address of data area,        X
          AREALEN=length of data area
```

When OPTCD=PERSESS is specified, VTAM builds an NIB for each session pending recovery and stores the NIB or the NIB list in the data area that is provided by the application program. AREA contains the address of the data area used to hold the information for the sessions pending recovery. AREALEN specifies the length of this area.

Each NIB has a bit that indicates whether the application is the PLU or SLU. In addition, the NIB provides a pointer (NIBRPARM) to the restore parameter list. This parameter list contains the address of the BIND data for that particular session. The INQUIRE data includes the NIB, the restore parameter list, and the copy of the BIND for that session. Following the BIND for LU 6.1 and LU 6.2 sessions, selected user data structured subfields were passed on the original BIND. This information is present if the user data length field in the BIND (BINUSEL) is non zero and if the user data key is zero. For LU 6.1 sessions, the session qualifier pair structured subfield is returned. For LU 6.2 sessions, the modename pointer, session instance ID pointer, and the PLU or SLU network-qualified name structured subfields are returned. All of the structure subfields include the

preceding length and key fields. The restore parameter list points to the session qualifier pair for LU 6.1, and to the modename and session instance ID pointers for LU 6.2. You can obtain the LU 6.2 PLU or SLU network-qualified name structured subfield by parsing the vector. Recovery data that is available following INQUIRE OPTCD=PERSESS is shown in Figure 17, Figure 18 on page 141, and Figure 19 on page 142.

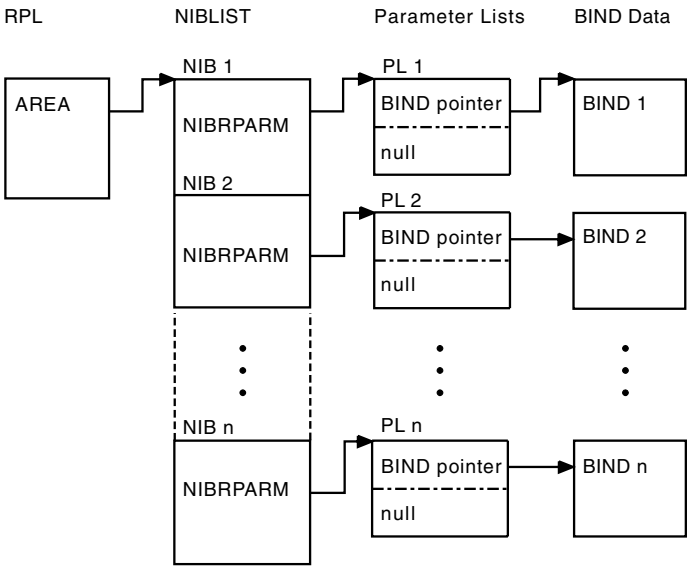


Figure 17. Recovery data for INQUIRE OPTCD=PERSESS for sessions other than LU 6.1 and LU 6.2

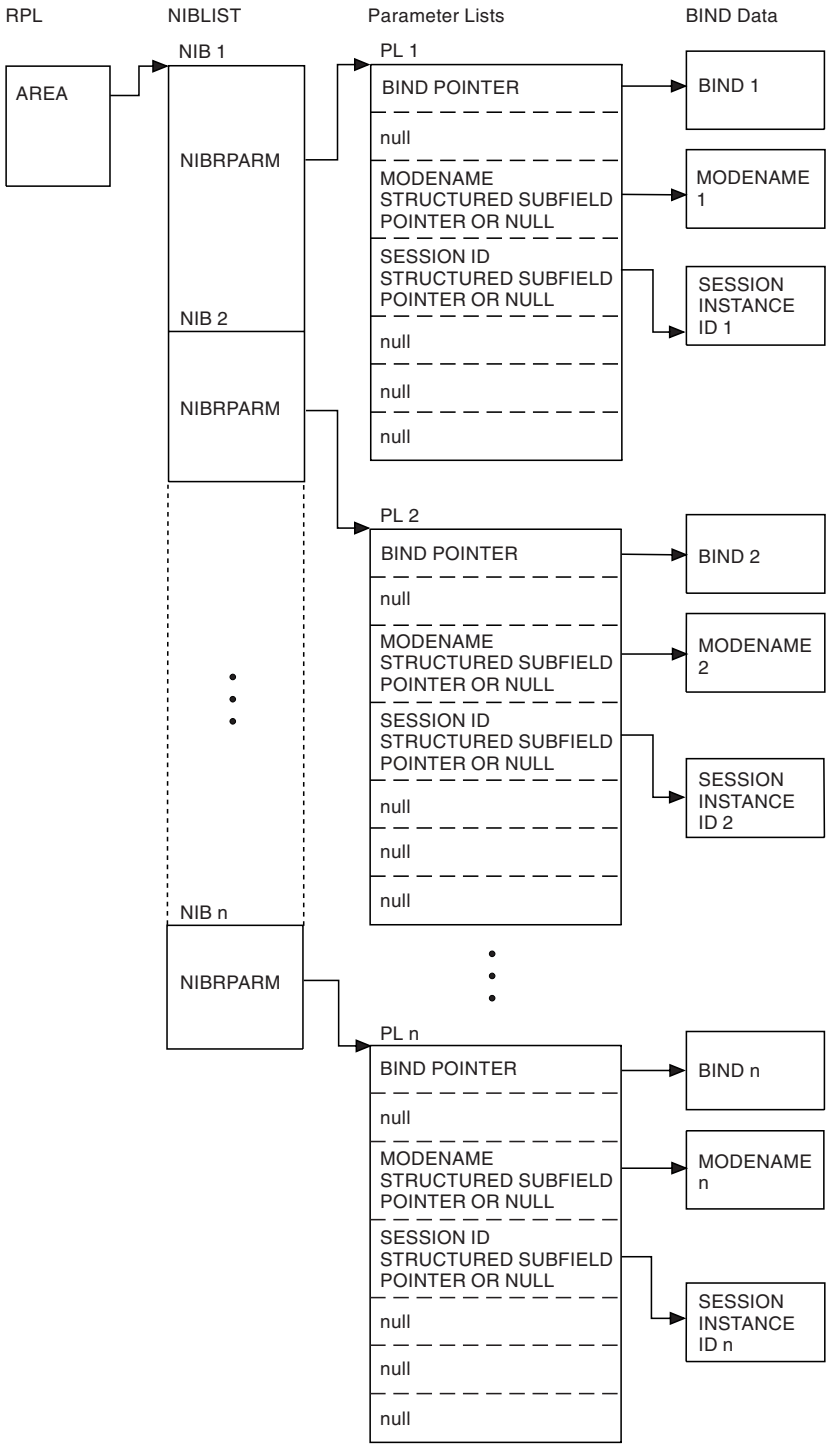


Figure 18. Recovery data for INQUIRE OPTCD=PERSESS for LU 6.2 sessions

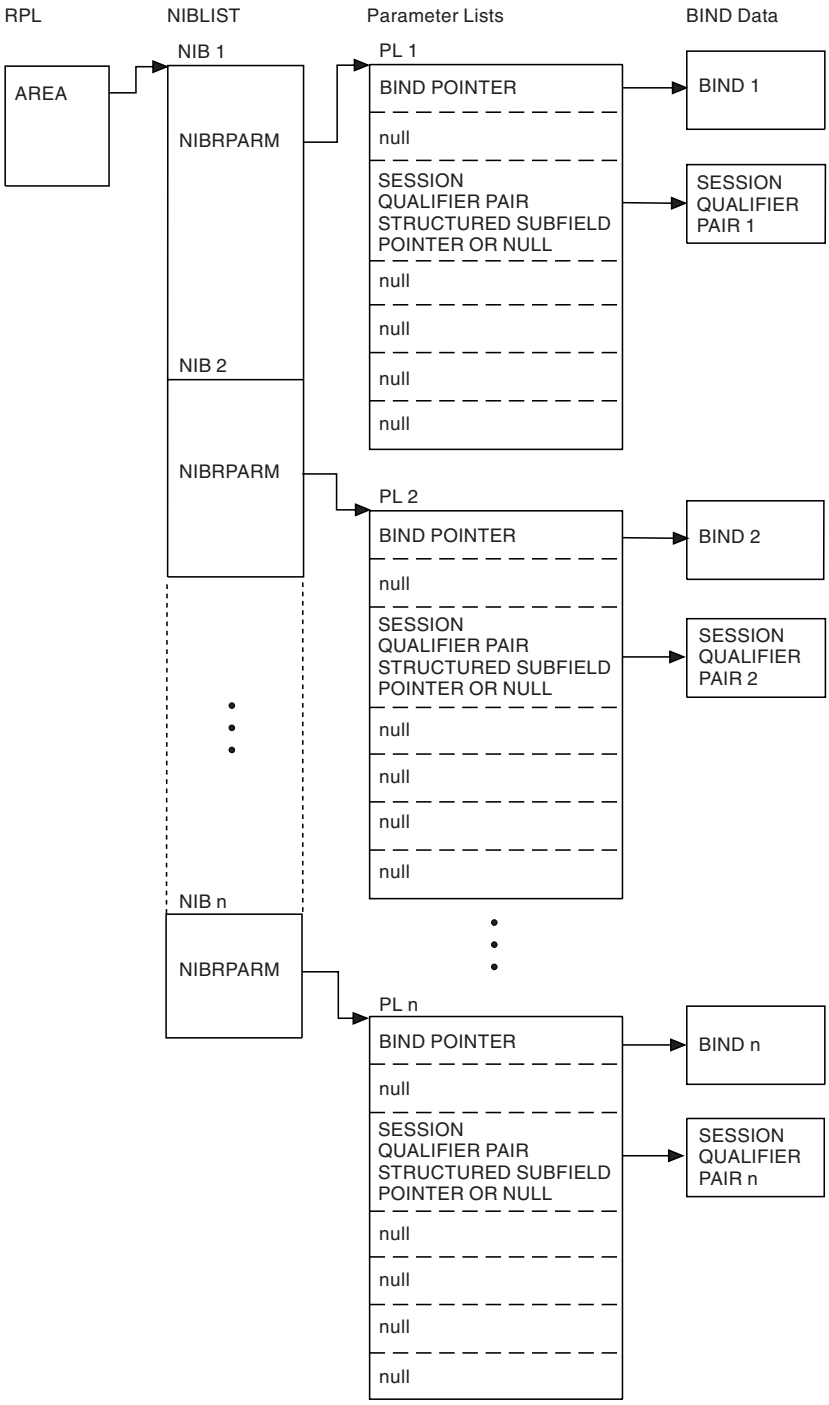


Figure 19. Recovery data for INQUIRE OPTCD=PERSESS for LU 6.1 sessions

If AREALEN in the INQUIRE macroinstruction is insufficient to hold the information for at least one session pending recovery, VTAM sets the return code (RTNCD,FDB2)=(X'00',X'05') along with a length field that shows the amount of storage needed to INQUIRE on at least one session. If AREALEN is insufficient to hold the information for all sessions pending recovery, VTAM builds as many NIBs as possible and sets the return code (RTNCD,FDB2)=(X'00',X'0D') along with a length field that shows the amount of storage that is used to build the partial list. To identify and recover the outstanding sessions, continue reissuing the INQUIRE

macroinstruction until VTAM sets either (RTNCD,FDB2)=(X'00',X'00'), which indicates that VTAM has processed all sessions pending recovery, or (RTNCD,FDB2)=(X'00',X'07'), which indicates that there are no more sessions pending recovery.

To restore the sessions represented by the NIBs in AREA, the application program issues OPNDST OPTCD=RESTORE. VTAM then puts the session state control vectors that correspond to the sessions in an alternate data area that is specified by the OPNDST macroinstruction. When all sessions pending recovery have been restored or terminated, the application recovery is complete.

The OPNDST coding might look like:

```
OPNDST   RPL=rpl address,                          X
         OPTCD=RESTORE,                            X
         NIB=NIB address,                          X
         AAREA=address of alternate data area,     X
         AAREALN=length of alternate data area
```

The application can point to the NIB list that is created by the INQUIRE OPTCD=PERSESS macroinstruction and stored in AREA, or the application program can point to its own NIB list. AAREA is an alternate area that is provided by the application program to hold the recovery information provided by OPNDST. AAREALN specifies the length of this area. If the area supplied is not large enough to hold all the OPNDST data, VTAM sets the return code (RTNCD,FDB2)=(X'00',X'05') and indicates the amount of space needed. VTAM does not recover any of the sessions until AAREALN is sufficient.

The recovery data that is provided following OPNDST OPTCD=RESTORE depends on whether NIBRPARM from a previous INQUIRE OPTCD=PERSESS was used. If the NIB pointed to the restore parameter list at the time the application program issued OPNDST (NIBRPARM from INQUIRE was used), the OPNDST data includes the control vector hex 29 data for the session pending recovery. In this case, as shown in Figure 20 on page 144, Figure 21 on page 145, and Figure 22 on page 146, VTAM does not modify the NIBRPARM, the BIND pointer, or the BIND data that was received from INQUIRE OPTCD=PERSESS. VTAM adds to each parameter list a pointer to the control vector hex 29 data that is located in AAREA.

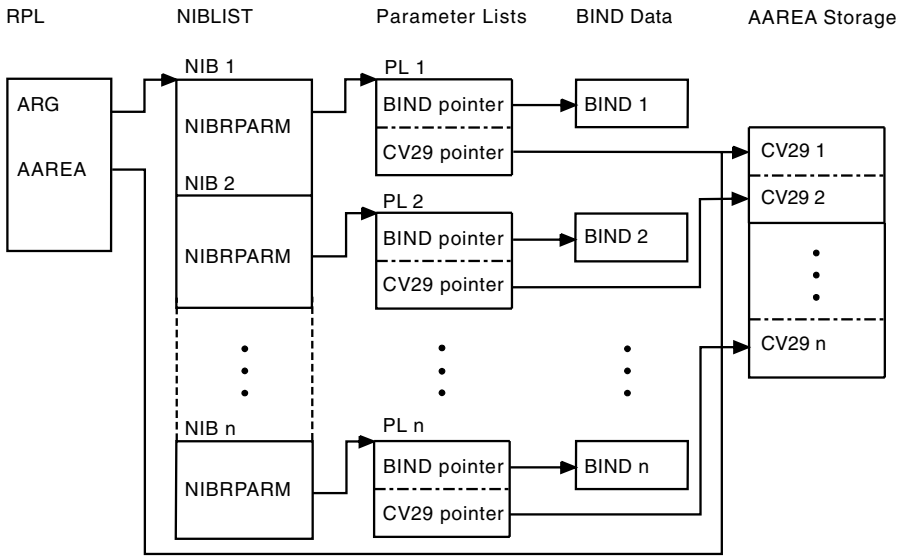


Figure 20. Recovery data for OPNDST OPTCD=RESTORE for sessions other than LU 6.1 and LU 6.2 (NIBRPARM from INQUIRE is used)

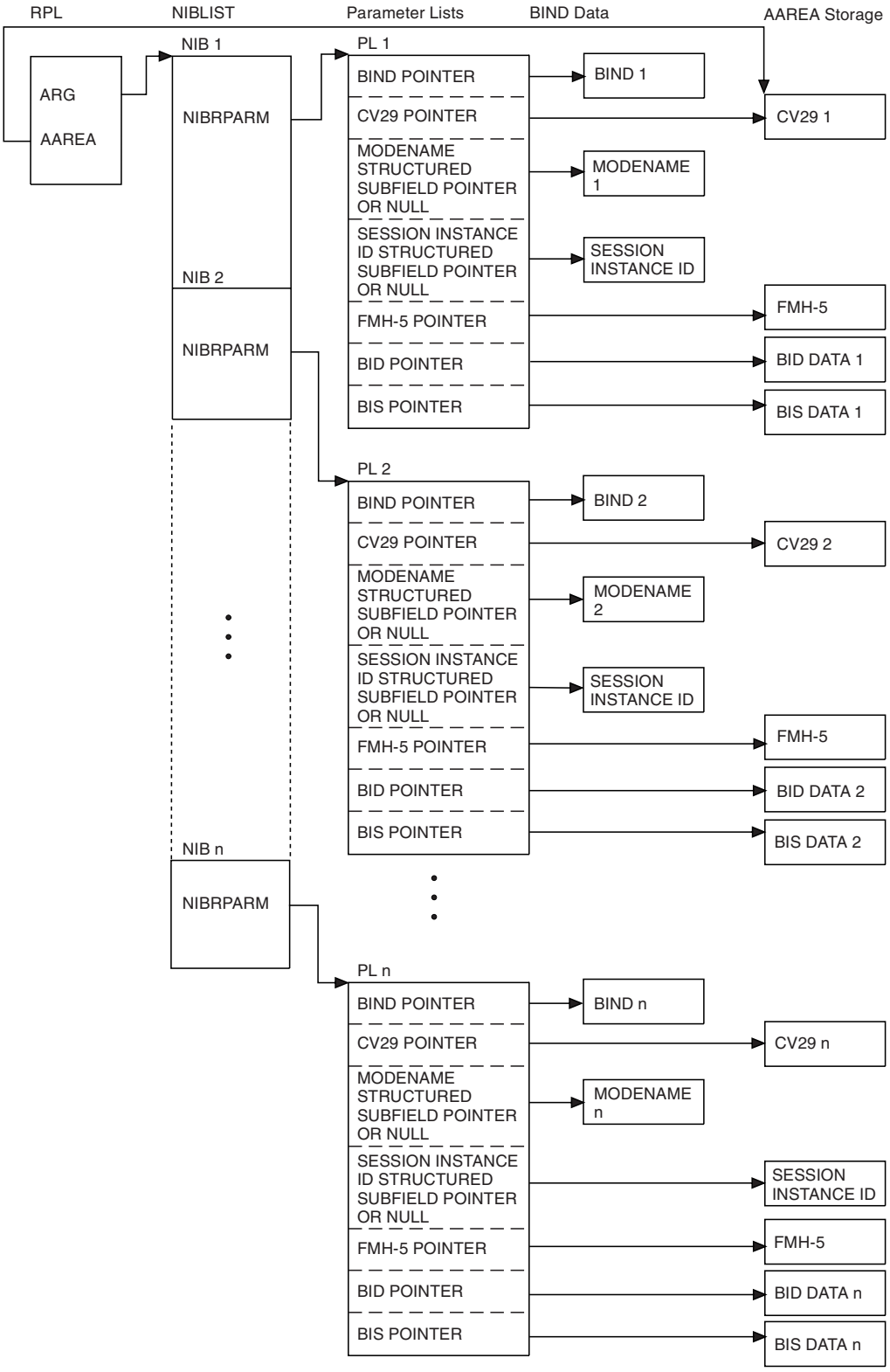


Figure 21. Recovery data for OPNDST OPTCD=RESTORE for LU 6.2 sessions (NIBRPARM from INQUIRE is used)

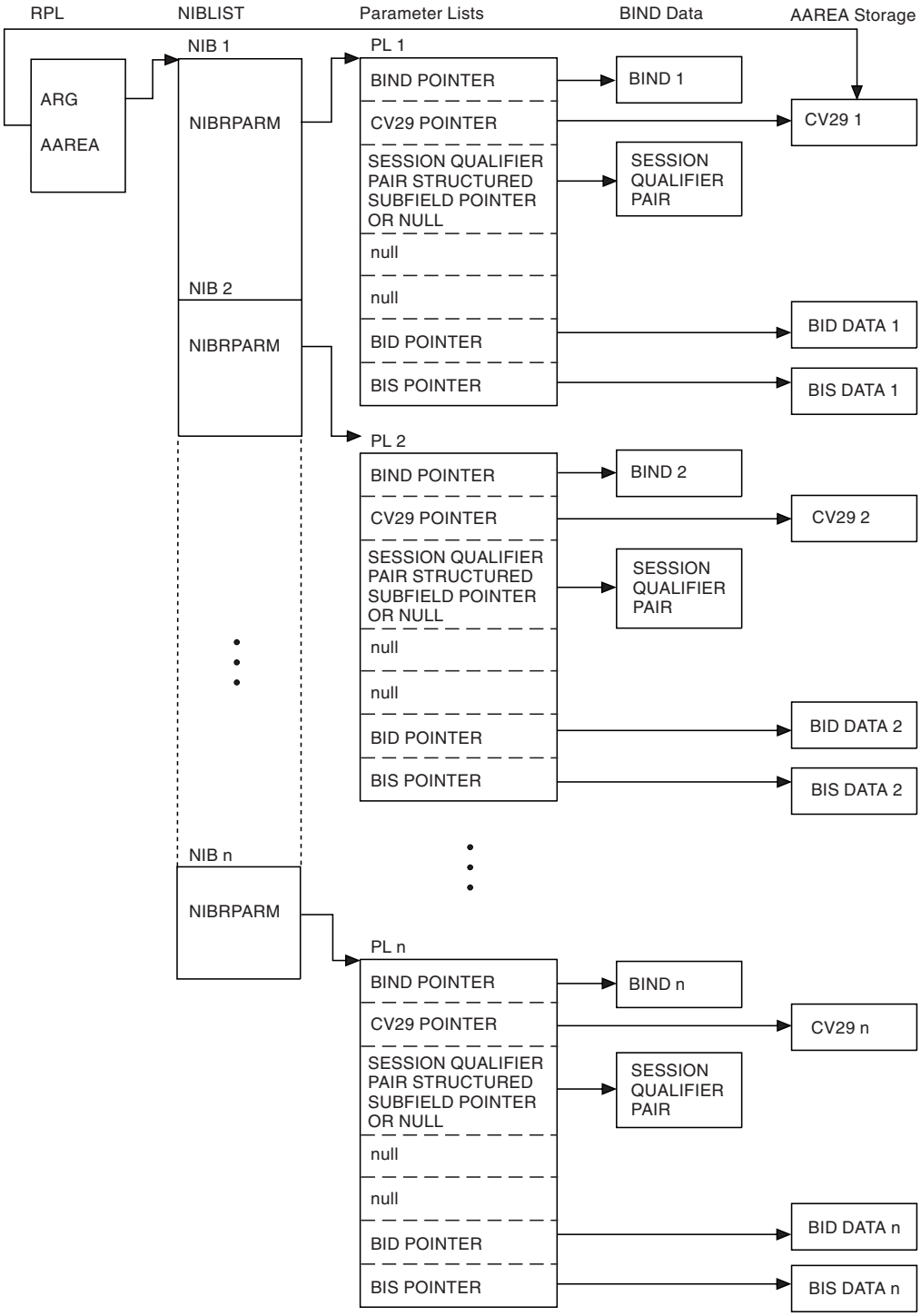


Figure 22. Recovery data for OPNDST OPTCD=RESTORE for LU 6.1 sessions (NIBRPARM from INQUIRE is used)

If the NIB did not point to the restore parameter list at the time the application program issued OPNDST (NIBRPARM = 0 when OPNDST OPTCD=RESTORE was issued), the OPNDST data includes the restore parameter list and the control vector hex 29 data. In this case, as shown in Figure 23 on page 147, Figure 24 on page 148, and Figure 25 on page 149, VTAM provides NIBRPARM. NIBRPARM points to a parameter list that contains a pointer to the control vector data only.

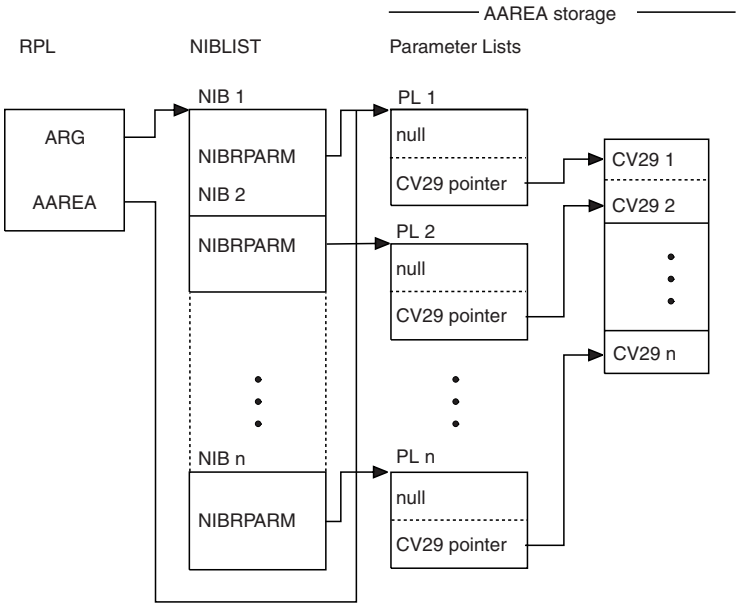


Figure 23. Recovery data for OPNDST OPTCD=RESTORE for sessions other than LU 6.1 and LU 6.2 (NIBRPARM from INQUIRE is not used)

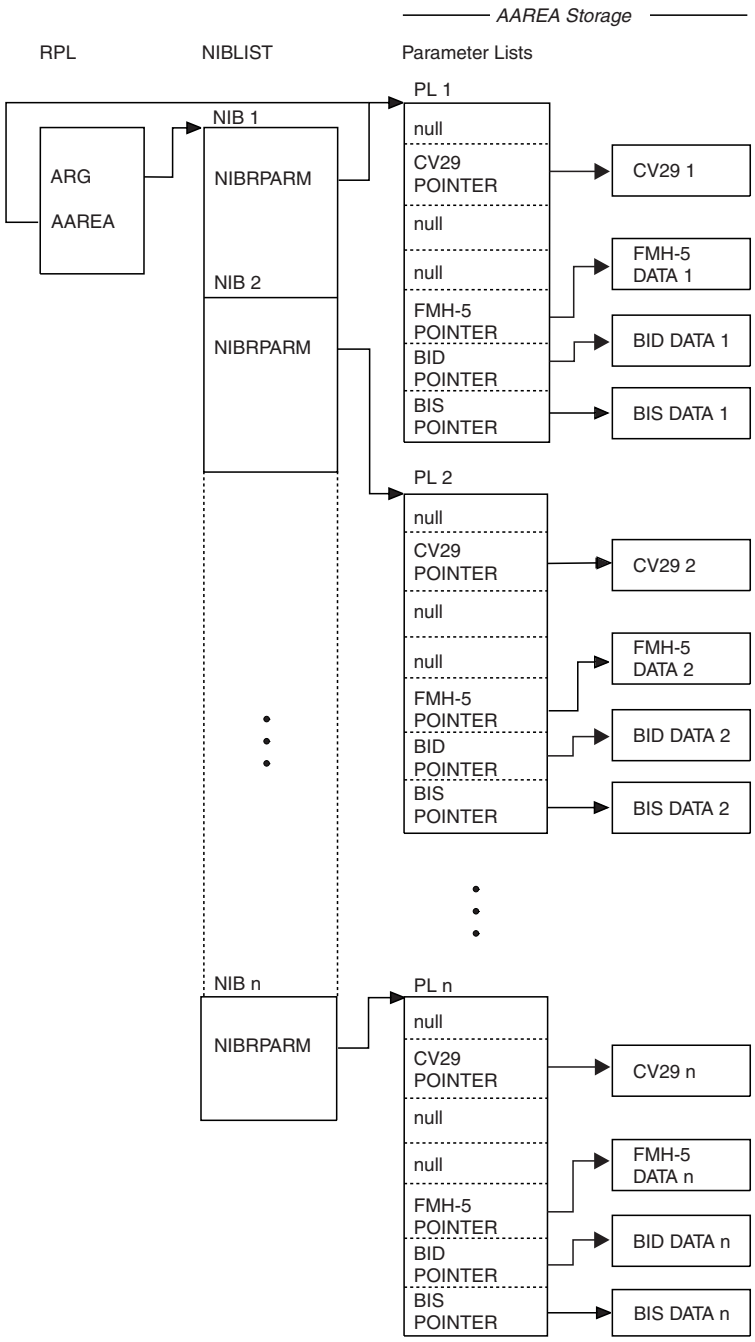


Figure 24. Recovery data for OPNDST OPTCD=RESTORE for LU 6.2 sessions (NIBRPARM from INQUIRE is not used)

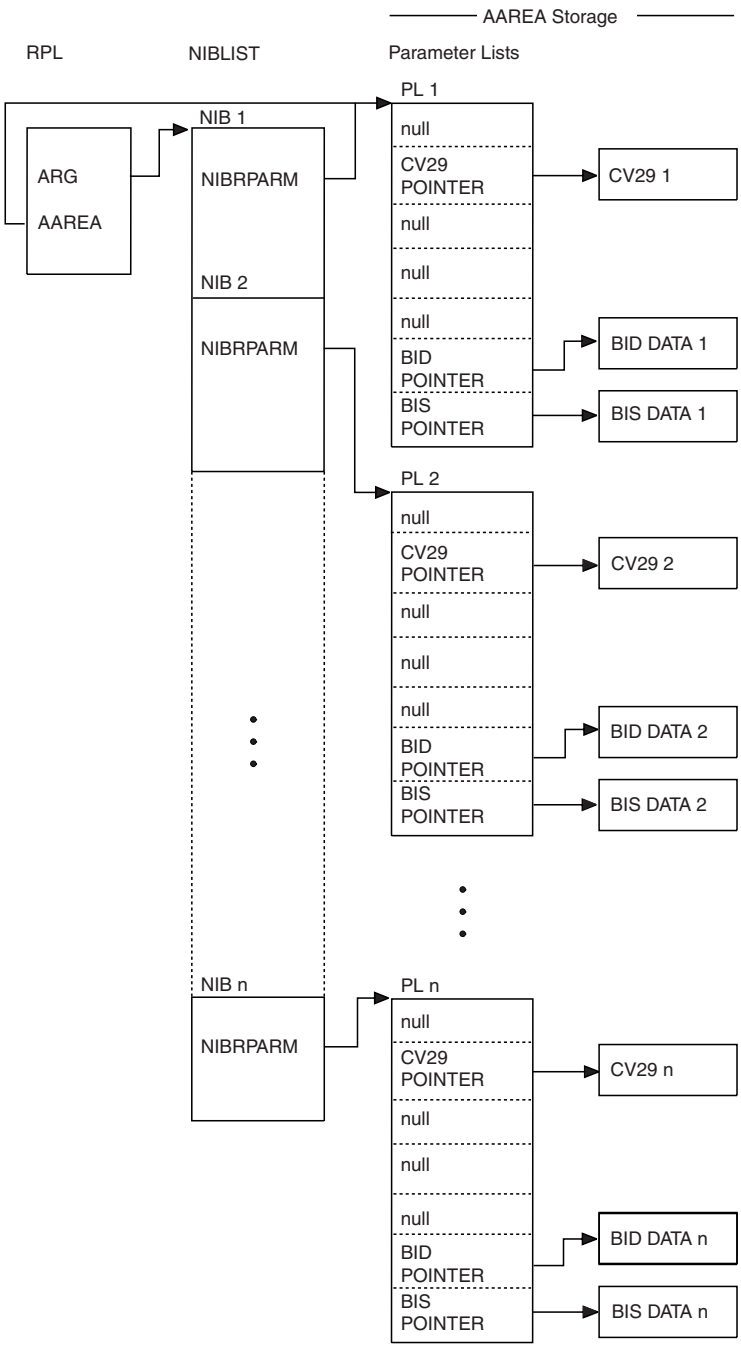


Figure 25. Recovery data for OPNDST OPTCD=RESTORE for LU 6.1 sessions (NIBRPARM from INQUIRE is not used)

Extended recovery facility (XRF) programming

The USERVAR, or user variable, maps a generic application name specified in a terminal logon to a specific application, based on the value of the variable. The specific application can be the active application program in an extended recovery facility (XRF) complex. An XRF complex consists of an application program and a backup copy of the application program. The USERVAR routes logon requests to

the active copy. This function is used by Information Management System (IMS) and Customer Information System (CICS) XRF complexes to map user logons to the IMS or CICS subsystem that is currently active.

Application programs communicate with an XRF complex through the USERVAR. This means that the application can use USERVAR names in place of LU names. VTAM automatically translates USERVAR names into LU names. The actual macroinstruction invocation always returns a successful RTNCD,FDB2 and returns as the translated name, the same name passed as input. You can use OPTCD=APPSTAT in place of OPTCD=USERVAR.

After an alternate subsystem takes over, the name of the VTAM application associated with the USERVAR must be changed to indicate the currently active application. This is done by the VTAM network operator, IMS, the NetView program, Network Communications Control Facility (NCCF), or any other program operator.

The primary XRF session is started with a BIND specifying XRF and carrying a subsystem-generated correlation ID. The NCP uses the correlation ID to verify that the primary and backup XRF sessions are related. This BIND flows through the NCP to the terminal. When a session is established with the active subsystem, the active subsystem informs the alternate. The primary XRF session then continues as usual. The alternate is given all relevant information about the active application's session with the LU, including the correlation ID. This must include BIND information, user name, and security information. The alternate subsystem then establishes a backup session using a BIND for the backup XRF session, which includes the same correlation ID used by the primary XRF session.

The activation of the primary and backup XRF sessions must be coordinated by the subsystem so that a BIND(BACKUP) is sent only after the primary XRF session has been established (that is, after the active PLU has sent a BIND and received the positive response). The BIND is intercepted and validated by the NCP to insure that a primary XRF session with the same correlation ID exists. The NCP then sends back the response to the BIND which informs VTAM and the subsystem that the session established is a backup XRF session. The Start Data Traffic (SDT) is sent as usual. The alternate subsystem then waits to be directed either to take over or UNBIND the session.

When the alternate subsystem takes over, it sends SESSIONC CONTROL=SWITCH. This RU informs the NCP that it is taking over the session. The NCP then terminates the original session, establishes the backup XRF session, and passes stored-session status back on the response. VTAM and the subsystem update any related control blocks from this response. The alternate subsystem then reads the active system's log and performs the necessary recovery.

Note: During a session takeover, VTAM does not provide responses for expedited flow requests to the failed active subsystem. The alternate subsystem is responsible for providing these responses, and PROC=APPLRESP must be coded on the NIB macroinstruction for all XRF sessions.

Chapter 6. Communicating with logical units

This chapter provides a general description of communication facilities. The chapter assumes that you use the operating system environment described in "Normal operating system environment for a VTAM application program" on page 30. Chapter 10, "Operating system facilities," on page 295, describes additional considerations if certain optional operating system facilities are used. Chapter 11, "Programming for the IBM 3270 Information Display System," on page 327, discusses some special considerations for communicating with selected SNA and non-SNA 3270 terminals.

Who is communicating: The VTAM application program and LUs

Both a VTAM application program, which is itself an LU, and the LUs with which it communicates can contain program logic. Because of this, communication between VTAM application programs and LUs have the following characteristics:

- The design and coding of the parts of an LU and of a VTAM application program that communicate with each other must be coordinated. In some cases, for example, both the application program and the LU can be designed by the same person; perhaps one is designed first and the other designed to complement it. This is a probable approach for application programs designed to service a particular kind of LU (for example, a 3600 logical work station). Or the application program can be designed as a standard program with which all LUs must conform, and LUs might be required to meet the application program's interface. In either case, both ends of the session must be coordinated.
- The existence of program logic in a terminal or cluster controller makes it possible to remove work from the host processor. The data that is exchanged between an application program and an LU can vary considerably, depending on what data processing (including the addition and deletion of device-control and format characters and data editing) can be performed by the LU rather than by the application program in the host processor.

What is communicated: Requests and responses

A VTAM application program and an LU exchange requests and responses to requests. A request normally contains data. In addition to data, or instead of data, a request can contain control information (described in VTAM publications as control requests or indicators). A response normally contains information about whether a particular request arrived and was processed successfully or unsuccessfully. The response can also contain control information. As explained later in this chapter, a response does not have to be returned for every request; it is possible for an application program and an LU to communicate without either side ever sending a response. (In this book, a request corresponds to an SNA request unit [RU] and associated request header [RH]. A response corresponds to an SNA response unit [RU] and associated response header [RH].)

In this chapter, the legend in Figure 26 on page 152 is used in figures that depict request and response flows:

© Copyright IBM Corp. 2000, 2013

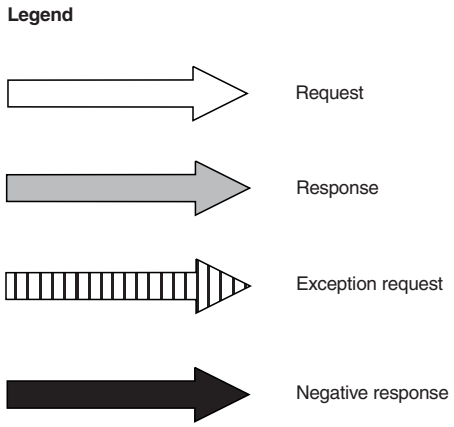


Figure 26. Legend for request and response flows

Figure 27 illustrates an exchange of requests and responses between an application program and an LU.

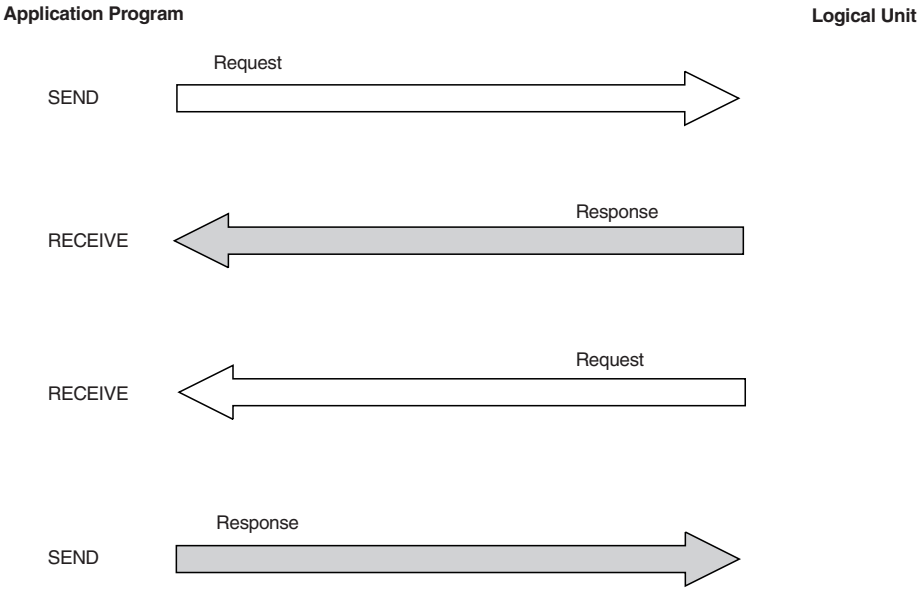


Figure 27. Exchange requests and responses

What a request contains

A request contains:

- Data
- Control information
- Combinations of the preceding (for example, data and indicators)

Data consists of information sent from or received in a VTAM application program's input/output area. Because both an application program and an LU contain program logic, each has the ability to insert, interpret, and extract information before forwarding it to a terminal operator, a recording medium, or some other destination.

Example of data exchange

This data, from a banking application, might be exchanged between an application program and an LU. An application program receives data on a session as the result of issuing RECEIVE. When RECEIVE completes, the input area specified in the AREA operand of RECEIVE contains data. For instance, after completion of RECEIVE, the input area might contain data in this format:

Code	Account	Amount
	Number	Deposited

The code might have been typed in by an operator at a terminal associated with the LU. The code is interpreted by the application program as a request for passbook update processing, and control is passed to the routine that handles that processing. The application program might prepare a data reply in this format:

Code	Account	Amount	New
	Number	Deposited	Balance

The application program sends the reply to the LU with a SEND macroinstruction, specifying the output area in the AREA operand. Any device-control or format information required to print the data at a printer or keyboard-display unit is furnished by the LU when the data arrives.

In addition to the transaction data, the LU can also send certain control indicators. For example, the application program and the LU can use change-direction indicators to ensure that only one of them at a time is sending (this method of communication is described in more detail later in this chapter). On receiving the request that contains data, the application program also checks the change-direction field of the RPL associated with the completed RECEIVE request:

```
TESTCB   RPL=(2),CHNGDIR=CMD
```

TESTCB tests whether a change-direction indicator is set on as part of the request. If not, the program prepares to receive a further request. If the indicator is on in the request, the program can send the reply. When a data reply, such as the preceding passbook update reply, is prepared, the program can indicate in the reply that the next request is to come from the LU. To do this, the program sets the change-direction indicator by specifying SEND CHNGDIR=CMD to send the reply.

The I/O area of the application program sends data only. The LUs specify all control and response information symbolically and receive this information by examining the appropriate fields of the RPL.

Certain control information can be sent only in requests that do not contain data. Examples are given later in this chapter.

What a response contains

A response to a request contains information about the success or failure of transmission and processing of a particular request. In sending a request, the VTAM application program or LU specifies the circumstances under which it expects a response to the request. When sending a response from a VTAM application program, control information is specified symbolically in a SEND macroinstruction. When receiving a response, VTAM makes response information

available in appropriate fields of the RPL associated with the completed RECEIVE or in a read-only RPL provided by VTAM on scheduling the VTAM application program's RESP exit routine.

Definite, exception, or no response indication

In our banking example, the request sent by the LU (requesting a passbook update) might indicate that:

- No response is returned, whether the request arrived and was processed successfully or not (no response requested).
- A response is returned only if the request encountered a transmission error or could not be processed successfully (exception response requested). Such a response is called a negative response.
- A response is returned, whether the request arrived successfully or not and was processed successfully or not (definite response requested). A positive response is returned for a successful operation; a negative response is returned for an unsuccessful operation.

A request for no response is feasible if the LU has its own means of determining failure of the request's transmission, such as using a timer or assuming that the terminal operator resends the request if there is no reply to it from the host processor after a certain length of time. In these cases, neither VTAM nor the host application program sends a response, because the LU cannot receive it.

Frequently, an LU requests that a response be returned only if the request is not received and processed successfully. If the request is received and processed successfully, no response is returned by the application program. However, if the request is not received successfully, VTAM indicates this in a return code and in additional information provided in RPL fields upon completion of the RECEIVE; the application program then sends a negative response. Even when the request arrives successfully, the VTAM application program, for its own reasons (for example, because it discovers the format of the request is improper), can send back a negative response, using SEND STYPE=RESP. The negative response is indicated by specifying RESPOND=EX; sense information can be provided by using the SSENSEO, SSENSMO, and USENSEO field of the RPL.

In our passbook-update example, if the request is received and processed successfully and a definite response was requested, the VTAM application program sends a positive response, using a SEND macroinstruction and specifying STYPE=RESP (a response) and RESPOND=NEX (positive). If some requests require a definite response and others do not, the application program determines whether to send a response by testing for a NEX indication in the RESPOND field of the RPL associated with a completed RECEIVE.

Figure 28 on page 155 illustrates the LU requesting (A) that a response be returned in either case and (B) that a response be returned only if the request does not arrive or is not processed successfully. Figure 28 on page 155 also shows (C) how the application program receives an exception request from VTAM.

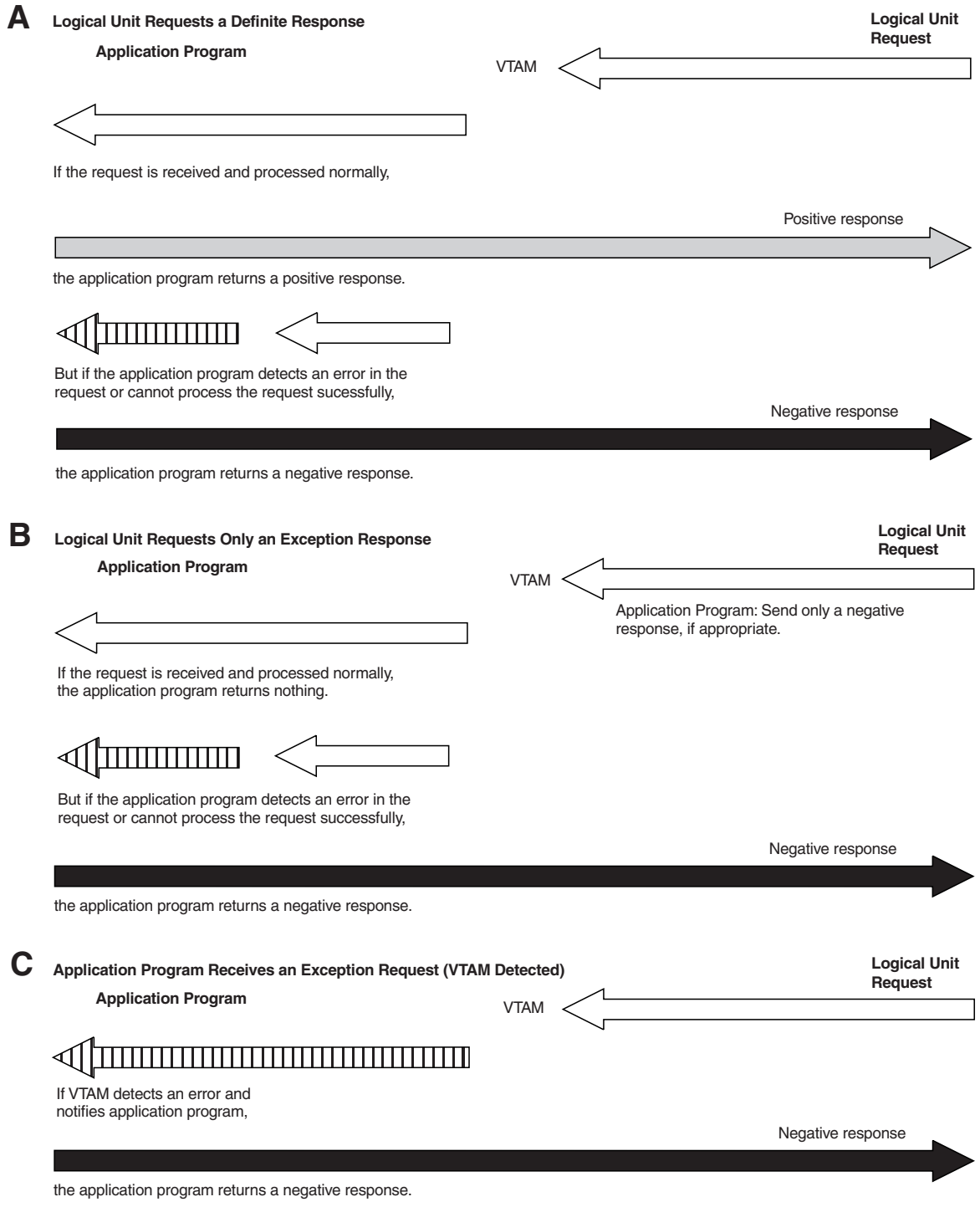


Figure 28. Receiving requests from an LU

Again, in the passbook-update example, when the application program sends the request that it prepares after performing the passbook update, the application program indicates the type of response it wants (no response, a response to an unsuccessful request only, or a response to every successful and unsuccessful request). The application program does this by specifying an appropriate indication in the RESPOND operand of the SEND macroinstruction. If a response is requested, it is usually received by the application program either by RECEIVE

RTYPE=RESP or by VTAM's scheduling the program's RESP exit routine. When the RESPOND field of the SEND RPL is set to NEX and the SEND macroinstruction includes the POST=RESP option, the SEND is not completed until the response is received. In this case, RECEIVE is not used to obtain the response; the response information is available in the SEND RPL when the operation is complete.

Definite response 1 and 2 indication

Besides being positive or negative, responses also differ as response type 1 (formerly known in SNA as an FME response) or response type 2 (formerly known in SNA as an RRN response). Every response, independent of being positive or negative, is designated by its sender as a response type 1, response type 2, or both. The meanings of the types of responses are agreed upon by the application program and the LU involved in the session, and can be determined by SNA protocols for the particular type of LU.

The application program indicates on each request whether it expects a definite response 1, a definite response 2, or both, to be returned. Combining these types of responses with the positive and negative response types described in the preceding section yields seven possible combinations of response types that can be indicated for a given request:

- Return a definite response 1 (either positive or negative)
- Return a definite response 2 (either positive or negative)
- Return definite responses 1 and 2 (either positive or negative)
- Return only a negative response 1
- Return only a negative response 2
- Return only negative responses 1 and 2
- Return no response of any kind.

When definite responses 1 and 2 are requested, it is possible that the responses might not be returned to the application program together. If the responses are returned to VTAM together, VTAM indicates that both responses have been received. However, if the responses are returned to VTAM separately, VTAM passes them to the application program as two separate responses.

The user should be aware that SNA protocols dictate when responses should be requested and what responses are returned. For example, returning the definite response 1 and 2 indicators in the same response is allowed by SNA, but returning them in two separate responses (that is, having multiple responses to the same request) is not allowed by SNA.

Another indicator that can be set in requests and responses is the queued response indicator, QRESP. It is described in detail in “Queued response notification” on page 182.

The LU, like the application program, also specifies, for each request, the types of responses it wants.

Three key elements in a RESPOND operand

When an application program that sends a request specifies the type of response it wants to receive, it either sets the RESPOND fields of the RPL before issuing the macroinstruction or specifies parameters in the RESPOND operand of the macroinstruction when the macroinstruction is issued. Three potential parameters can be specified in the RESPOND field or operand for each request:

- Nature of response desired for the request:

  NEX Positive or negative response

  EX Negative response only.
- Type of response desired for the request:

  FME Response type 1

  RRN Response type 2

  FME, RRN
  Both response types 1 and 2.
- Handling of a normal-flow response desired for the request when PROC=ORDRESP was set in the NIB:

  NQRESP
  Regular handling

  QRESP
  Handling as though the response is a normal-flow request from the LU.

Thus, an example of a RESPOND operand in which all three parameters are specified is:

```
RESPOND=(NEX,FME,NQRESP)
```

These parameters indicate that a positive or negative response is to be returned; the response is to be type 1; and the response is not to receive any special handling by VTAM.

How requests and responses are exchanged

Requests and responses are exchanged with an LU on a session by using SEND, RECEIVE, and SESSIONC macroinstructions and by using certain exit routines. Using SEND/RECEIVE communication, requests can be sent simultaneously by the application program and by the LU. Certain control requests can be sent ahead of other control requests and requests that contain data. Requests can be queued and responses correlated by using sequence numbers (a sequence number is automatically assigned to each request). This flow of requests and responses between a VTAM application program and an LU can be synchronized, if necessary, by stopping the flow, resetting sequence numbers at one or both ends of the session, and then restarting the flow. See “SEND, RECEIVE, and SESSIONC macroinstructions” for more information about these concepts.

Using the large message performance enhancement outbound (LMPEO) option, an application program can send data that exceeds the maximum RU size (for outbound RUs) with a single SEND operation. VTAM splits the data into multiple request units, which form a chain or partial chain of RUs. Refer to “Large message performance enhancement outbound (LMPEO) option” on page 183 for details about LMPEO.

SEND, RECEIVE, and SESSIONC macroinstructions

The VTAM application program sends and receives most requests and responses on a session by using SEND and RECEIVE and by using certain exit routines. A VTAM application program can receive some requests (expedited-flow data-flow-control requests) and some responses by having VTAM schedule an exit routine designed to handle these requests (a DFASY exit routine) and responses (a

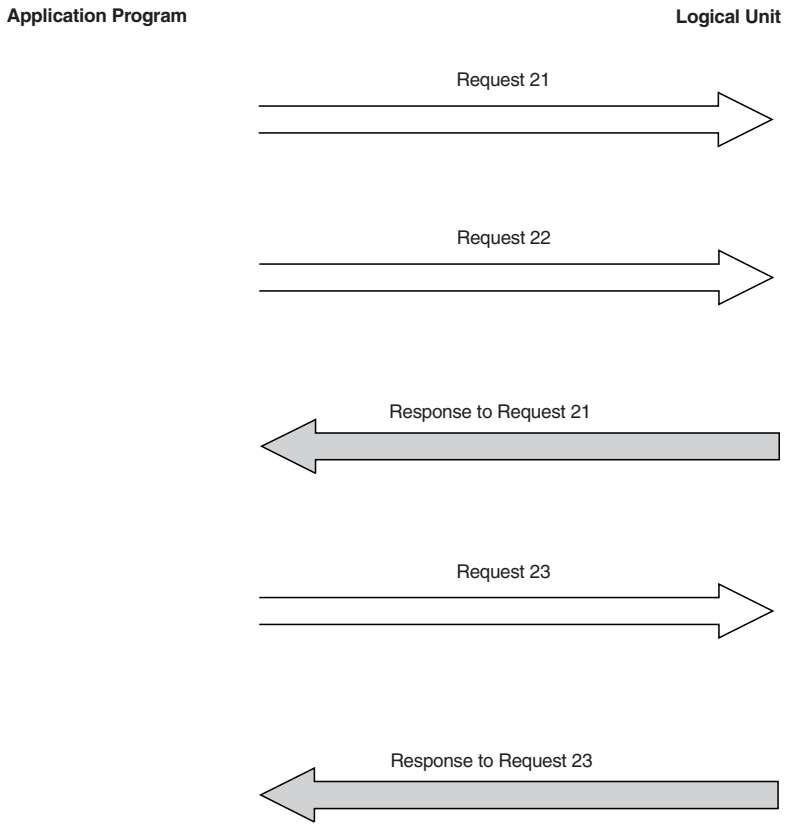
RESP exit routine). Alternately, requests and responses can be handled by RECEIVE RTYPE=DFASY or RECEIVE RTYPE=RESP, respectively. See Appendix A, "Summary of control block field usage," on page 633 for information on which RPL fields are automatically set by VTAM.

If the Start Data Traffic (SDT) indicator is required by the transmission-services (TS) profile in the session parameters, the sending and receiving of most requests and responses on a session cannot begin until an SDT request has been sent from the primary logical unit (PLU) to the secondary logical unit (SLU). See Appendix F, "Specifying a session parameter," on page 793, for more information on the TS profile. When required, the SDT request must be sent at the beginning of a session, and it must be sent within a session if the request flow is to be restarted after being stopped (with a Clear request). At the beginning of a session, either VTAM or the PLU application program sends the SDT request, depending on how the SDT field of the NIB is set when OPNDST is issued. If the SDT field indicates SYSTEM, the SDT request is sent by VTAM as part of the OPNDST processing. If the SDT field indicated APPL, the SDT request must be sent by the PLU application program, using SESSIONC. SESSIONC CONTROL=CLEAR can also be used to halt the flow of requests and responses. To resume request flow after it has been stopped, the PLU application program issues SESSIONC to send the SDT request.

If the SLU is an application program, VTAM or the SLU application program can send a response to that SDT request, depending upon how the SDT field of the NIB is set when the SLU application program issues the OPNSEC macroinstruction. If the SDT field indicates SYSTEM, VTAM sends the SDT response when the SLU receives the SDT request from the PLU. If the SDT field indicates APPL, the response must be sent by the SLU application program (using a SESSIONC macroinstruction).

Normal-flow and expedited-flow requests and responses

The normal-flow traffic between an application program and an LU includes requests containing data, requests that contain certain control information (called normal-flow data requests), and the responses to such data requests and control requests. Normal-flow requests are sent sequentially, one after the other, through the network, and a normal-flow request that is sent before another such request arrives sooner. Figure 29 on page 159 illustrates this principle.



Note: No response for request 22 was required.

Figure 29. Normal-flow requests are sent sequentially

Similarly, responses to normal-flow requests (called normal-flow responses) keep their order as they travel through the network. However, VTAM does not maintain the exact sequence relationship between requests and responses in relation to each other; that is, a response sent by an LU after a request can be presented to the application program before the request. The only way that an application program can be sure of receiving normal-flow requests and responses in the exact order that they are sent by the LU is by specifying RESPOND=QRESP (and POST=SCHED) on the macroinstruction or in the RPL used to send the request. See "Controlling the handling of normal-flow responses" on page 161 for more information about how VTAM handles responses. (Using the authorized path affects the order in which asynchronous operations complete, and because of this, the sequence in which requests are received can be affected. See "Additional coding considerations for authorized path" on page 302 for more information about authorized path usage.)

Certain control requests (called expedited-flow data-flow-control-requests) and responses to those requests (called expedited-flow traffic) are sent in a separate flow (called expedited flows) from the normal-flow requests and responses. These, together with all session-control requests and their responses, form the expedited-flow traffic in the network. The expedited-flow requests tell the receiver to do something that has higher priority than receiving normal-flow requests (for example, to stop sending normal-flow requests or to prepare to shut down communication with the other end of the session). Because of this, VTAM sends an expedited-flow request immediately, before sending any normal-flow traffic that

might be waiting to be sent. Normal-flow traffic is handled separately from the expedited-flow traffic. Figure 30 illustrates how VTAM gives priority to expedited-flow traffic.

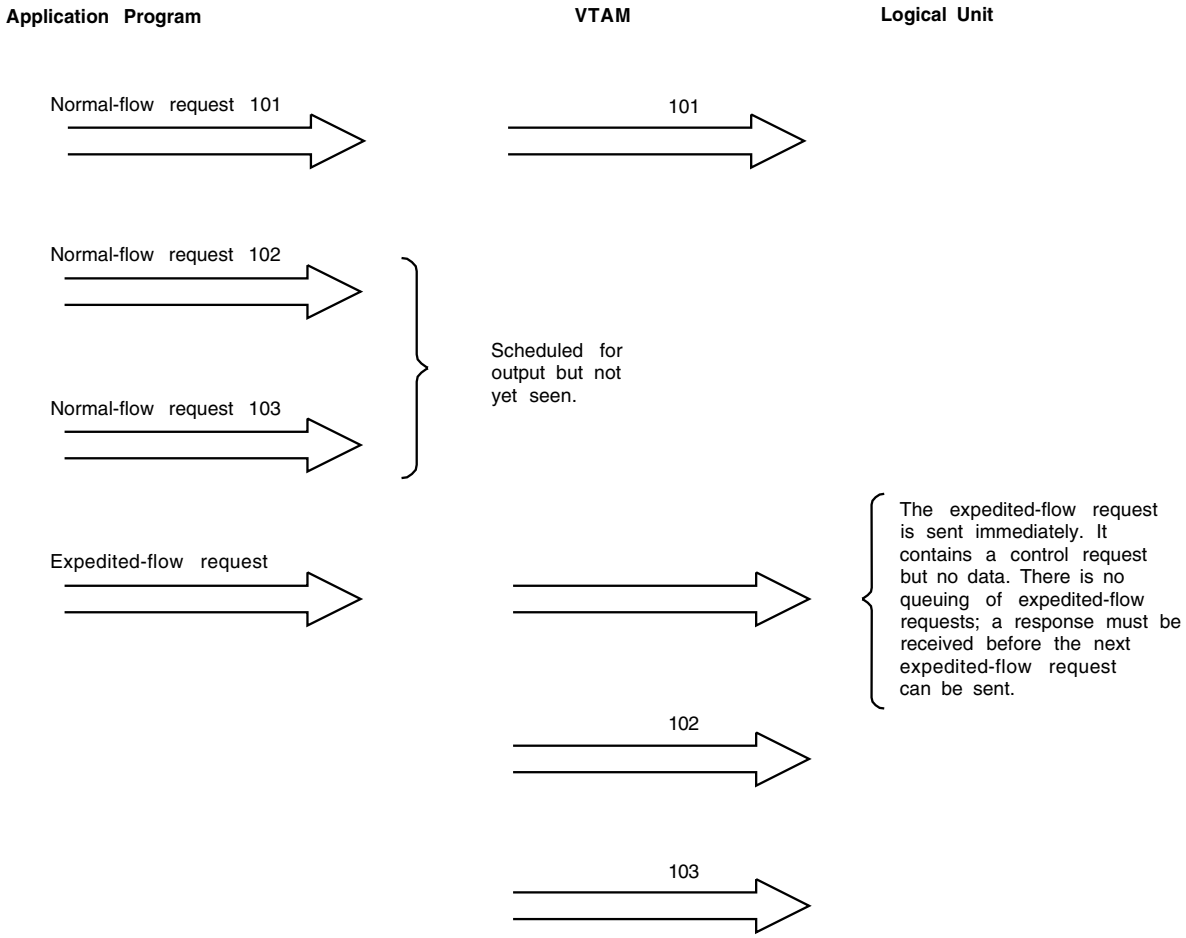


Figure 30. Difference between normal-flow and expedited-flow requests

The requests and responses that are sent on the normal flow and the expedited flow are listed in Table 24 on page 161. Only one expedited-flow data-flow-control request can be sent at a time by each LU in the session; a response must be received to one such request before another can be sent. Similarly, only one session-control request can be sent at a time with the SESSIONC macroinstruction.

DFSYN, DFASY, and RESP types of RUs

VTAM classifies the data and data-flow-control request and response units that can be received by an application program into three types:

- DFSYN RUs, the normal-flow request units for both data and data-flow-control requests
- DFASY RUs, the expedited-flow data-flow-control requests
- RESP RUs, normal-flow response units for both data and data-flow-control requests.

Table 24 on page 161 describes these three types in more detail. Optionally, as described in "Normal operating system environment for a VTAM application program" on page 30, certain normal-flow response units can be treated as DFSYN

RUs instead of as RESP RUs. These special responses are called DFSYN responses. The application program cannot receive expedited data-flow-control responses. These are intercepted by VTAM.

Table 24. Summary of requests and responses transmitted on normal flow and expedited flow

Normal-flow	Expedited-flow
Data	Session Control: BIND CLEAR Request Recovery (RQR) Set and Test Sequence Numbers (STSN) Start Data Traffic (SDT) UNBIND SWITCH
Data Flow Control: BID Bracket Initiation Stopped (BIS) CANCEL CHASE Logical Unit Status (LUSTAT) Quiesce Complete (QC) Ready to Receive (RTR)	Data Flow Control: Quiesce at End of Chain (QEC) Release Quiesce (RELQ) Request Shutdown (RSHUTD) Shutdown (SHUTD) Shutdown Complete (SHUTC) Stop Bracket Initiation (SBI)

This classification of RUs is useful because it allows the application program to have separate routines to handle RUs of each type. On each RECEIVE, the application program can specify which RU type will satisfy the RECEIVE. More than one type can be specified as eligible. Also, exit routines instead of RECEIVE macroinstructions can be used to receive DFASY RUs and to receive RESP RUs. Finally, the three types are handled independently for eligibility to satisfy a RECEIVE OPTCD=ANY macroinstruction, as described in "Receiving input from any session versus from a specific session" on page 174.

For more information about RECEIVE and about the DFASY and RESP exits, see "Explicit RECEIVEs and EXLST exit routines" on page 178.

Controlling the handling of normal-flow responses

The macroinstruction that sends a normal-flow request can be used to control how VTAM handles the response to that request. The ability to exercise that control depends on whether PROC=NORDRESP or PROC=ORDRESP was specified in the NIB when the session was established.

If PROC=ORDRESP was in effect in the NIB when the session was established, the programmer establishes, at the time the normal-flow request is sent, the way in which VTAM handles the response, as follows:

- When the request is sent with RESPOND=NQRESP in the RPL, the response is handled as an ordinary normal-flow response—meaning that it can cause completion of a POST=RESP operation, scheduling of a RESP exit routine, and completion of RECEIVE RTYPE=RESP.
- When the request is sent with RESPOND=QRESP, the response is not handled as a response, but instead is handled almost as if it is an incoming normal-flow request from the LU. Such a response is called a ***DFSYN response***. A DFSYN response does not cause scheduling of a RESP exit routine and does not cause completion of a RECEIVE RTYPE=RESP. It does, however, cause completion of the original SEND operation if the operation specified POST=RESP. If POST=RESP was not specified in the original operation, the application program can get the response by using a RECEIVE RTYPE=DFSYN and testing the RTYPE field of the RPL upon completion. If RTYPE=(DFSYN,RESP) after completion, the program knows it has received a normal-flow response instead of a normal-flow request.

If PROC=NORDRESP was in effect when the session was established, the programmer has no control over how VTAM handles the responses. In this case, all normal-flow responses (regardless of the QRESP setting) are handled as if NQRESP had been specified in the original SEND. Thus, PROC=NORDRESP is specified in the NIB when a user wants the application program to be executed in VTAM.

Additionally, the PROC=NORDRESP or PROC=ORDRESP setting controls how VTAM interprets the RPL POST and RESPOND operands for normal-flow requests.

The NQRESP response and QRESP response differ in that the NQRESP response is handled as a regular normal-flow response and is presented to the application program in sequence with other normal-flow responses; the QRESP response is treated as an incoming normal-flow request and is presented to the application program in sequence with those requests. A response that satisfies a SEND macroinstruction that specifies POST=RESP and either QRESP or NQRESP is always delivered immediately by VTAM. This response could, therefore, get ahead of other normal-flow responses.

An application program sends most of its normal-flow requests with RESPOND=NQRESP. However, you might want to use QRESP with bracket protocol. See "The Chase request" on page 216 for a description of QRESP in relation to the Chase request. See "Special use of RESPOND=QRESP with bracket protocol" on page 216 for a description of the bracket protocols and how they are used with QRESP. All requests of a chain should have the same QRESP or NQRESP setting.

When an application program receives a series of requests and responses on a session and either the NIB specifies PROC=NORDRESP or the requests from the application program specify RESPOND=NQRESP, all of the requests arrive in the same order that they are sent, and all of the responses arrive in the same order that they are sent, but the order of the combination of requests and responses can be changed. For example, request 1 always arrives before request 2, and response 2 always arrives before response 3; however, a response sent after a particular request can arrive before it. As an example, the LU sends the following on a session:

```
Response 1, Request 1, Request 2, Request 3,
Response 2, Response 3, Request 4
```

They could arrive at the application program as follows:

```
Response 1, Response 2, Request 1, Response 3,
Request 2, Request 3, Request 4
```

If the NIB specifies PROC=ORDRESP and the requests from the application program specify RESPOND=QRESP, all the responses (which are now DFSYN responses) and normal-flow requests arrive in the same order in which they are sent. (In effect, VTAM handles a DFSYN response as it does a normal-flow request.) For example, if the LU sends the following:

```
Response 1, Request 1, Request 2, Request 3,
Response 2, Response 3, Request 4
```

They arrive in the same order at the application program as follows:

```
Response 1, Request 1, Request 2, Request 3,
Response 2, Response 3, Request 4
```

When a program sends a normal-flow request on a session established with the NIB PROC=ORDRESP, the POST operand in the macroinstruction can be set to SCHED or RESP, and the completion of the macroinstruction is based on that setting. If NORDRESP is specified in the NIB, when a normal-flow data-flow-control request (as opposed to a data request) is sent, VTAM ignores the POST operand and automatically establishes POST=RESP (meaning that the operation is not completed until the response has been received). Similarly, if ORDRESP is specified, the application program must specify the correct RESPOND setting for normal-flow data-flow-control requests (ordinarily (NEX,FME)). For NORDRESP (unless OPTCD=USERRH), the (NEX,FME) value of RESPOND is assumed automatically by VTAM. For USERRH considerations, see "Relationship to NIB PROC=ORDRESP or NORDRESP operand" on page 199.

Sequence numbers

The transmission services (TS) profile might indicate that the session is to use sequence numbers. If so, each normal-flow request sent by an application program to an LU is assigned a sequence number by VTAM.

The sender knows this number as the outbound sequence number; the receiver knows the number as the inbound sequence number. The numbering begins with 1 for the first normal-flow request sent after BIND and is increased by 1 for each subsequent request. This process continues until the session is terminated. Sequence numbers can be reset during the session. See "Controlling flow" on page 164 for a description of session-control requests. (Also, either the application program or VTAM assigns an identification number to each expedited-flow request sent in a session, but those numbers are handled separately from the normal-flow sequence numbers.)

Similarly, an LU in session with an application program assigns a sequence number to each normal-flow request it sends to the application program. The numbering begins with 1 and is increased by 1 for each subsequent normal-flow request that the LU sends. To the LU, this number is known as the outbound sequence number. To the application program, the number is known as the inbound sequence number. VTAM checks the inbound sequence numbers on the normal-flow requests it receives that are destined for an application program. If a request arrives out of sequence (that is, its sequence number is not 1 greater than that of the last normal-flow request received on the session), VTAM considers this to be a transmission error and indicates to the application program that an out-of-sequence request has been received by passing an exception request to the application program.

When a normal-flow response is sent (either a positive or a negative response), the response sender usually assigns to it the sequence number of the request being responded to. This provides the request sender (the response receiver) a way to match the response with its request. For example, an application program can send a group of requests, with each request indicating that only exception responses should be returned. If the session partner returns a negative response, the application program can use the sequence number to determine where in the group the error occurred. However, for certain situations, the error can be localized only to a chain. Sequence numbers are also useful for LUs that log each request that is received or sent. Figure 31 on page 164 illustrates how sequence numbers are used. Other examples in this chapter show more specific examples of their use.

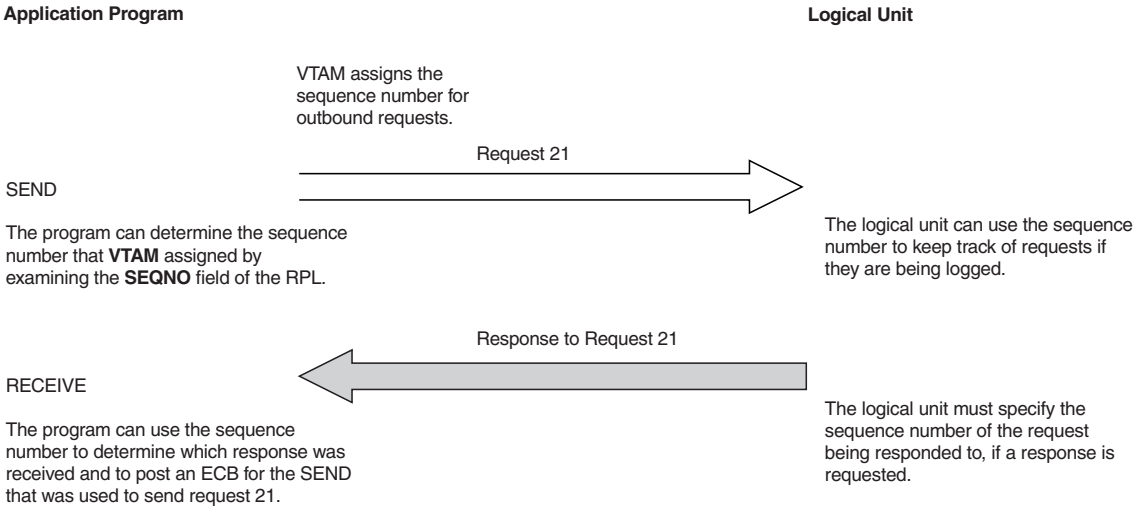


Figure 31. How sequence numbers are used

The SEQNO field of the RPL is used to convey sequence numbers between VTAM and the application program. The application program can determine the sequence number that VTAM assigned to an outbound normal-flow request by checking the SEQNO field after completion of the SEND macroinstruction. For an inbound request or response, the application program determines the sequence number that was contained in the request or response by examining the SEQNO field after completion of the RECEIVE macroinstruction. To assign a sequence number to an outgoing response, the application program puts the sequence number into the SEQNO field before issuing the SEND macroinstruction.

For a description of how sequence numbers are assigned when OPTCD=LMPEO is used, refer to "LMPEO sequence number handling" on page 189. When OPTCD=LMPEO is specified, the OBSQVAL RPL field is also used.

The application program can also assign sequence numbers when sending expedited data-flow-control requests (such as a Signal request). For these requests, VTAM uses the current setting of the SEQNO field of the SEND RPL instead of generating a sequence number. This allows the application program to use the SEQNO field as another data field (for example, to relate a Signal request to a particular bracket). VTAM generates sequence numbers for all session-control requests whether sent by VTAM (such as UNBIND), or by the application program (such as RQR).

In summary, when sending a response or an expedited data-flow-control request, the application program specifies the sequence number. When sending a normal-flow request or a session-control request, VTAM generates the sequence number.

Controlling flow

The Start Data Traffic (SDT) and Clear requests

The PLU can start and stop the flow of all data and data-flow-control requests and responses on a session, if the TS profile indicates that the session supports the Start Data Traffic (SDT) session-control request. In most cases, the flow begins when the PLU sends the SDT at the beginning of a session. Depending on how the SDT field of the NIB is set, the SDT can be sent automatically by VTAM as part of the

OPNDST processing, or it might have to be sent by the application program by using the SESSIONC macroinstruction. The flow of requests and responses is stopped when the PLU sends a Clear session-control request by using the SESSIONC macroinstruction. This not only prohibits any further transmission of such requests and responses, but also causes the sequence numbers of the LU and VTAM to be reset to 0. The first data request or normal-flow data-flow-control request sent is assigned the sequence number 1. Clear also causes all incoming and outgoing data and data-flow-control requests and responses in the network pertaining to the session to be discarded. Any SEND OPTCD=LMPEO request that is being processed for the session when Clear is issued is also terminated prior to normal completion; one or more RUs might have been sent for each SEND OPTCD=LMPEO before it is terminated by Clear. Clear is sent whenever it is needed to stop the flow of data and data-flow-control requests and to clean up traffic flowing in session. (Under some error conditions VTAM automatically sends a Clear.) When Clear is sent by the PLU in the middle of a session, the flow can be restarted with SDT. The flow of requests and responses can be started and stopped any number of times, as illustrated in Figure 32 on page 166.

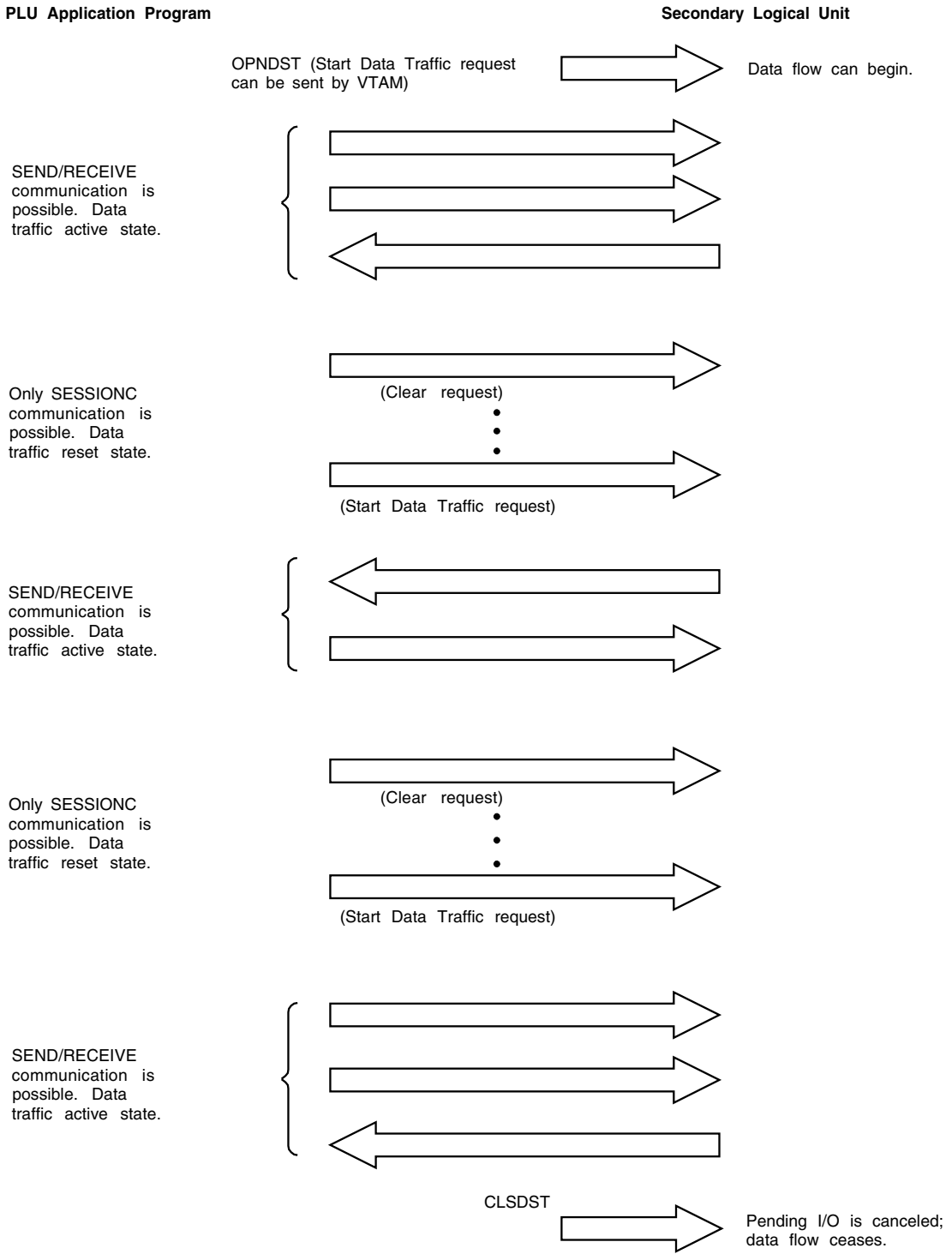


Figure 32. Starting and stopping the flow of requests and responses

For sessions that support SDT, (1) after a session is established and before the first SDT response is sent or received and (2) during the time after Clear is sent or received and before a subsequent SDT response is sent or received, the session is said to be in a data-traffic-reset state. After an SDT response is sent or received, but before Clear is sent or received, the session is in a data-traffic-active state. Requests and responses for data and data-flow-control can be sent only when the session is in a data-traffic-active state.

The Set and Test Sequence Numbers (STSN) and Request Recovery (RQR) Requests

About this task

Another session-control request sent with the SESSIONC macroinstruction is Set and Test Sequence Numbers (STSN). This request allows the PLU to reset the normal-flow sequence numbers and to communicate with the SLU to establish the proper sequence numbers. For example, an attempt to resynchronize sequence numbers can begin when one of the LUs recognizes that the sequence number of a request it has received on the session is not 1 greater than the sequence number of the previous request it received. When the SLU recognizes the sequence number error, it sends the Request Recovery (RQR) session-control request to ask the PLU to take recovery action. When the VTAM application program PLU receives the request, its SCIP exit routine is scheduled.

Another use of STSN is for restarting request flow, where the PLU that periodically checks normal-flow requests that it sends to an SLU on a session wants to inform the SLU of the sequence numbers at which it is restarting after a system, session, application program, or LU failure.

Procedure

A VTAM application program acting as the PLU in a session normally uses the following procedure to resynchronize sequence numbers with the SLU:

1. The PLU issues SESSIONC CONTROL=CLEAR to stop the request/response flow and to remove all requests not delivered and all responses pertaining to its session.
2. The PLU then issues SESSIONC CONTROL=STSN to question the LU about normal-flow sequence numbers on the session. With this macroinstruction, the PLU can do one of the following tasks:
   - Send sequence number values to the SLU and, from the response, determine whether the SLU "agrees" with those numbers
   - Request that the SLU return whatever values it considers to be the correct sequence numbers.
   - Tell the SLU to set its sequence numbers to particular values.

   To reach agreement with the SLU, the PLU might have to send several STSN requests, with the SLU responding to each request. When agreement is finally reached, either session partner or both, might have to return to a previous point in their operations and resend one or more normal-flow requests.
3. After agreement on sequence numbers is reached, the PLU issues SESSIONC CONTROL=SDT to restart the flow of requests and responses.

Results

For examples of the use of SESSIONC CONTROL=STSN, see Figure 109 on page 696 and Figure 119 on page 706. For details about the options available with STSN, refer to "SESSIONC—Send a session-control request or response" on page 532.

Data-flow-control requests and indicators

SNA defines protocols for controlling the flow of data within a session (for example, specifying which end of the session can send data at a particular time). Data-flow-control requests and indicators are used in these protocols, some of which are summarized in "Using SNA protocols" on page 201. These requests and indicators can be specified with a SEND macroinstruction and are made available

with a RECEIVE macroinstruction, or in an exit routine. For additional information, refer to the SEND and RECEIVE macroinstruction descriptions in Chapter 13, "Conventions and descriptions of VTAM macroinstructions," on page 371.

Identifying LUs and sessions

After a session has been established, the application program has both the communication identifier (CID) that identifies the session and the symbolic name of the LU (the session partner). For details, refer to the sections on the LOGON and SCIP exit routines and on the OPNDST and OPNSEC macroinstructions in Chapter 5, "Establishing and terminating sessions with logical units," on page 81.

The CID must be specified in the RPL for all communications directed to a particular session.

When a RECEIVE macroinstruction issued in the any-mode (described in the following) is completed, VTAM provides the CID of the session on which the request or response is received. If the application program requires the LU's symbolic name, the application program has three ways to relate the CID to the LU's symbolic (user-supplied) name:

- The application program can use an INQUIRE OPTCD=CIDXLATE macroinstruction to translate the CID into a symbolic name.
- The application program can maintain a table of CIDs and their symbolic equivalents. Applications that are enabled for persistence should be capable of rebuilding this table during recovery. This is because the CID of a particular session is different after VTAM recovers.
- When the application program establishes a session with the LU, the application program can initially assign a 4-byte value to the session (by putting the value in the USERFLD field of the NIB), and VTAM returns the value each time that session's data satisfies RECEIVE. The 4-byte value can be anything the application program chooses to associate with the session. For example, it can be used to identify the session (and thus the LU), or it can contain the address of a subroutine that is to handle that session's data.

Using VTAM to communicate with LUs

Using VTAM to communicate with LUs requires an understanding of these major alternatives:

- VTAM can perform an operation synchronously or asynchronously with respect to execution of the VTAM application program (OPTCD=SYN or ASY on SEND or RECEIVE).
- For asynchronous operations, VTAM can post an ECB or schedule an exit routine when the operation completes (OPTCD=ASY and either ECB=*address* or EXIT=*address* are specified in the SEND or RECEIVE macroinstruction).
- VTAM can schedule a request to be sent or can send a request and confirm its arrival (SEND POST=SCHED or RESP).
- Input from any session can satisfy RECEIVE, or input from a specific session can satisfy RECEIVE (RECEIVE OPTCD=ANY or SPEC).
- A session can be in continue-any (CA) mode or in continue-specific (CS) mode (OPTCD=CA or CS on certain RPL-based macroinstructions).
- RECEIVE RTYPE=DFASY can be satisfied or a DFASY exit routine can be scheduled when an expedited-flow (DFASY) request is received.
- RECEIVE RTYPE=RESP can be satisfied, or an RESP exit routine can be scheduled when a response is received.

- The application program request can be notified of responses queued.
- VTAM can retain or discard portions of an incoming request that is too long to fit in the program's input area (PROC=KEEP or TRUNC).
- VTAM can split any function management (FM) data that exceeds the maximum RU size into a chain (or partial chain).
- VTAM can send FM data from a number of discontiguous buffers.
- VTAM can refrain from enforcing particular RH indicator settings related to data flow control and FM data.

Major alternatives

Some of these alternatives are also discussed in Chapter 3, "Organizing an application program," on page 33. Here they are discussed specifically in relation to communicating on a session.

Synchronous versus asynchronous operations:

Synchronous requests:

A VTAM application program can specify that a communication operation be performed synchronously with respect to the execution of the program. For example:

```
SEND  RPL=(2),STYPE=REQ,AREA=AREA1,RESPOND=(NEX,FME),         C
      OPTCD=SYN,POST=SCHED
```

This SEND specifies that a request (STYPE=REQ) be sent from AREA1 and that a definite response be returned whether or not the request arrives and is processed successfully (RESPOND=(NEX,FME)). Execution of the VTAM application program (or at least execution of the task or SRB from which the macroinstruction is issued) is suspended because the application program has specified OPTCD=SYN, and the next instruction is not executed until VTAM has determined that the requested operation has been performed. In this case, however, the requested operation is the scheduling of a SEND (POST=SCHED) rather than the actual transmission. Certain circumstances can delay the actual scheduling. For example, the scheduling can be delayed until the LU has returned a session-level pacing response indicating that it is ready to receive the next request on the session. Another example is scheduling being delayed while waiting for a virtual-route-pacing response for the route associated with the session. The ASY option is preferable because it will not cause the application program to be delayed.

Here is another example of a synchronous SEND:

```
SEND  RPL=(2),STYPE=REQ,AREA=AREA1,RESPOND=(NEX,FME),         C
      OPTCD=SYN,POST=RESP
```

For this SEND, the VTAM application program has to wait until VTAM receives a response to the request (POST=RESP). A program that communicates on a few sessions and waits for each communication request to be completed before doing any further processing might use this kind of synchronous operation; for most programs, however, this is not efficient.

POST=RESP cannot be specified unless a definite response is requested; that is, no-response or exception-response-only cannot be specified with POST=RESP, because VTAM would never know that the request had arrived at the LU.

Here is an example of a RECEIVE for input on a specific session with OPTCD=SYN:

```
RECEIVE RPL=(2),RTYPE=DFSYN,AREA=AREA1,AREALEN=100,          C
        OPTCD=(SYN,SPEC)
```

Here, execution of the VTAM application program is suspended until input arrives on the session (whose CID is located in the RPLARG field). This is efficient only in simple programs where batch input is being received, or in programs where a request is known to be in VTAM buffers. If a request received in VTAM's buffers is larger than the amount of data read each time a RECEIVE is issued, the KEEP option (described in "Handling overlength input data" on page 182) is used.

Here is an example of a RECEIVE for input from any session with OPTCD=SYN:

```
RECEIVE RPL=(2),RTYPE=DFSYN,AREA=AREA1,AREALEN=200,          C
        OPTCD=(SYN,ANY)
```

Here, execution of the VTAM application program is suspended until input arrives on any session that is not in the CS mode. This type of request is most likely to be used in a program that communicates on a few sessions. It can also be used with a large number of sessions if response time is not important.

Asynchronous requests:

A VTAM application program can also request that a communication operation be performed asynchronously with respect to the execution of the program. For example:

```
SEND  RPL=(2),AREA=AREA1,STYPE=REQ,RESPOND=(NEX,FME),        C
      OPTCD=ASY,POST=SCHED,ECB=ECB1
```

This SEND requests that VTAM schedule the sending of the data from AREA1 and immediately return control to the program. As soon as scheduling of the output has been completed, VTAM notifies the program either by posting an ECB (shown here) or by scheduling an RPL exit routine. (The relative advantages of posting ECBs and scheduling RPL exit routines are discussed in Chapter 3, "Organizing an application program," on page 33, and in "ECBs versus RPL exit routines.") The actual sending of a request can be requested to another type of asynchronous request. For example:

```
SEND  RPL=(2),AREA=AREA1,STYPE=REQ,RESPOND=(NEX,FME),        C
      OPTCD=ASY,POST=RESP,EXIT=RPLEXIT
```

This SEND specifies that VTAM begin sending the request at AREA1 and immediately return control to the program. When VTAM receives a response indicating the success or failure of the transmission and processing, VTAM schedules an RPL exit routine at RPLEXIT. The program continues processing; the RPLEXIT exit routine automatically gets control when this operation is completed. Or, if ECB-posting is specified instead of the exit routine, the program continues processing (minus the time VTAM takes to get control and post the ECB) until it discovers the ECB is posted or until the program issues a WAIT or a CHECK macroinstruction.

While synchronous operations are easier to program, they are inefficient with regard to the amount of processing that the program can do. Asynchronous operations are more difficult to program, but are required to handle communication with a reasonably large number of sessions.

ECBs versus RPL exit routines:

If asynchronous operations are requested, each macroinstruction can specify that VTAM do either of two things when the operation is completed: post an ECB or schedule an RPL exit routine.

Here is an example of a SEND macroinstruction that specifies that an ECB be posted upon completion:

```
SEND  RPL=(2),AREA=AREA1,STYPE=REQ,RESPOND=(NEX,FME),          C
      OPTCD=ASY,POST=RESP,ECB=ECB1
```

Figure 33 shows the sequence of events that might occur following the issuance of this macroinstruction.

Here is an example of a SEND macroinstruction that specifies that an RPL exit routine be scheduled upon completion:

```
SEND  RPL=(2),AREA=AREA1,STYPE=REQ,RESPOND=(NEX,FME),          C
      OPTCD=ASY,POST=RESP,EXIT=RPLEXIT
```

Figure 34 on page 172 shows the sequence of events that might occur following the issuance of this macroinstruction.

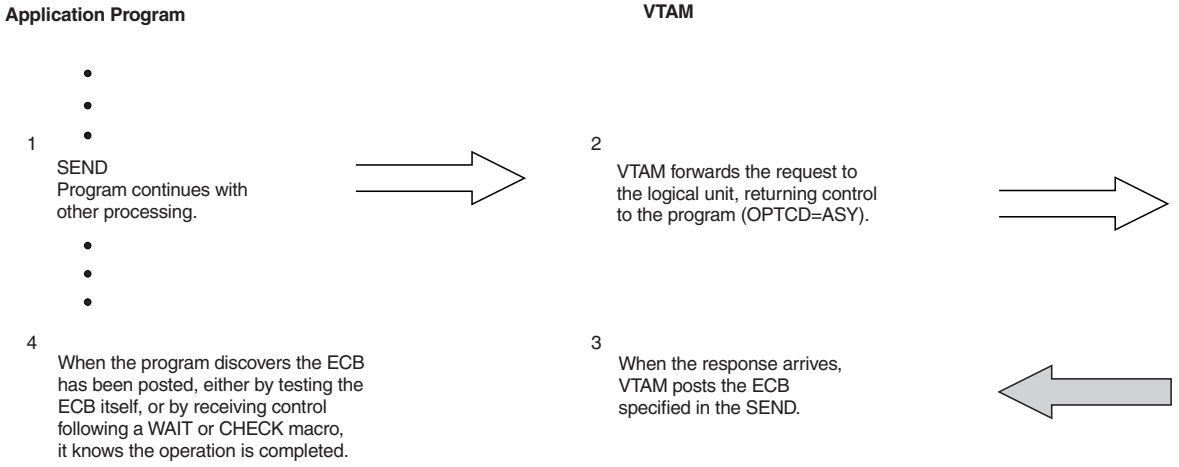


Figure 33. General sequence of events when ECB-posting is specified

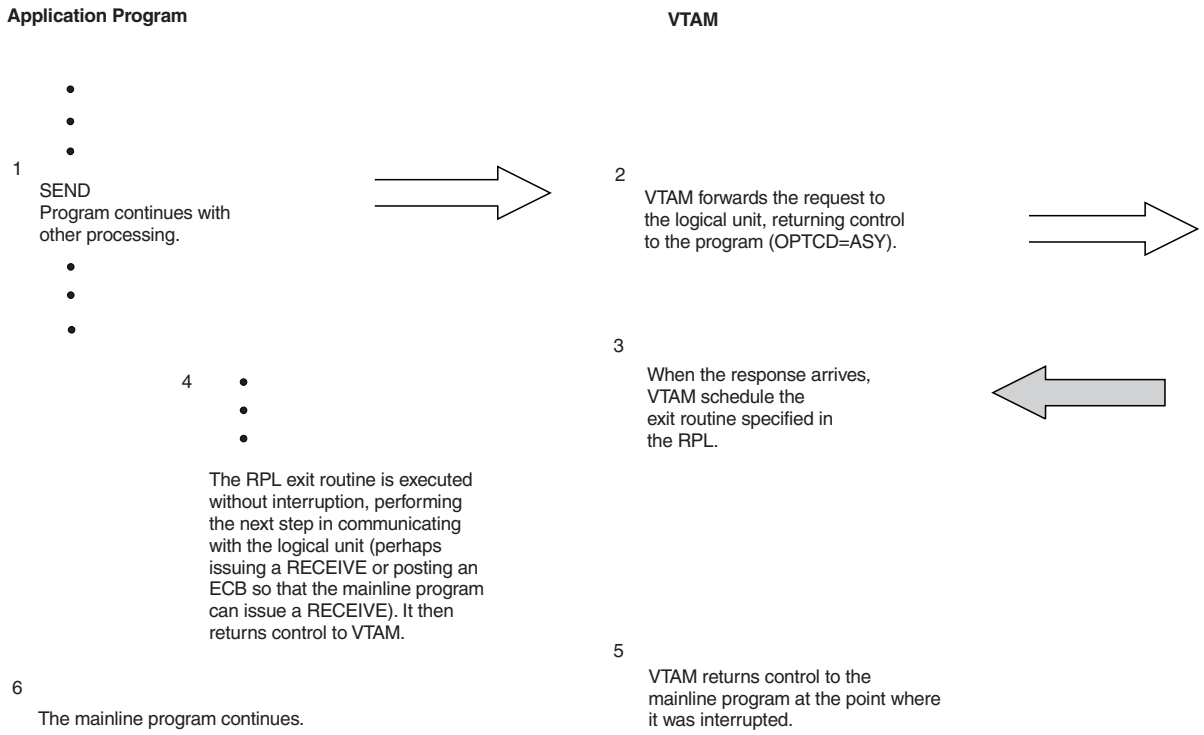


Figure 34. General sequence of events when an RPL exit routine is specified

Scheduled versus responded output operations:

The VTAM application program specifies the sending of a request to an LU in one of two ways:

- The application program can indicate that as soon as the request has been scheduled for transmission and transferred to a VTAM buffer area, thus freeing the application program's output data area, VTAM is to consider the output operation completed. This is called scheduled output and is illustrated in Figure 35 on page 173.
- The application program can indicate that VTAM is not to consider the operation completed until the request has been received by the LU and a response has been returned. This is called responded output and is illustrated in Figure 36 on page 174.

Note: This alternative is also discussed as an example in "Synchronous versus asynchronous operations" on page 169. Certain details are given under the POST operand in the description of the SEND macroinstruction, refer to "SEND—Send output on a session" on page 514.

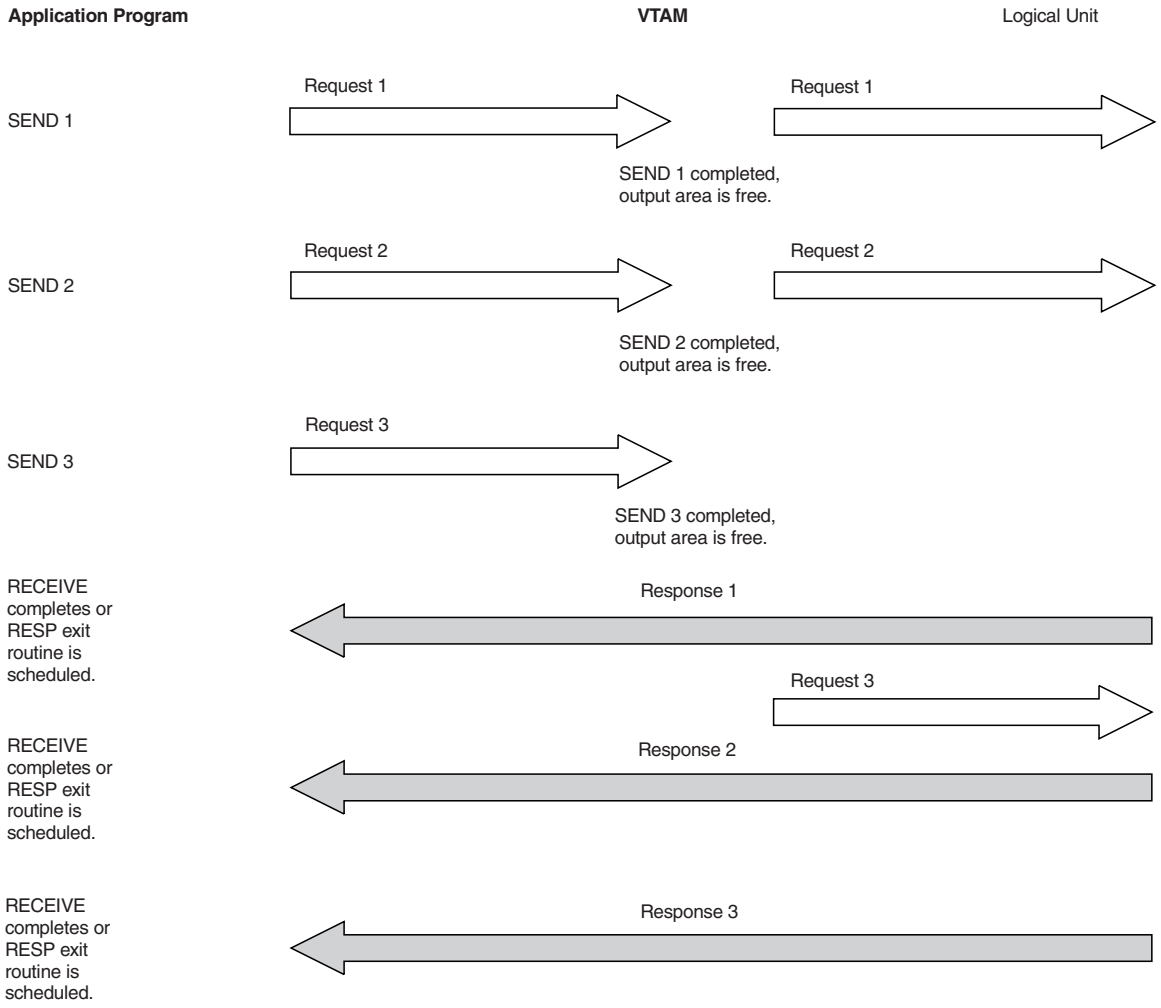


Figure 35. Scheduled output

Responded output is easier to use, but requires that the output data area not be reused until a response has been received by VTAM. If the response indicates that an error occurred, the data is still available for retransmission. Scheduled output allows the application program to send a series of requests that all use the same RPL and, possibly, the same output area. It also allows the program to decide whether a response to the request must be returned. If request chaining is used (see "Chaining" on page 201), a definite response is not required for every request that is sent.

With responded output, response information is returned as part of the operation. With scheduled output, the operation is completed when the request is scheduled, before any response information is available. To determine how the output was processed, the application program can issue an input request to obtain a response. This is why the application program in Figure 35 issues three input requests in addition to the three output requests. Alternatively, responses can be received through a response exit routine.

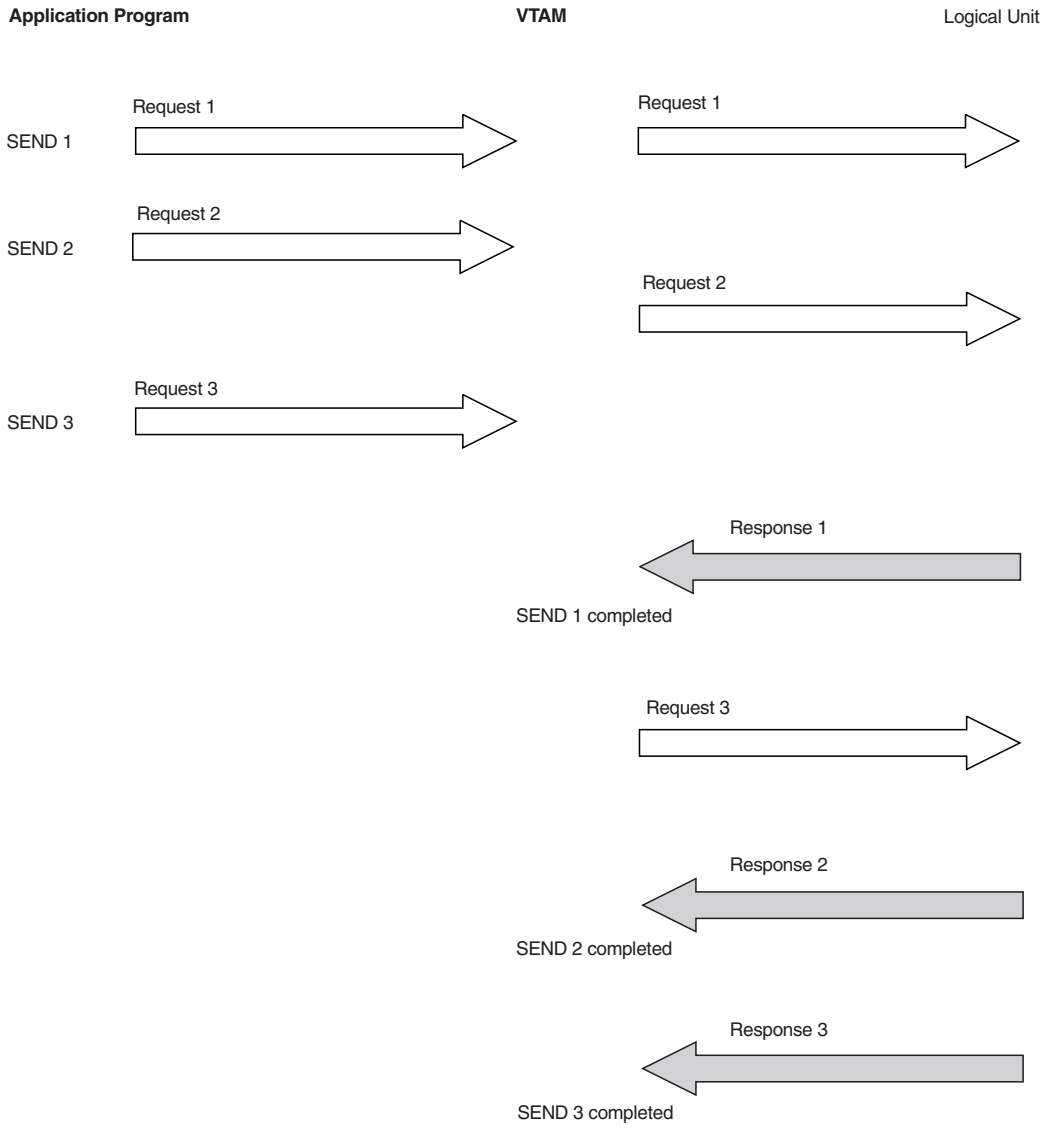


Figure 36. Responded output

Receiving input from any session versus from a specific session:

The VTAM application program can ask for input from a specific session or it can ask for input from any one of its sessions. The application program designates the desired mode (specific or any) with each RECEIVE macroinstruction. These two modes are called, respectively, the specific-mode, and the any-mode. The two modes apply independently to DFSYN, DFASY, and RESP types of input. See "DFSYN, DFASY, and RESP types of RUs" on page 160. The discussion and examples in this and the following sections apply to data requests (DFSYN). However, the specific-mode and any-mode also apply to the normal-flow data-flow-control requests (DFSYN) and to the DFASY and RESP types of input.

In general, an application program initially asks for input on a session in the any-mode, and then communicates on the session in the specific-mode until the transaction, inquiry, or conversation is completed. While communication proceeds on one session, the application program keeps a RECEIVE macroinstruction (issued

in the any-mode) pending so that a new transaction, inquiry, or conversation can be handled from another session while the previous ones continue in specific-mode.

A disadvantage of the any-mode is that the application program does not know the identity of the session until RECEIVE completes. Because the session is initially unknown, the amount of incoming data might also be unknown. This means that the application program must either reserve an input area large enough to hold the largest possible amount of incoming data or execute additional instructions to handle overlength data. The advantage of the any-mode is that it allows the application program to use just one input area for data from all of its sessions rather than using a separate input area for each of its sessions.

With the specific-mode, the application program must specify the identity of the session supplying the data. Because the identity of the session is known, the size of the input data is more predictable than with any-mode. However, because any given session might not supply data for some time, the application program might have to contend with unused data areas. The simplest way to avoid this problem is not to issue RECEIVE requests in the specific-mode unless data has already arrived in VTAM's buffers or is expected to arrive in a relatively short time.

The application can more efficiently manage input data areas by using a combination of specific-mode and any-mode. As an example, consider an application program that obtains an inquiry from any of its LUs, handles that inquiry with a series of SEND and RECEIVE macroinstructions, and then obtains a new inquiry. Part of such a program is illustrated in Figure 37 on page 176.

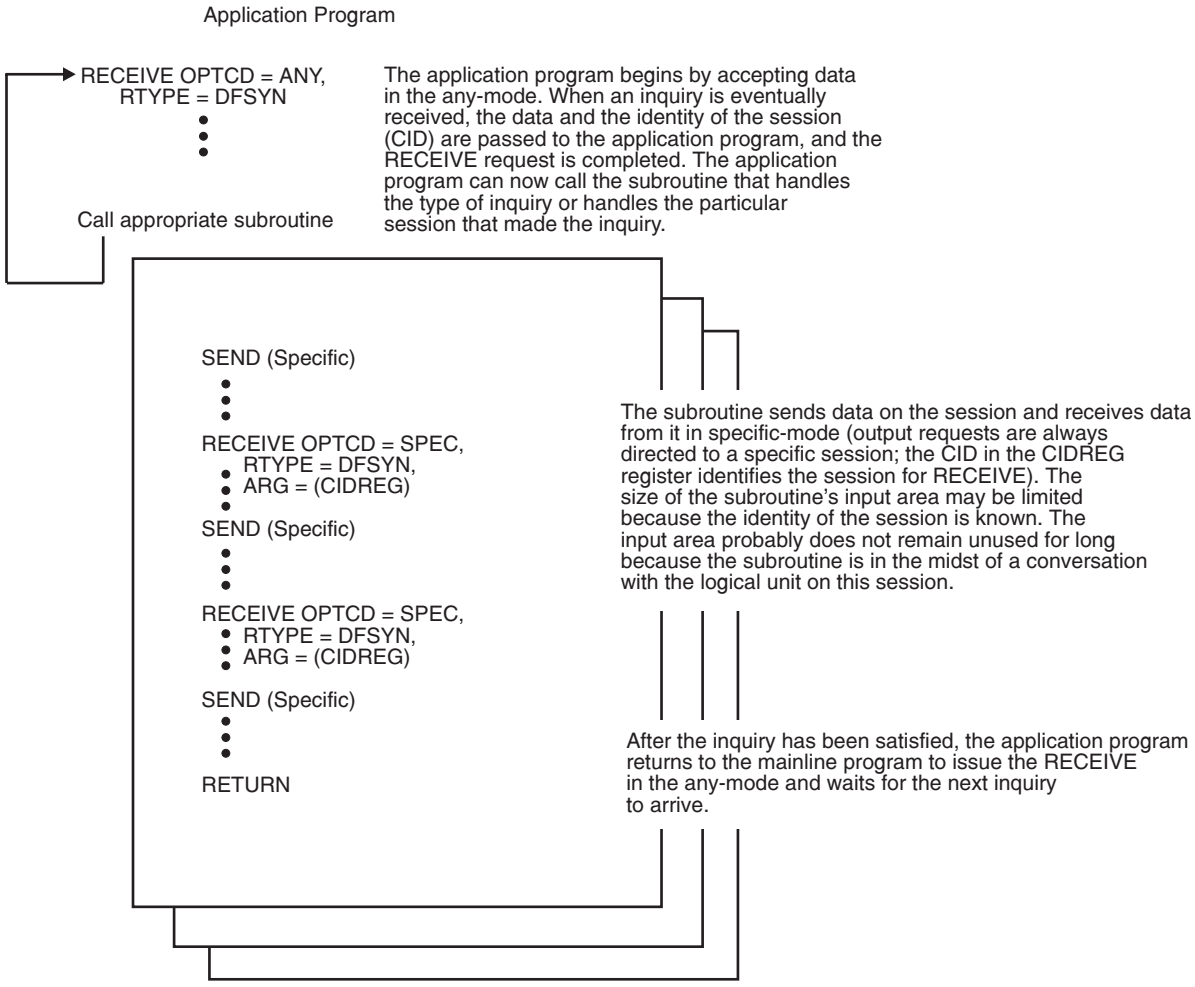


Figure 37. Example of using any-mode and specific-mode to handle an inquiry on a session

Continue-any mode versus continue-specific mode:

In the example in Figure 37, the communication macroinstructions are issued synchronously. The application program handles each inquiry serially, never accepting a new inquiry until it has completed the previous one. Although this procedure might be suitable for application programs that deal with short inquiries and a few sessions, most application programs require handling inquiries in parallel.

An application program that handles more than one inquiry concurrently can use asynchronous request handling and issue new RECEIVEs in the any-mode before the previous inquiry is completed. For an example, see Sample Program 2 in Chapter 16, "Logic of a more complicated application program," on page 605. This, however, raises the possibility that both a RECEIVE for a specific session and a RECEIVE for any session (which includes the specific session as well) might be waiting for data at the same time. Consequently, data that is meant to satisfy the subroutine's RECEIVE might instead be intercepted by RECEIVE in the mainline program, which is meant only to receive new inquiries.

To eliminate this sort of problem, VTAM allows the application program to indicate when a particular session's input can be received by a RECEIVE

macroinstruction issued in the any-mode, and when the input must be received by a RECEIVE macroinstruction issued in the specific-mode. The former is called continue-any mode (CA), and the latter is called continue-specific mode (CS). The desired mode for a session is designated when a communication macroinstruction is issued, but does not become effective until the operation is completed.

Although the CA-CS option code affects only RECEIVE operations, you can switch a session from one mode to the other by specifying the CA or CS option code in any OPNDST, OPNSEC, SEND, RECEIVE, or RESETSR macroinstruction for the session. The change from one mode to another is effective for the next communication operation on the session after this macroinstruction completes, not when the macroinstruction itself is executed. The session that is the object of the macroinstruction is the one whose CA-CS mode is changed. (For RECEIVE OPTCD=ANY, the session whose mode is changed is the one whose input is received by the RECEIVE operation.) If an error occurs and a macroinstruction that specifies a change in a session's CA-CS mode is not completed successfully (that is, (RTNCD,FDB2) does not equal (X'00',X'00'), (X'04',X'03') or (X'04',X'04')), the mode is not changed.

Continue-any and continue-specific modes can be set individually for the three types of input. For example, a session can be placed in a continue-specific mode for DFSYN RUs while it is in continue-any mode for DFASY and RESP RUs. The RTYPE operand on the macroinstruction specifies which type of input RU is to have its continue-mode changed. Any combination of input types, including NDFSYN, NDFASY, and NRESP (meaning no change is to be made), is valid.

Figure 38 on page 178 illustrates how the various modes described in the preceding section relate to one another.

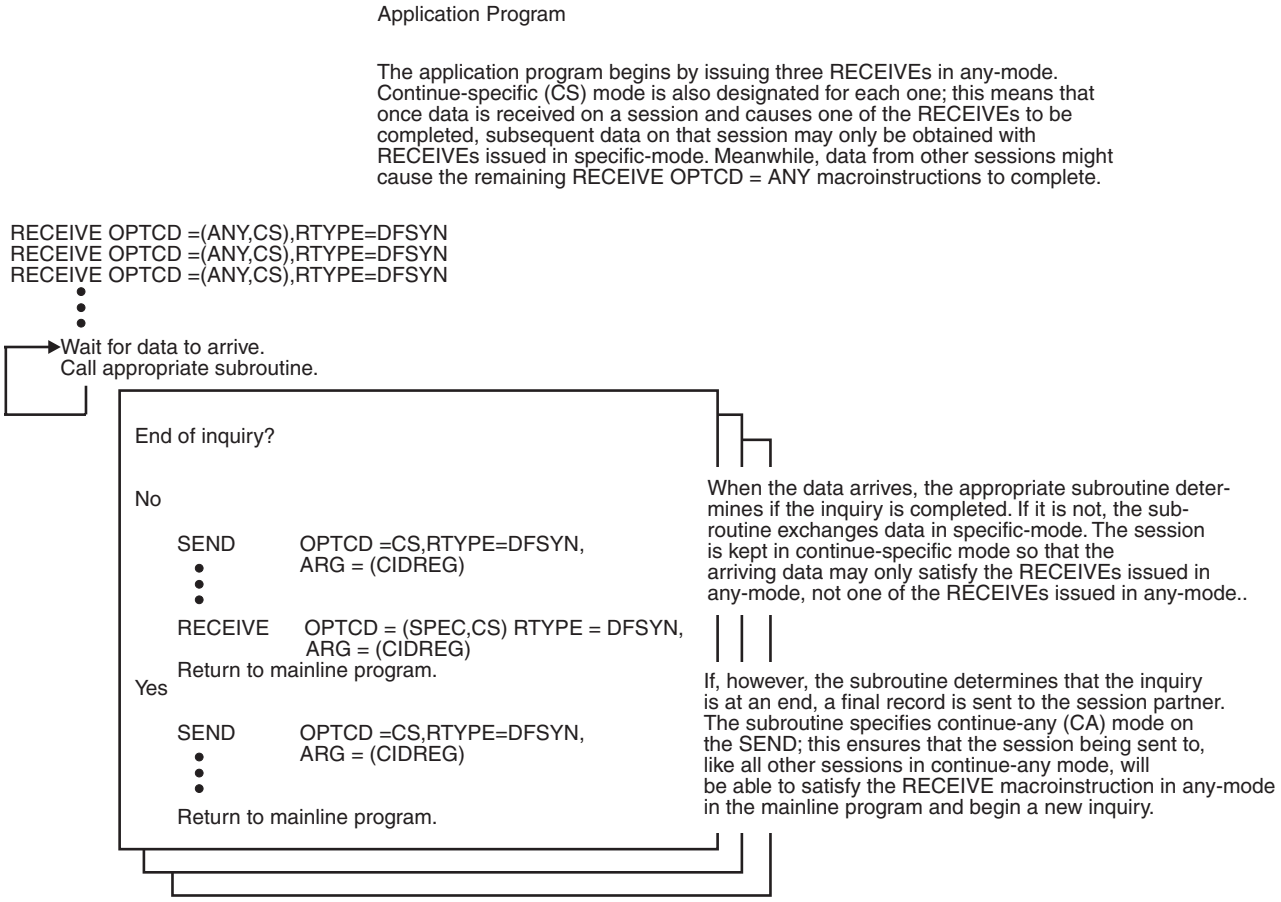


Figure 38. Example of using continue-any and continue-specific modes to handle concurrent inquiries

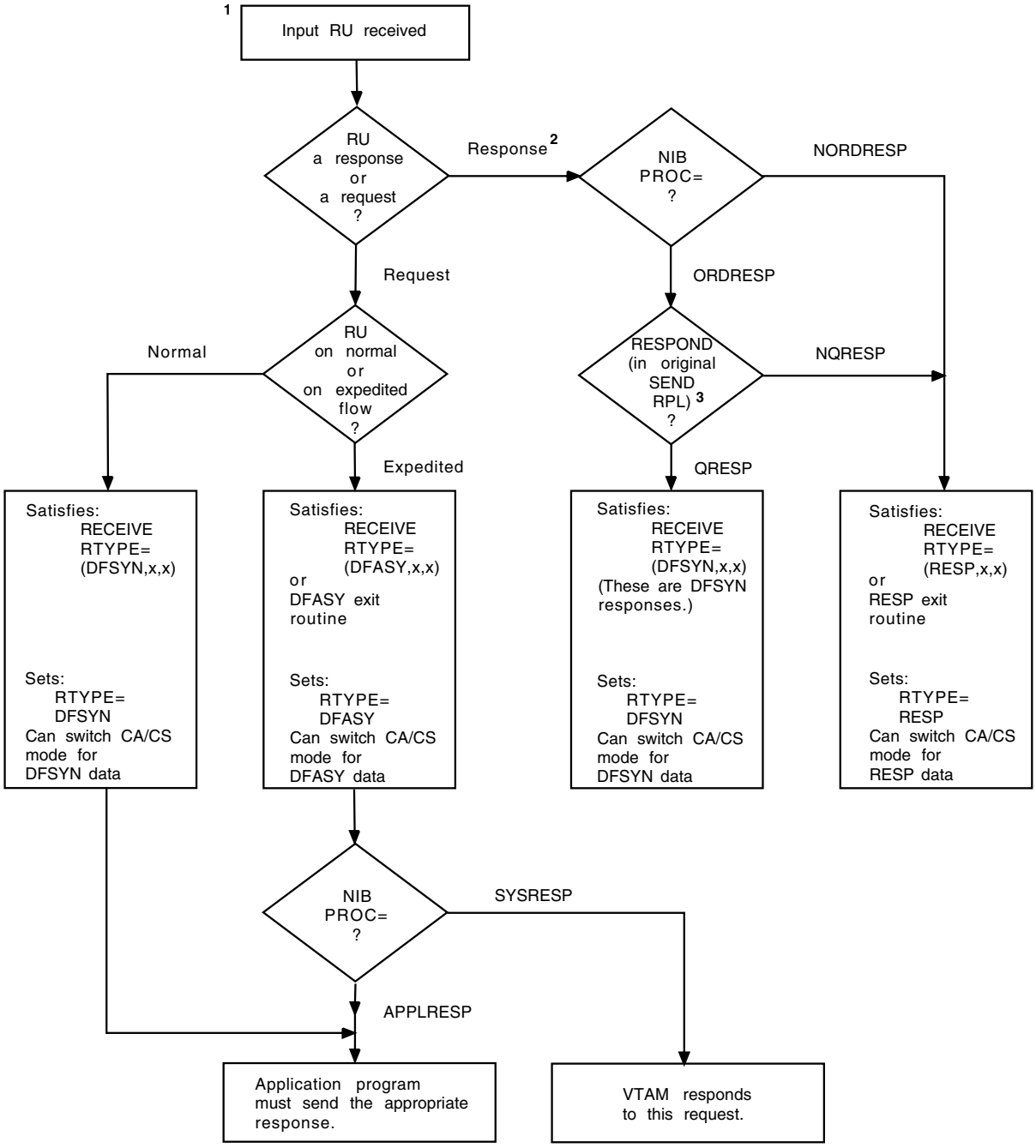
Explicit RECEIVEs and EXLST exit routines:

A VTAM application program can receive expedited-flow data-flow-control requests (for example, a Quiesce at End of Chain request) or responses in a number of ways. A RECEIVE can specify RTYPE=DFASY (for data-flow-control requests) or RTYPE=RESP (for responses) or both. In addition, data input can complete the same RECEIVE (for example, RTYPE=(DFSYN,DFASY,RESP) can be specified). When the RECEIVE is posted complete, the program examines the RTYPE field of the RPL to determine which kind of input was received and branches to an appropriate routine. Alternatively, RECEIVEs can be used only for normal-flow requests, and the addresses of the special input routines can be designated (the DFASY and RESP exit routines) in an EXLST macroinstruction to handle responses and expedited-flow data-flow-control requests.

Using exit routines requires execution of more system instructions than checking the RTYPE field. On the other hand, using RECEIVE requires the use of an RPL to await the input. See Figure 39 on page 179 through Figure 42 on page 182 for the detailed logic used by VTAM to classify input RUs, to complete RECEIVEs, and to schedule EXLST exit routines.

Note: If an application program issues multiple RECEIVE OPTCD=ANY macroinstructions that can be satisfied by a given input RU, it is unpredictable which particular RECEIVE macroinstruction is posted complete with the RU. A similar statement can be made if multiple RECEIVE OPTCD=SPEC

macroinstructions are issued for a session and then an RU arrives that could satisfy any one of them.



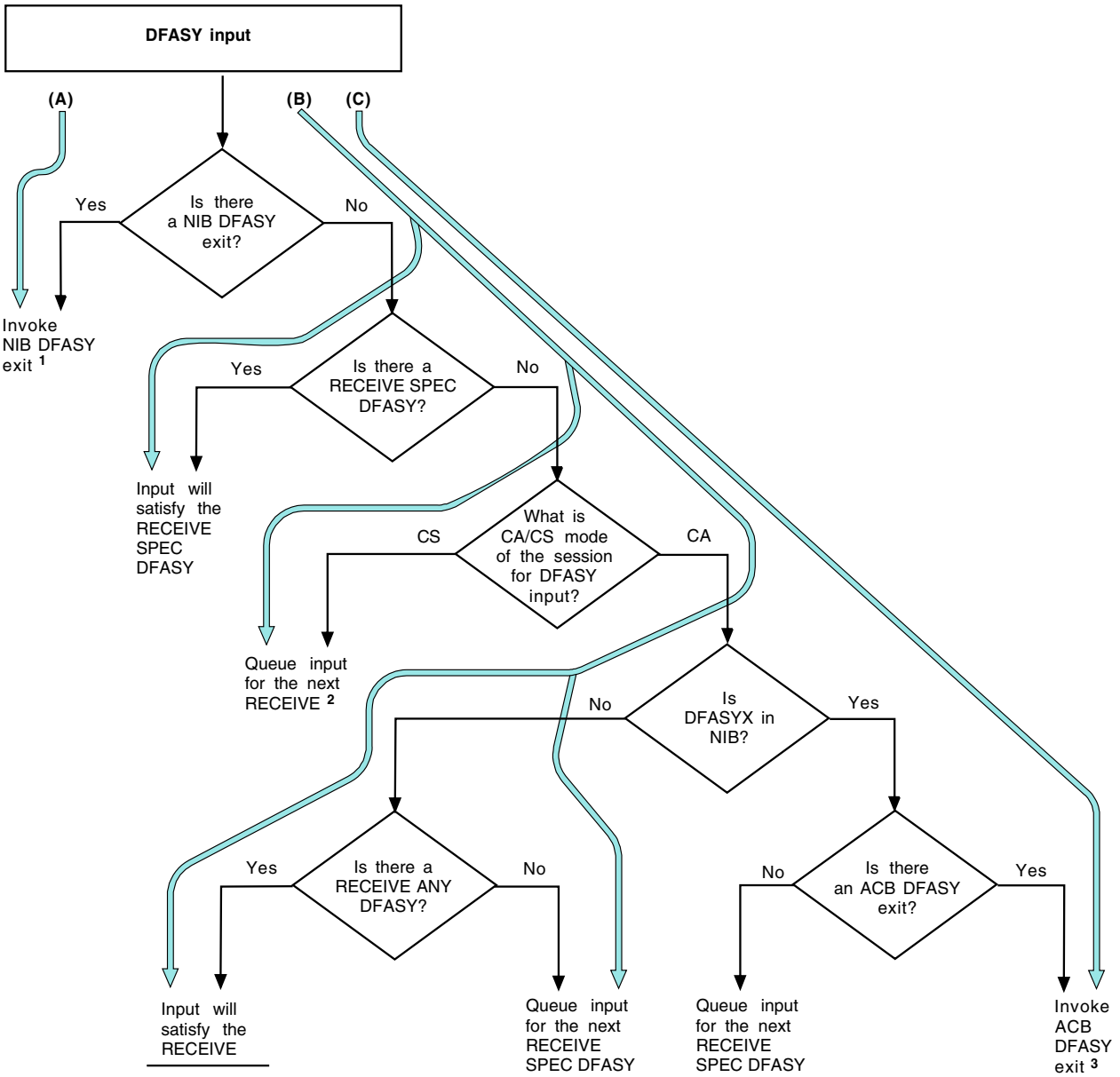
1 If multiple types of input RUs that could satisfy a RECEIVE are concurrently queued when the RECEIVE is issued, the following order is used to attempt to match the RECEIVE with input: DFASY, RESP, and then DFSYN.

2 Because VTAM intercepts and processes all expedited-flow responses, the application program only receives normal-flow responses. If the application program has for the response an associated pending SEND POST=RESP macroinstruction specified, the response information is set in the SEND's RPL fields and does not satisfy any RECEIVE macroinstruction or cause entry to any RESP exit routine.

3 When returning a response, the other end of the session should return the same value (QRESP or NQRESP) that was contained in the original request.

Figure 39. How input RUs are classified by VTAM

For DFASY input, see also Figure 39 on page 179.



(A) Logic flow if NIB exit technique is used.
(B) Logic flow if RECEIVE technique is used.
(C) Logic flow if ACB exit technique is used.

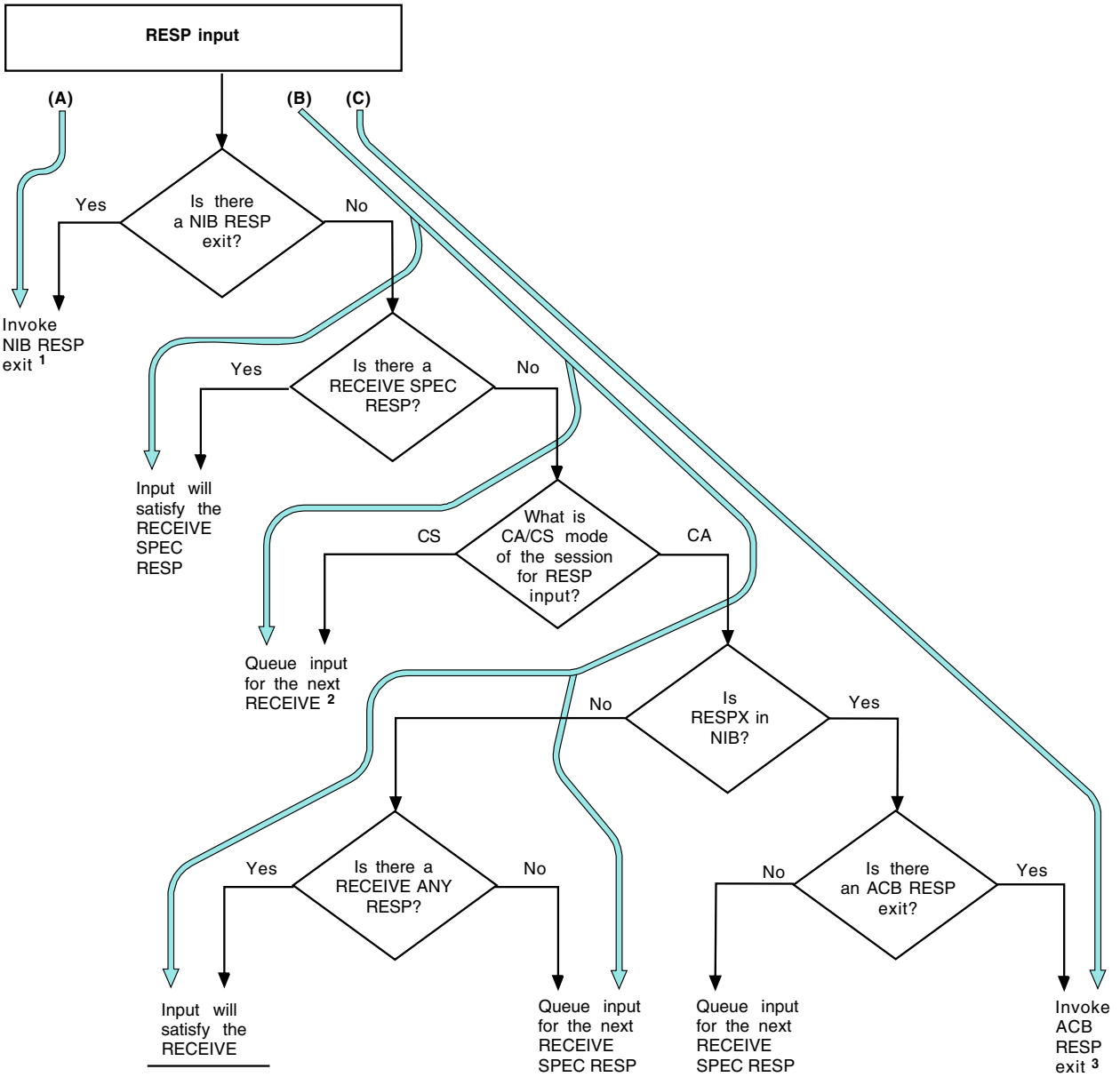
[1] The exit routine is entered if no other exit routine (including the NIB DFASY exit routine) is currently running. If another exit routine is running, the input is queued for the NIB DFASY exit routine. This queued input cannot satisfy a RECEIVE.

[2] The input will satisfy a RECEIVE SPEC DFASY. The response can also be obtained by a RECEIVE ANY DFASY (only if NDFASYX is specified in the NIB) if the mode is switched to CA mode for DFASY input on the session. The ACB DFASY exit is not scheduled if the session mode is changed to CA.

[3] The exit routine is entered if no other exit routine (including the ACB DFASY exit routine) is currently running. If another exit routine is running, the input is queued for the ACB DFASY exit routine. This queued input cannot satisfy a RECEIVE.

[4] If the session mode for DFASY input is subsequently changed to CS mode, only RECEIVE SPEC DFASY can be satisfied.

Figure 40. How VTAM handles DFASY (expedited-flow data-flow-control request) input



(A) Logic flow if NIB exit technique is used.
(B) Logic flow if RECEIVE technique is used.
(C) Logic flow if ACB exit technique is used.

1 The exit routine is entered if no other exit routine (including the NIB RESP exit routine) is currently running. If another exit routine is running, the input is queued for the NIB RESP exit routine. This queued input cannot satisfy a RECEIVE.

2 The input will satisfy a RECEIVE SPEC RESP. The response can also be obtained by a RECEIVE ANY RESP (only if NDFASYX is specified in the NIB) if the mode is switched to CA mode for RESP input on the session. The ACB RESP exit is not scheduled if the session mode is changed to CA.

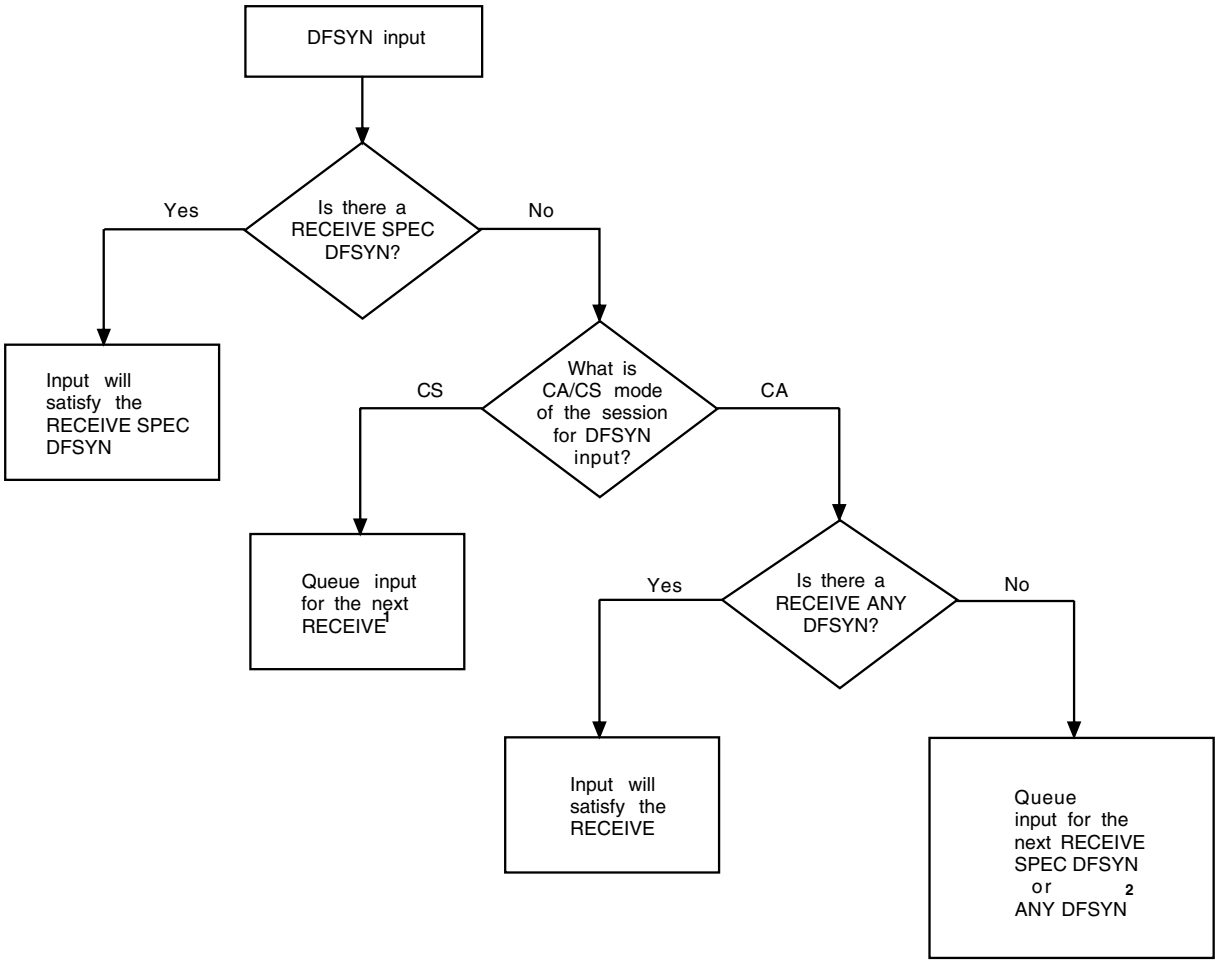
3 The exit routine is entered if no other exit routine (including the ACB RESP exit routine) is currently running. If another exit routine is running, the input is queued for the ACB RESP exit routine. This queued input cannot satisfy a RECEIVE.

4 If the session mode for RESP input is subsequently changed to CS mode, only RECEIVE SPEC RESP can be satisfied.

Figure 41. How VTAM handles RESP (normal-flow response) input

For RESP input, see Figure 39 on page 179.

For DFSYN input, see Figure 39 on page 179.



[1] The input will satisfy a RECEIVE SPEC DFSYN. The request or DFSYN response can also be obtained by a RECEIVE ANY DFSYN if the mode is switched to CA mode for DFSYN input on the session.

[2] If the session mode for DFSYN input is subsequently changed to CS mode, only RECEIVE SPEC DFSYN can be satisfied.

Figure 42. How VTAM handles DFSYN (normal-flow request and DFSYN response) input

Queued response notification:

The application might specify that when SEND completes, the RPL indicates whether any responses are queued. If SEND OPTCD=RSPQUED is specified, VTAM sets RPLRSPNM if any responses are on the normal-flow inbound-response queue, and RPLRSPQR if any responses are on the normal-flow inbound-data queue. The application must use DSECT references to examine the RPL flags named in the preceding section to determine if there are any queued responses.

Note: Responses to SEND requests specifying RESPOND=QRESP are put in the normal-flow inbound-data queue.

Handling overlength input data:

When an application program issues a RECEIVE macroinstruction, the length of the incoming data is often unpredictable. As noted earlier, this is particularly true of RECEIVE macroinstructions issued in the any-mode. VTAM provides two ways of handling data that is too large for the input area:

- VTAM can discard the overlength data. The excess data is lost. This facility, called the truncate (TRUNC) option, is useful in application programs that must impose rigid size limitations on input data. For example, an inventory control application program might require the session partner to supply an account number no more than 10 bytes in length.
- VTAM can keep the data. VTAM fills the input area, saves the remainder, and completes the input request. Additional input requests must be issued to obtain the excess data. This facility is called the KEEP option.

When the data request read by VTAM is larger than the number of bytes specified in the AREALEN operand of a RECEIVE macroinstruction, the RECLEN field of the RPL indicates, after completion of the RECEIVE, the number of bytes that are available before the RECEIVE was executed. This characteristic of the RECLEN field is shown in Figure 43.

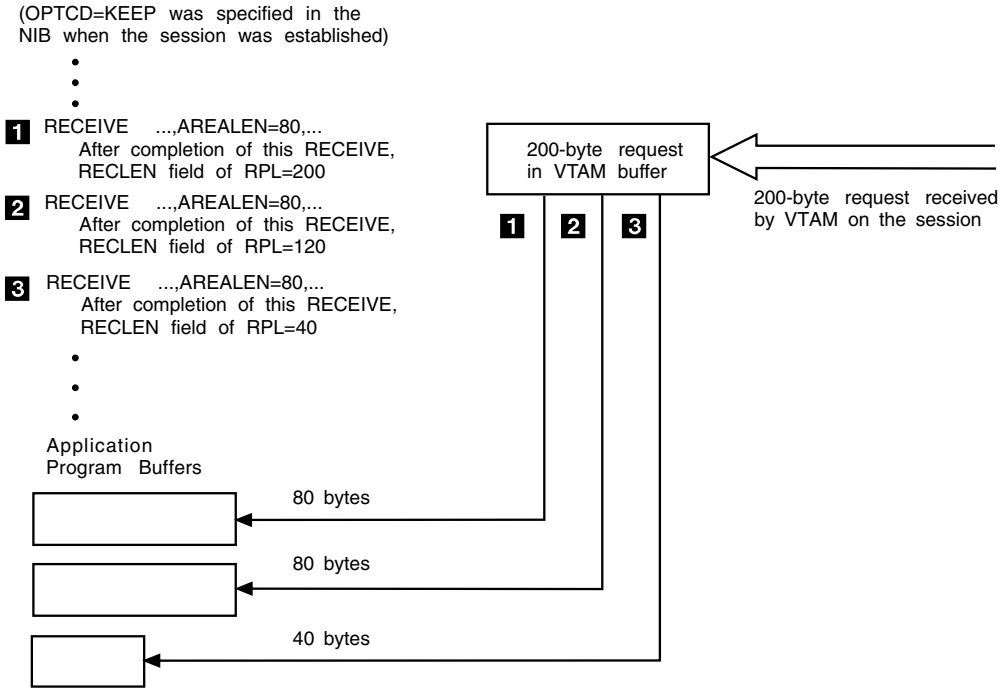


Figure 43. Example showing values in the RECLEN field of an RPL

The application program can select the appropriate option when the session with the LU is established (PROC=TRUNC or KEEP specified in the NIB). Or it can select it when RECEIVE is issued (OPTCD=TRUNC or KEEP specified in the RPL).

Large message performance enhancement outbound (LMPEO) option:

The large message performance enhancement outbound (LMPEO) option simplifies the task of writing an application program by significantly reducing the considerations of outbound chaining and maximum RU-size enforcement. LMPEO support is available only for sessions between SNA LUs that support SNA chaining.

LMPEO operating considerations:

When an application program issues SEND OPTCD=LMPEO, it passes FM data to VTAM and also passes information for an initial RH. VTAM reformats the FM data into one or more request units which form a chain or partial chain of RUs, and then sends each generated request to the other LU in the session as seen in Figure 44. None of these generated requests is larger than the maximum RU size specified in the BIND for sending in that direction on the session (that is, from the application program to the other LU in the session). If there is sufficient data, the first RU generated by VTAM has the maximum RU size allowed. If the amount of data specified by SEND is less than the maximum RU size, or if the maximum RU size in the BIND is 0, only a single request unit is generated. A maximum RU size of 0 specified in the BIND is inconsistent with the purpose of LMPEO, because VTAM does not break the data into multiple RUs if 0 is specified, no matter how large the message is.

The field in the BIND that specifies the maximum RU size is described in "Request unit size" on page 797. There are two such fields, one for each direction of data flow in the session (PLU to SLU and SLU to PLU). If the application program is the SLU, LMPEO uses the first field (byte 9) as the maximum RU size; if the application program is the PLU, LMPEO uses the second field (byte 10) as the maximum RU size.

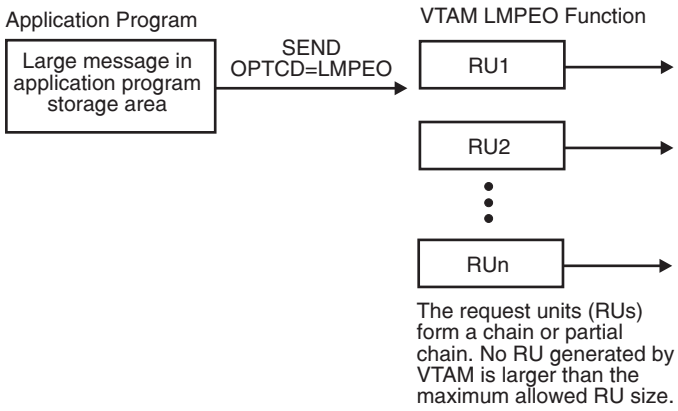


Figure 44. LMPEO operation on a message sent to an SNA LU

The location of the FM data to be sent by LMPEO varies depending upon whether the buffer-list option (OPTCD=BUFFLST) is used with LMPEO. BUFFLST also allows the application program to influence how LMPEO splits the FM data to be sent. Without the BUFFLST option (OPTCD=NBUFFLST), LMPEO sends the FM data from the area pointed to directly by the RPL AREA field, and splits the FM data (if necessary) into multiple RUs without regard for the data content. See "The buffer-list (BUFFLST) option" on page 191 for information on using BUFFLST with LMPEO.

If SEND OPTCD=NLMPEO is issued, then VTAM does not split the data into separate RUs and does not enforce the maximum RU size restriction. In that case, the application program must ensure that each RU is within the size limits agreed upon at BIND time.

If OPTCD=LMPEO and the RU being sent is not an FM data request (for example, CONTROL is not DATA), then SEND is rejected with (RTNCD,FDB2)=(X'14',X'77').

If the VTAM encrypt/decrypt facility is used, each of the requests generated by VTAM for a SEND is enciphered, if appropriate. For each SEND, all or none of the FM data is enciphered. This is under the control of the RPL CRYPT operand and the NIB ENCR operand. See "Sending and receiving enciphered data requests" on page 218 for information on cryptographic requirements.

Handling request headers (RH):

The initial RH is obtained from one of the following places as summarized in Table 25:

- RPL flags, if OPTCD=(NUSERRH,NBUFFLST) or OPTCD=(NUSERRH,BUFFLST)
- RPL user-RH field, if OPTCD=(USERRH,NBUFFLST)
- First buffer-list entry for each buffer group, if OPTCD=(USERRH,BUFFLST).

Table 25. Location of the initial RH

BUFFLST selected	LMPEO selected	USERRH selected	Number of RUs generated	Location of initial RH
no	no	no	1	RPL flags
no	no	yes	1	RPL user-RH field
no	yes	no	1 or more	RPL flags
no	yes	yes	1 or more	RPL user-RH field
yes	no	no	1	RPL flags
yes	no	yes	1	First entry of buffer list
yes	yes	no	1 or more	RPL flags
yes	yes	yes	1 or more	First entry of buffer list associated with each buffer group

The initial RH is used as a model to build the RH for each generated request. The initial RH indicates one of the following: only-in-chain (OIC), first-in-chain (FIC), middle-in-chain (MIC), or last-in-chain (LIC). See "Chaining" on page 201. VTAM generates requests as shown in Table 26. All generated requests are part of the same chain. If the initial RH specified OIC, a whole chain is generated. If the initial RH specified FIC, MIC, or LIC, a partial chain is generated.

The indicators that VTAM puts into the RH of each generated request depend on the initial RH and on whether the generated request is OIC, FIC, MIC, or LIC. See Figure 45 on page 187 and Figure 46 on page 188 for the rules VTAM uses to set the RH of each generated request.

Note: For OPTCD=NLMPEO, the RH propagation rules shown in Figure 45 on page 187 and Figure 46 on page 188 do not apply. The initial RH is used for the single request sent.

Table 26. Possible chain indicators resulting from initial RH-chain indicator settings

Initial RH-chain indicators	Resultant RH-chain indicators
OIC	OIC or FIC,LIC or FIC,MIC,...,MIC,LIC
FIC	FIC or FIC,MIC,...,MIC
MIC	MIC,...,MIC
LIC	LIC or MIC,...,MIC,LIC

Table 26. Possible chain indicators resulting from initial RH-chain indicator settings (continued)

Initial RH-chain indicators	Resultant RH-chain indicators
Note: In this table MIC,...,MIC represents a single MIC request or multiple MIC requests.	

In summary, for SEND OPTCD=LMPEO, the input request results in one or more output requests as follows:

- All chains are standard SNA chains, requesting no response (RQN), exception response (RQE), or definite response (RQD) as shown in Figure 45 on page 187 and Figure 46 on page 188.
- POST=SCHED or RESP can be used with SEND OPTCD=LMPEO. The SEND macroinstruction is posted complete when the last request generated from the message has been handled. The POST operand applies to this last request. Thus, posting occurs when the RPL is no longer needed for the request (POST=SCHED) or when the response asked for by the last request has been returned (POST=RESP).
- The normal session-level pacing and virtual-route pacing rules apply to requests generated by LMPEO. A large message can take a long time to transmit; therefore, OPTCD=ASY rather than OPTCD=SYN should be used if it is undesirable to suspend the task issuing SEND for that length of time. With OPTCD=ASY, the application program regains control while VTAM is sending the requests generated from the large message.

Additional considerations for posting of SEND are discussed in "Exception conditions" on page 190.

Initial RH settings		RHs generated by LMPEO					
		FIC + MIC + MIC + . . . + MIC + LIC					OIC
		See Notes:					
BBI or EBI in:	OIC	2	3	3	3	3	2
	FIC	2	3	3	3	4	4
(Note 5)	MIC	4	2	3	3	4	4
(Note 5)	LIC	4	2	3	3	3,7	4
QRI, CSI or EDI in:	OIC	2	2	2	2	2	2
	FIC	2	2	2	2	4	4
	MIC	4	2	2	2	4	4
	LIC	4	2	2	2	2	4
FI, RCDI, or SDI (Note 6) in:	OIC	2	3	3	3	3	2
	FIC	2	3	3	3	4	4
	MIC	4	2	3	3	4	4
	LIC	4	2	3	3	3,7	4
CDI or CEBI in:	OIC	3	3	3	3	2	2
(Note 5)	FIC	3,7	3	3	2	4	4
(Note 5)	MIC	4	3	3	2	4	4
	LIC	4	3	3	3	4	4

Figure 45. LMPEO handling of selected RH indicators

Legend:

Indicator	Meaning	Indicator	Meaning
BBI	Begin Bracket Indicator	FIC	First-in-Chain
CDI	Change Direction Indicator	LIC	Last-in-Chain
CEBI	Conditional End Bracket Indicator	MIC	Middle-in-Chain
CSI	Code Selection Indicator	OIC	Only-in-Chain
EBI	End Bracket Indicator	QRI	Queued Response Indicator
EDI	Enciphered Data Indicator	RCDI	Request Change Direction Indicator
FI	Format Indicator	SDI	Sense Data Included Indicator

Notes:

1. Any LMPEO request is rejected with (RTNCD,FDB2)=(X'14',X'7B'), if the pacing indicator (PI), padded data indicator (PDI), or reserved bits (byte 0, bit 3, byte 1, bits 1, 4, or 5) are set.
2. Output RH has the same setting as the initial RH.
3. Indicator is set to 0.
4. Output RH cannot be created from initial RH.
5. ***Warning:*** These lines show what VTAM currently implements if the specified input is given. This input is not valid in SNA, and the VTAM implementation is subject to change. The application program should not specify this input.
6. Sense Data Included (SDI) requests are not split. FM data exception requests can contain only 4 bytes of data which VTAM gets from the SSENSEO, SSENSMO, and USENSEO fields of the RPL. The SDI indicator is set on the first and only output request.
7. If no MICs are generated, the output request has the same setting as the input request.

Initial RH settings		RHs generated by LMPEO						
		FIC	+ MIC	+ MIC	+ . . . +	MIC	+ LIC	OIC
RQN (Note 1)	in:	See Notes:						
	OIC	RQN	RQN	RQN		RQN	RQN	RQN
	FIC	RQN	RQN	RQN		RQN	Note 5	Note 5
	MIC	Note 5	RQN	RQN		RQN	Note 5	Note 5
	LIC	Note 5	RQN	RQN		RQN	RQN	Note 5
RQE* (Note 2)	in:							
	OIC	RQE*	RQE*	RQE*		RQE*	RQE*	RQE*
	FIC	RQE*	RQE*	RQE*		RQE*	Note 5	Note 5
	MIC	Note 5	RQE*	RQE*		RQE*	Note 5	Note 5
	LIC	Note 5	RQE*	RQE*		RQE*	RQE*	Note 5
RQD* (Note 3)	in:							
(Note 6)	OIC	RQE*	RQE*	RQE*		RQE*	RQD*	RQD*
(Note 6)	FIC	RQE*	RQE*	RQE*		RQE*	Note 5	Note 5
(Note 6)	MIC	Note 5	RQE*	RQE*		RQE*	Note 5	Note 5
(Note 6)	LIC	Note 5	RQE*	RQE*		RQE*	RQD*	Note 5
RQX (Note 4)	in:							
(Note 6)	OIC	RQN	RQN	RQN		RQN	RQN	RQN
(Note 6)	FIC	RQN	RQN	RQN		RQN	Note 5	Note 5
(Note 6)	MIC	Note 5	RQN	RQN		RQN	Note 5	Note 5
(Note 6)	LIC	Note 5	RQN	RQN		RQN	RQN	Note 5

Figure 46. LMPEO handling of selected RH-chain indicators

Legend:

* 1, 2, or 3

RQN No response requested

RQE Exception response requested

RQD Definite response requested

RQX Invalid combination in SNA

Notes:

1. RQN = (¬ER,¬DR1,¬DR2) and RESPOND = (NEX,NFME,NRRN)
2. RQE* = RQE1 or RQE2 or RQE3 where:
   - RQE1 = (ER,DR1,¬DR2) and RESPOND = (EX,FME,NRRN)
   - RQE2 = (ER,¬DR1,DR2) and RESPOND = (EX,NFME,RRN)
   - RQE3 = (ER,DR1,DR2) and RESPOND = (EX,FME,RRN)
3. RQD* = RQD1 or RQD2 or RQD3 where:
   - RQD1 = (¬ER,DR1,¬DR2) and RESPOND = (NEX,FME,NRRN)
   - RQE2 = (¬ER,¬DR1,DR2) and RESPOND = (NEX,NFME,RRN)
   - RQE3 = (¬ER,DR1,DR2) and RESPOND = (NEX,FME,RRN)
4. RQX = (ER,¬DR1,¬DR2) and RESPOND = (EX,NFME,NRRN)
5. The output request cannot be created from input request.
6. ***Warning:*** These lines show what VTAM currently implements if the specified input is given. This input is not valid in SNA and the VTAM implementation is subject to change. The application program should not specify this input.

LMPEO sequence number handling:

Each request unit generated by VTAM during an LMPEO operation is assigned a sequence number just as if the application program had passed that request directly to VTAM with a SEND macroinstruction. When the SEND OPTCD=LMPEO macroinstruction is posted complete, the RPL OBSQVAL field contains the sequence number of the first generated request unit and the RPL SEQNO field contains the sequence number of the last generated request unit. If a large amount of data is specified for SEND OPTCD=LMPEO, it is possible for duplicate sequence numbers to be generated (because of sequence number wraparound) before SEND is posted complete. This can occur for SNA LUs only if more than 65535 requests are generated for SEND OPTCD=LMPEO.

If a response is returned containing the sequence number of one of the generated requests, the application program can usually determine the message involved by noting that the sequence number falls in the range between and including the OBSQVAL and SEQNO values posted for the SEND of that message.

Data-stream considerations:

When generating request units from a message sent with OPTCD=LMPEO, VTAM gives no consideration to the contents of the data. In general, the points at which VTAM splits the data, the RU boundaries, can occur anywhere in the data. VTAM ensures only that no RU is larger than the applicable maximum RU size, and that the first generated RU is equal to the maximum RU size, if there is enough FM data to make this possible.

Because VTAM does not consider data content when splitting a message into RUs, LMPEO should not be used where RU boundaries imposed by LMPEO are significant relative to the data. Use of the buffer-list option, in combination with LMPEO, allows the application program to specify RU boundaries and, therefore, to minimize any adverse effects of RU boundary selection by LMPEO. Refer to "The buffer-list (BUFFLST) option" on page 191 for information about how the buffer-list option is used with LMPEO.

Some cases in which data-stream architectures or device implementations place restrictions on the use of LMPEO are as follows:

1. For LU type 1, if card data streams are to be used, the data stream profile (DSP) in the BIND must specify that cards can span RUs; otherwise, LMPEO might place an RU boundary in the middle of a card image. If an LU type 1 implementation does not support this DSP, do not use LMPEO.
2. FM headers cannot span RUs for LU type 1. For LU type 1, all FM headers are contiguous and begin with the first byte of the chain. If the maximum RU size is equal to or greater than the largest concatenated set of FM headers sent by the application program to the LU, there is no problem with using LMPEO because the first RU generated for LMPEO data splitting is always equal to the maximum RU size. However, if the FM headers are larger, the application program can define RU boundaries for the part of the chain that has FM headers through use of the buffer-list option and LMPEO control flags. Alternatively, the application program can use multiple SENDs. It can send the FM header RUs with OPTCD=NLMPEO and send the rest of the chain using OPTCD=LMPEO.
3. For LU type 1, the string control byte (SCB) must be at the front of each RU. This is not guaranteed by VTAM when using LMPEO. Therefore, compression

cannot be used with LMPEO unless the application program uses the buffer-list option and LMPEO control flags to define its own RU boundaries.

Exception conditions:

It may happen that a negative response is received by VTAM for a chain that is generated by a SEND OPTCD=LMPEO. If the response is received before SEND is posted complete, then OPTCD=CONTCHN or NCONTCHN and POST=SCHED or RESP on the SEND control what action is taken. Table 27 and the following paragraphs describe how these operands are used. VTAM either continues to send the remainder of the chain (OPTCD=CONTCHN), or aborts the chain if there is any part of the message not yet sent (OPTCD=NCONTCHN). Normally, an application program should use NCONTCHN, the default.

Table 27. Negative-response handling by VTAM for SEND OPTCD=LMPEO

	POST=SCHED	**POST=RESP**
OPTCD=CONTCHN	Last or Not Last (see Note): • Deliver response to application program. • Send rest of chain. • Post SEND with (RTNCD,FDB2)=(X'0C',X'0D').	Not Last: • Deliver response to application program. • Send rest of chain. • Post SEND with (RTNCD,FDB2)=(X'0C',X'0D'). Last: • Post SEND with (RTNCD,FDB2)=(X'04',X'04') and negative response.
OPTCD=NCONTCHN	Last or Not Last: • Deliver response to application program. • Send zero-length LIC RU if SEND specified CHAIN=ONLY or LAST. • Post SEND with (RTNCD,FDB2)=(X'0C',X'0D').	Not Last: • Deliver response to application program. • Send zero-length LIC RU if SEND specified CHAIN=ONLY or LAST. • Post SEND with (RTNCD,FDB2)=(X'0C',X'0D'). Last: • Post SEND with (RTNCD,FDB2)=(X'04',X'04') and negative response.

Note:

Last means that unless there is a negative response received, this is the last request sent for this SEND. ***Deliver response to application program*** means that the application program must issue a RECEIVE RPL (RTYPE=RESP) or use a RESP exit routine.

When VTAM receives a negative response, VTAM ends the chain by immediately posting the SEND complete with (RTNCD,FDB2)=(X'0C',X'0D') and by not sending the remaining data of the message. Additionally, when a chain ends, if the last request normally sent for the message is an LIC request, then an LIC zero-length RU is sent to end the chain. This request has the same RH that would have been sent with the final request for that message if no error occurred.

When SEND fails with (RTNCD,FDB2)=(X'04',X'04'), the sense information is available in RPLFDBK2. However, when SEND fails with (RTNCD,FDB2)=(X'0C',X'0D'), then the application program must issue the RECEIVE RPL (or an available RESP exit routine) to obtain the failing sense code.

In all cases, the negative response is passed back to the application program (by means of a RESP exit routine, a RECEIVE, or by posting SEND POST=RESP). For SEND POST=RESP, if the sequence number of the negative response is the same as the sequence number of the last request sent, and if that request was the last request for the message (that is, no data remains to be sent for the message), then SEND is posted complete with (RTNCD,FDB2)= (X'04',X'04') instead of (X'0C',X'0D').

The OBSQVAL and SEQNO fields always indicate the sequence numbers of the first and last requests (including the zero-length RU discussed above) that are actually sent by VTAM.

An outstanding SEND is posted complete when a Clear request is sent or received or when the associated session ends. This is independent of whether OPTCD=LMPEO is used. For OPTCD=LMPEO, part of the chain might have been sent. In general, if any part of the data to be sent, including the address of any buffer list or buffer-list entry, is found to be not valid (cannot be referenced) when VTAM tries to send it, SEND fails with (RTNCD,FDB2)= (X'14',X'1E'). VTAM verifies that the data address is read/write capable.

VTAM does not verify the data address area if you use the authorized path facility described in Chapter 10, “Operating system facilities,” on page 295. The application program is responsible for terminating the chain if one has begun.

Performance considerations for LMPEO:

The major performance advantage in using OPTCD=LMPEO to SEND large messages is the significant reduction in path length (the number of instructions executed) achieved when compared to the application program itself splitting a large message and issuing individual SENDs. This advantage should be weighed, however, against the possibility of less efficient use of application program storage by LMPEO.

Application program storage might be used less efficiently because SEND is not posted complete (indicating that the output area is now available) until the whole message has been handled by VTAM. Pacing might delay the completion of the LMPEO SEND. With OPTCD=NLMPEO, the application program can reuse each SEND's output area as soon as that SEND completes. This assumes that the data is available elsewhere (for example, on a disk) if it must be present.

It is somewhat less efficient to use OPTCD=LMPEO instead of OPTCD=NLMPEO to SEND a message that does not require splitting (if the size of the message to be sent is less than or equal to the maximum RU size). The application program could pretest the size of a message to be sent to decide whether to use OPTCD=LMPEO.

The buffer-list (BUFFLST) option:

The buffer-list (BUFFLST) option simplifies the task of writing an application program by eliminating the need for data to be sent from one contiguous storage area.

BUFFLST operating considerations:

SEND OPTCD=BUFFLST allows FM data to be sent from a number of discontiguous buffers; this can result in more efficient use of application program storage areas. The RPLAREA field points to a buffer list, which is a contiguous set

of 16-byte control blocks called buffer-list entries. Each buffer-list entry points to a buffer that contains part of the data to be sent.

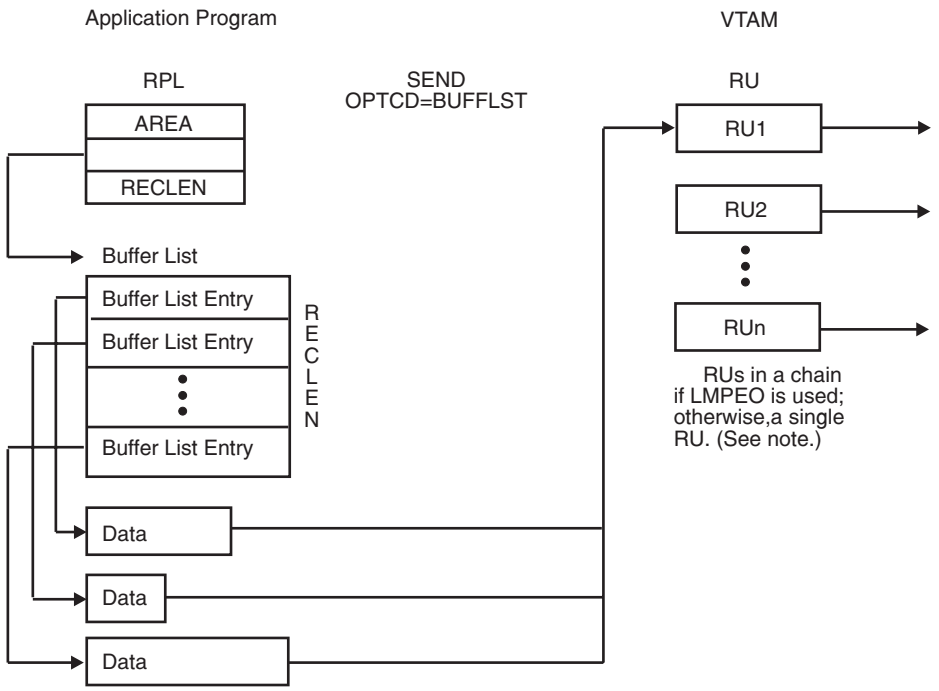


Figure 47. Buffer-list operation

Note: If LMPEO is used to split data into RUs, the RU boundaries do not usually correspond to the buffer boundaries. There may be more or fewer RUs than the buffer-list entries.

If the message to be sent using OPTCD=BUFFLST is not an FM data request or a set of FM data requests, SEND is rejected with (RTNCD,FDB2)=(X'14',X'77').

The RPL RECLEN field specifies the length (in bytes) of the buffer list. If RECLEN does not specify a nonzero multiple of 16, SEND is rejected with (RTNCD,FDB2)=(X'14',X'79'). Each buffer-list entry is mapped in the ISTBLENT DSECT described in Appendix E, “Control block formats and DSECTs,” on page 739.

If OPTCD=(BUFFLST,LMPEO), then each buffer-list entry contains LMPEO control flags which govern whether the buffers associated with adjacent buffer-list entries should be grouped together as a single RU, or whether the data is eligible to be split into multiple RUs. See “Buffer-list LMPEO states” on page 193 for a description of how LMPEO control flags are used to govern RU boundary settings. See also Table 25 on page 185 for information about using combinations of the BUFFLST, LMPEO, and USERRH options. The format of the buffer list follows. A buffer list consists of one or more contiguous buffer list entries aligned on a fullword boundary.

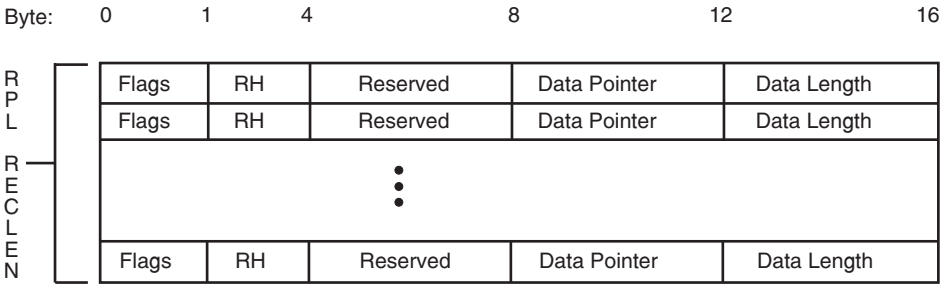


Figure 48. Buffer-list entry format

Table 28. Buffer-list entry format

Word	Contents
1	**Byte 0** Flag Byte. The first two flags are LMPEO control flags. They are used only if OPTCD=LMPEO is specified. Otherwise, they must be set to 0. [1] • Bit 0 The Begin RU flag (BLEBEGRU) in the ISTBLENT DSECT. If on, it indicates that the data in the associated buffer begins an RU. VTAM does not split this data into multiple RUs. • Bit 1 The End RU flag (BLEENDRU) in the ISTBLENT DSECT. If on, it indicates that the data in the associated buffer ends an RU. • Bits 2–7 Reserved. Bits must be set to 0. **Bytes 1-3** Request header. If either OPTCD=NUSERRH or this buffer-list entry does not contain a user RH (it is not the first entry of a buffer group), these bytes are reserved and must be set to 0.
2	Reserved. Must be set to 0.
3	Pointer to the beginning of the data buffer.
4	Length of the data in the buffer. If 0 is specified, the third word is ignored. However, the flags in byte 0 of the first word are processed and can change the LMPEO states. For RUs whose boundaries are specified by the application program (using the Begin RU and End RU flags), the total RU length must not be larger than 65 535 bytes. If the LMPEO option is used to split the data into RUs, a length greater than 65 535 may be specified.

Note:

1. These flags are discussed in detail, see "(Begin RU, End RU)=(1,1)" on page 195, "(Begin RU, End RU)=(1,0)" on page 195, "(Begin RU, End RU)=(0,1)" on page 196, and "(Begin RU, End RU)=(0,0)" on page 196.

Buffer-list LMPEO states:

The LMPEO and BUFFLST options can be used in combination for a SEND macroinstruction. In this case, LMPEO control flags in each buffer-list entry allow the application program to either explicitly specify RU boundaries for the associated data or let VTAM determine the RU boundaries. As indicated in

"Data-stream considerations" on page 189, it is sometimes necessary for an application program to specify where RUs should begin and end in the data stream.

One or more adjacent entries in a buffer list form a buffer-list-entry group. The associated buffers form a buffer group. The data in such a buffer group can be designated either as data that should be accumulated, that is, considered as a single RU and thus should not be split by LMPEO, or as data that can be split into multiple RUs by LMPEO.

Processing of a buffer list by VTAM involves three states: the reset state, the split state, and the accumulate state.

The reset state is entered at the beginning and end of processing the buffer list, and whenever an entry flagged as End RU is processed.

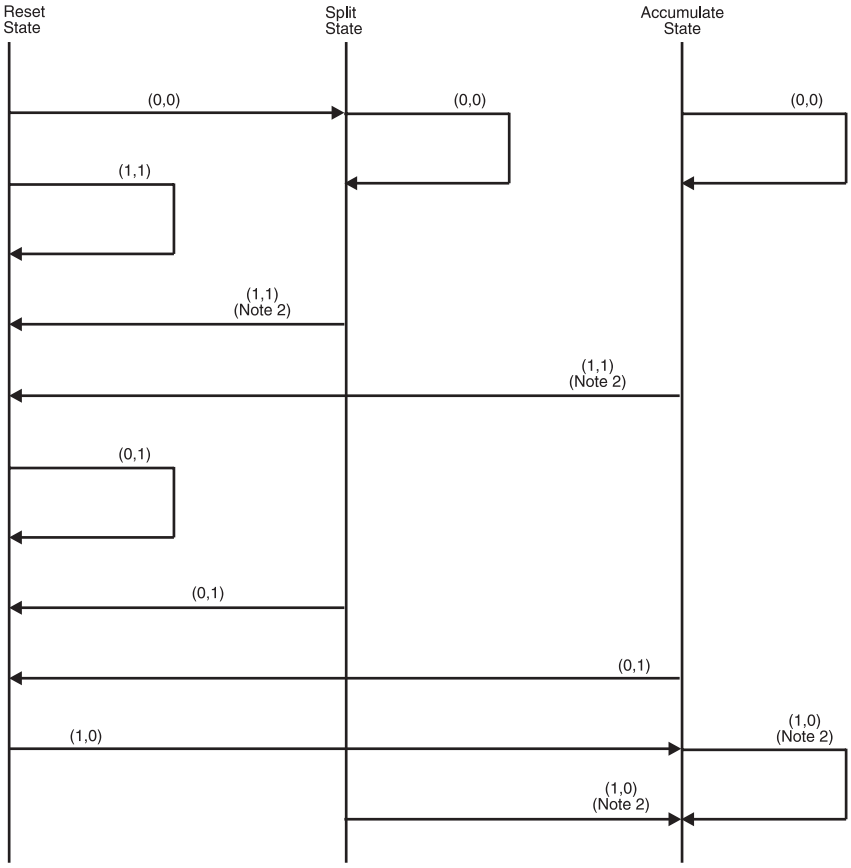
In the split state, VTAM is allowed to split the data into RUs based on the maximum RU size for the session. See "LMPEO operating considerations" on page 183 for further information.

In the accumulate state, VTAM accumulates data into a single RU.

Figure 49 on page 195 shows the state transitions that occur when each buffer-list entry is processed. For those transitions in which a new buffer group begins, the data from the previous group is sent as one or more RUs, and the data from the new group forms one or more separate RUs. Therefore, data from the two groups is never part of the same RU.

Within a buffer group, the number of buffers and the way the data is divided among the buffers does not affect the final result. If a buffer-list entry specifies a length of zero for the buffer data, the associated buffer pointer is ignored. However, the LMPEO control flags are still used to delimit buffer groups and to determine the state transitions (discussed below).

If OPTCD=USERRH is used with OPTCD=(LMPEO,BUFFLST), the initial RH is taken from the first buffer-list entry associated with the buffer group. The RH fields of the other buffer-list entries associated with the buffer group are reserved. For an example, see "Example of using LMPEO, BUFFLST, and USERRH" on page 200.



Notes:

1. The pair of numbers in parantheses above each arrow represents the value of (Begin RU, End RU) in the current buffer list entry. Processing this entry causes a state transition from the state at the tail of the arrow to the state at the head of the arrow.

   The last entry of the list is always handled as if it had the End RU flag set on; the Begin RU flag can be either on or off.

2. This entry causes the previous buffer group to end before this entry is processed.

Figure 49. Buffer-list LMPEO-state transitions

Possible combinations of the Begin RU and End RU LMPEO control flags are shown in the following sections. The numbers in parentheses represent the values of the corresponding control flags.

(Begin RU, End RU)=(1,1): All the data pointed to by this entry forms a single RU. The buffer group consists solely of the buffer pointed to by this entry. If this entry is encountered while in split or accumulate state, the previous buffer group is ended and the RUs from the data in that group are completed and sent. Then the RU associated with this entry is sent and reset state is entered. If a (1,1) entry is encountered in reset state, the single associated RU is sent, and reset state is maintained.

(Begin RU, End RU)=(1,0): The data pointed to by this entry forms the initial part of a single RU. Subsequent entries in this group can add data to the RU. If this entry is encountered while in split or accumulate state, the previous buffer group is ended and the RUs from the data in that group are completed and sent;

accumulate state is then entered. If a (1,0) entry is encountered in reset state, accumulate state is entered and no RU is sent at this time.

(Begin RU, End RU)=(0,1): The data pointed to by this entry forms the final part of the RU or RUs associated with the current buffer group, and is appended to data accumulated from previous entries (if any) in this buffer group. This entry ends that buffer group. If this entry is encountered in split or accumulate state, the current buffer group is ended and the associated RUs are sent; reset state is then entered. If a (0,1) entry is encountered in reset state, this entry begins and ends a group. The associated data is eligible to be split into multiple RUs; the RUs are sent and reset state is maintained.

(Begin RU, End RU)=(0,0): The data pointed to by this entry is appended to the data pointed to by previous entries (if any) in this buffer group. If a (0,0) entry is encountered in accumulate state, the data becomes part of the single RU being accumulated; accumulate state is maintained. If a (0,0) entry is encountered in split state, the data is added to the data that is split by VTAM when a sufficient amount is gathered; split state is maintained. If a (0,0) entry is encountered in reset state, this entry begins a new buffer group and split state is entered. The data is the initial part of the data that is split by VTAM when a sufficient amount is gathered.

End of buffer list: This condition acts as the implicit end of the current buffer group. The last entry in the buffer list is handled as if the End RU flag were set. Thus, the last entry is handled as described above for the (0,1) or (1,1) values of (Begin RU, End RU). In all cases, when the end of the buffer list has been reached, all remaining data is formed into RUs and sent. Data is never carried forward to be used with the next SEND.

The user RH (USERRH) option:

When an application program issues a SEND macroinstruction, it has the option (OPTCD=USERRH) of specifying a 3-byte RH (request header or response header) for each RU sent. The header to be sent is specified in an RPL field (RPLURH) or in a buffer-list entry if the buffer-list option is used.

If the user RH option is not used (OPTCD=NUSERRH), VTAM translates the RH indicators from the contents of other RPL fields associated with RPL operands such as STYPE=REQ and RESPOND=(EX,FME). In this case, VTAM also enforces certain SNA rules governing combinations of RH indicators and specifying which indicators are sent with certain data-flow-control RUs. (This is consistent with earlier levels of VTAM.)

USERRH operating considerations:

With the user RH option, the application program specifies the RH in the SEND RPL or in a buffer-list entry. This user RH field has the same format as the SNA RH. The ISTRH DSECT, described in Appendix E, "Control block formats and DSECTs," on page 739, can be used as a map for this field.

The USERRH option can be used when sending any function-management-data or data-flow-control request or response. It can be used either with or without the LMPEO and BUFFLST options. See Table 25 on page 185 for the location of the initial RH. If the buffer list option is used, each buffer-list entry that defines the beginning of an RU (or group of RUs created by an LMPEO splitting operation) must contain an RH field.

With the user-RH option, VTAM does not enforce particular settings of RH indicators related to data flow control and FM data. However, VTAM does control the settings of RH indicators related to transmission services, session control, and network control; and also prevents the setting of those indicators reserved by SNA, except RH byte 2, bit 3, the request change direction indicator. Table 29 lists the RH indicators to which the user RH option is applicable.

The RH for a request or response inbound to the application program is made available in the RECEIVE RPL or in the read-only RPL of the RESP, DFASY, SCIP, and NSEXIT exit routines.

Operation for outbound RUs:

The application program invokes the user-RH option for outbound flow by issuing SEND OPTCD=USERRH. Both FM data and data-flow-control requests and responses can be sent. The RPL CONTROL operand must specify DATA or one of the valid data-flow-control RUs (either expedited-flow or normal-flow). VTAM rejects SEND with (RTNCD,FDB2)=(X'14',X'7B') if the user-RH RU-category flags are inconsistent with CONTROL (for example, CONTROL=SIGNAL, but the RU category indicates FMD instead of DFC).

If OPTCD=NBUFFLST, the user-RH field from the SEND RPL is used. If OPTCD=LMPEO, this RH defines the initial RH from which are derived the RHs of the request units generated by the LMPEO operation. If OPTCD=(BUFFLST,NLMPEO), a single request unit is being sent and the RH is obtained from the first buffer-list entry.

If OPTCD=(BUFFLST,LMPEO), the LMPEO control flags in the buffer-list entries are used to define RU boundaries. For any buffer group that is specified as not to be split by LMPEO, the associated RH for that RU comes from the first buffer list-entry for the RU. For any buffer group that can be split by LMPEO, the first buffer list entry for the group defines the initial RH from which are derived the RHs of the RUs generated by the LMPEO operation.

If the pacing indicator (PI), padded data indicator (PDI), or reserved indicators are set in the user-RH for SEND OPTCD=USERRH, SEND is rejected with (RTNCD,FDB2)=(X'14',X'7B'). The indicators that cannot be set are the indicators with “no” specified in the fourth column of Table 29.

Table 29. Relationship of the user RH field to the request/response header

Byte	Bit	SNA name	RH bits applicable to user RH field	RPL operands related to RH-bit settings[1]
0	0	RRI	Yes	STYPE=RESP or REQ
	1,2	RUCAT	Yes[2]	Implied by CONTROL
	3	Reserved	No	None
	4	FI	Yes	OPTCD=FMH
	5	SDI	Yes	Implied by SSENSO, SSENSMO, USENSEO
	6,7	BCI, ECI	Yes	RESPOND=EX for responses CHAIN

Table 29. Relationship of the user RH field to the request/response header (continued)

Byte	Bit	SNA name	RH bits applicable to user RH field	RPL operands related to RH-bit settings[1]
1	0	DR1I	Yes	RESPOND=FME
	1	Reserved	No	None
	2	DR2I	Yes	RESPOND=RRN
	3	ERI/RTI	Yes	RESPOND=EX
	4	Reserved	No	None
	5	Reserved	No	None
	6	QRI	Yes	RESPOND=QRESP
	7	PI	No	None
2	0	BBI	Yes	BRACKET=BB
	1	EBI	Yes	BRACKET=EB
	2	CDI	Yes	CHNGDIR=CMD
	3	Reserved	Yes	None
	4	CSI	Yes	CODESEL
	5	EDI	Yes	CRYPT
	6	PDI	No	None
	7	CEBI	Yes	BRACKET=CEB

Note:

1. These are the RPL operands used to set the RH bits when OPTCD=NUSERRH.
2. Only the data-flow-control and FM data categories can be set by the application program.

Legend

Indicator	Meaning	Indicator	Meaning
BBI	Begin Bracket Indicator	ERI	Exception Response Indicator
BCI	Begin Chain Indicator	FI	Format Indicator
CDI	Change Direction Indicator	PI	Pacing Indicator
CEBI	Conditional End Bracket Indicator	PDI	Padded Data Indicator
CSI	Code Selection Indicator	QRI	Queued Response Indicator
DR1I	Definite Response 1 Indicator	RRI	Request/Response Indicator
DR2I	Definite Response 2 Indicator	RTI	Response Type Indicator
EBI	End Bracket Indicator	RUCAT	RU Category
ECI	End Chain Indicator	SDI	Sense Data Included Indicator
EDI	End Data Indicator		

Operation for inbound RUs:

With one exception noted in "Relationship to POST=RESP" on page 199, the RPL user RH field is always set when a request or response is delivered to an application program. Therefore, setting OPTCD=USERRH is not necessary for an inbound operation. The RH field that is set is the one in the RECEIVE RPL or in the read-only RPL of the RESP, DFASY, SCIP, or NSEXIT exit routine in which the request or response is received. When the RPL RH field is set, the other RPL fields related to RH indicators (for example, those associated with STYPE and RESPOND) are also set.

The RH of the input request or response is moved directly to the RPL's user-RH field. Those indicators related to transmission services (the PI and PDI) and the currently reserved SNA indicators are set to 0, except for byte 2, bit 3, which is set as received in the input RH.

Relationship to POST=RESP:

The exception to the preceding description occurs when a response is received that completes the posting of an outstanding POST=RESP output operation. In this case, the input RH of the response is not placed in the user-RH field of the SEND or SESSIONC RPL. Thus, the original setting of the RPL's user-RH field is preserved. This applies whether the POST=RESP is explicit, for example:

```
SEND POST=RESP,STYPE=REQ,CONTROL=DATA,RESPOND=(NEX,FME)
```

or implicit, for example:

```
SESSIONC STYPE=REQ,CONTROL=SDT
```

Refer to the description of the SEND and SESSIONC macroinstructions in Chapter 13, “Conventions and descriptions of VTAM macroinstructions,” on page 371 and “Synchronous versus asynchronous operations” on page 169, in this chapter, for details about the POST operand.

Relationship to NIB PROC=ORDRESP or NORDRESP operand:

The NIB PROC=ORDRESP or NORDRESP operand controls how VTAM handles the receipt of responses in which the queued response indicator (QRI) is turned on. Refer to “Controlling the handling of normal-flow responses” on page 161 for a description of the PROC=ORDRESP or NORDRESP operands. This handling of responses is independent of the user-RH option. PROC=ORDRESP or NORDRESP also controls the setting of the RPL RESPOND operand for the sending of normal-flow data-flow-control (DFC) requests and responses, except when OPTCD=USERRH.

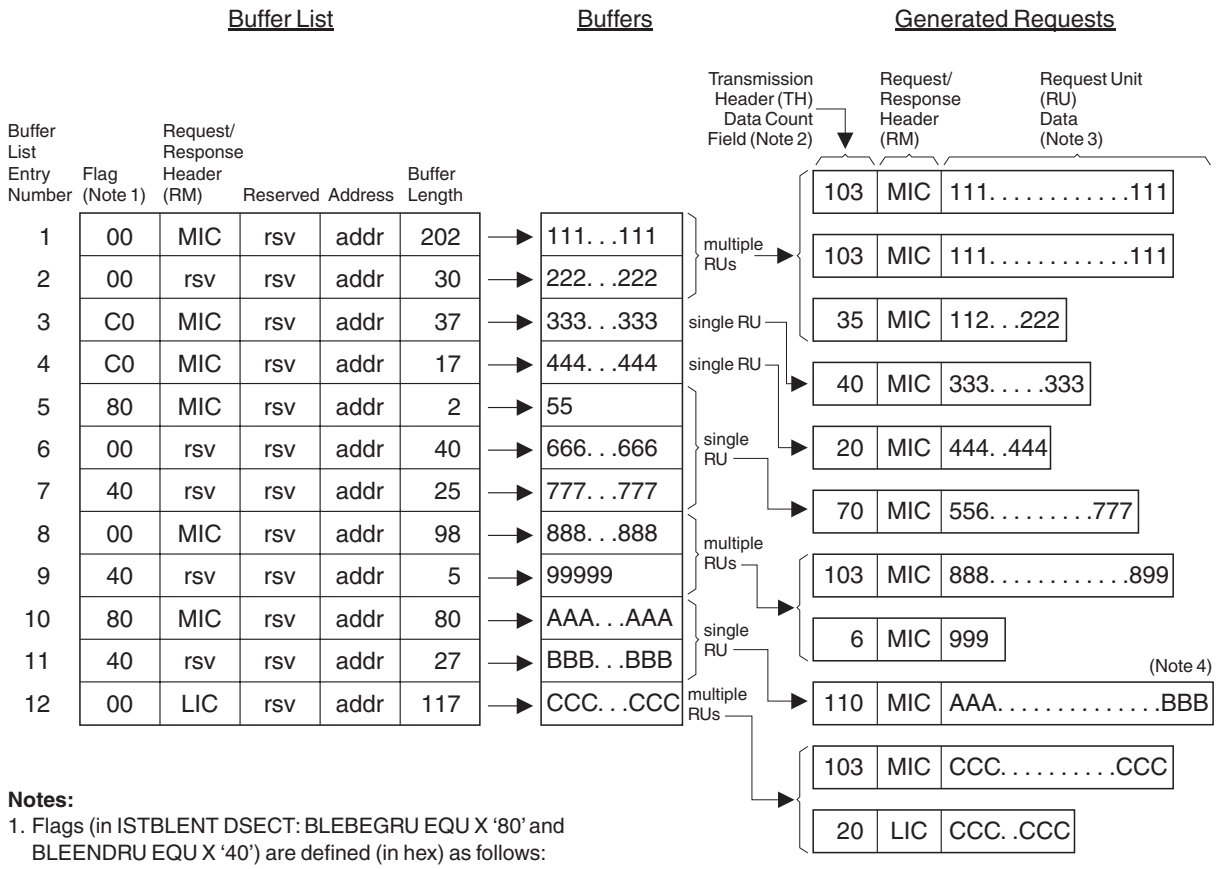
PROC=ORDRESP or NORDRESP is ignored for outbound flow when SEND OPTCD=USERRH is used. It still applies for any inbound flow. For SEND OPTCD=USERRH, the user-RH settings are used for all DFC requests and responses. It is the application program's responsibility to ensure that the indicators are set in accordance with the SNA protocols being used for that session.

Handling the sense data included (SDI) indicator:

If the sense data included (SDI) indicator is on in the user RH field for SEND OPTCD=USERRH, then the associated 4 bytes of sense data are obtained from the SSENSEO, SSENSMO, and USENSEO fields of the RPL. This is applicable to sending requests (exception requests) and responses (negative responses), both FM data and DFC. If the SDI indicator is on for an FM data RU, the RU consists only of those 4 bytes of sense data. For a DFC RU, VTAM appends the DFC request code to the sense data. If the SDI indicator is on, the data length (RECLEN or the fourth word of the corresponding buffer-list entry) must be 0. If the length is not zero, the SEND operation fails with (RTNCD,FDB2)=(X'14',X'1E'). The data pointer (AREA or the third word of the buffer list entry) is ignored.

When sending a negative response using OPTCD=USERRH, the application program must ensure that both the SDI and response type (RTI) indicators are

turned on. For OPTCD=NUSERRH, VTAM automatically turns on the SDI indicator when sending a negative response (for example, SEND STYPE=RESP,RESPOND=(EX,FME)).



Notes:

1. Flags (in ISTBLENT DSECT: BLEBEGRU EQU X '80' and BLEENDRU EQU X '40') are defined (in hex) as follows:
   - 80 = Marks the beginning of a user-defined RU.
   - 40 = Marks the end of a user-defined RU.
   - C0 = Marks a user-defined RU.
   - 00 = If accumulating a user-defined RU, marks a continuation of that RU. If not accumulating a user-defined RU, marks a buffer whose data is to be combined with the data from previous buffers; VTAM will split the resulting data into RUs.
2. Total number of bytes (RH + data).
3. The number of RU data bytes is never greater than the maximum RU size specified in the BIND when VTAM does the data splitting.
4. The application program may (but should not) specify an RU size greater than the maximum RU size agreed to in BIND. Specifying a larger RU size is in violation of SNA protocols, and will cause unpredictable results.

Figure 50. Example of a SEND operation, OPTCD=(LMPEO,BUFFLST,USERRH) with a maximum RU size of 100

Example of using LMPEO, BUFFLST, and USERRH:

The example contained in Figure 50 shows the results of a SEND OPTCD=(LMPEO,BUFFLST,USERRH) in which the application program has specified a buffer list of 12 entries, pointing to 12 discontiguous data areas. For some of the data, the application program wishes to define the RU boundaries. For the rest of the data, VTAM splits the data into RUs based on a maximum RU size of 100 bytes, which was obtained from the appropriate maximum RU size field in the BIND.

Buffer-list entries 1 and 2 point to data that is split by VTAM. Entry 1 has the initial RH. Because the RH specifies MIC, all three generated RUs are marked as MIC. This chain is assumed to have been started by a previous SEND. Even though entry 2 does not explicitly end the RU, VTAM recognizes that because entry 3 begins another RU, entry 2 must end the current RU.

Entry 3 points to a single RU. This data is not split and is not accumulated with data associated with another entry. Entry 3 has the RH, and is sent as MIC.

Entry 4 points to another single RU. It is handled the same as the preceding entry. Entry 4 has the RH, and is sent as MIC.

Entries 5, 6, and 7 point to data that is accumulated into a single RU. The RU is sent as MIC. Entry 5 has the RH. Entry 7 explicitly ends the current RU. This allows VTAM to recognize that entry 8 begins a new RU.

Entries 8 and 9 point to data that is split by VTAM. Entry 8 has the initial RH. Because the RH specifies MIC, both RUs are marked as MIC. Entry 9 explicitly ends this RU. This is optional because entry 10 begins a new RU.

Entries 10 and 11 point to data that is accumulated by VTAM into a single RU. Entry 10 has the RH. The data is combined into a single RU and sent as MIC. The application program defines this RU in such a way that it exceeds the maximum allowed RU size for the session. Although VTAM does not prevent this, it is a violation of the RU size agreement reached by the two LUs when the session was established, and should not be done by the application program. Depending upon the action taken by the other LU in the session, unpredictable results can occur.

Entry 12 points to data that is split by VTAM. Entry 12 contains the initial RH. Because the RH specifies LIC, the first RU is marked as MIC, and the next (last) RU is marked as LIC.

Using SNA protocols

The major alternatives previously described are of interest to all VTAM application program designers. Here are some additional facilities that not every user requires, but should be considered:

- The chaining of requests so that the number of responses required is minimized (CHAIN=ONLY or FIRST or MIDDLE or LAST)
- The quiescing of requests so that a sender may be told to temporarily stop sending when, for example, an input buffer is about to overflow (CONTROL=QEC)
- A method of communication that ensures that only the VTAM application program or the LU in session with it can be sending at one time, using either:
  - Quiesce protocol
  - Change-direction protocol.
- A method of communication that ensures that unexpected output from a VTAM application program is postponed until completion of an existing transaction (bracket protocol).

For further details about the session parameters discussed in the following sections, refer to Appendix F, “Specifying a session parameter,” on page 793.

Chaining:

LUs can group any number of data requests into a set called a chain. The sender can indicate which part of a chain is being transmitted—the first request of the chain, the last request of the chain, neither (the request is somewhere in the middle), or both (the request is the only request in the chain).

SNA allows you to use only three types of chains:

- ***No-response chain***, in which each request in the chain asks for no response
- ***Exception-response chain***, in which each request in the chain asks only for an exception response
- ***Definite-response chain***, in which the last request in the chain asks for a definite response and all other requests ask only for an exception response.

The sender of a chain that has not yet been completely sent can at any time send a Cancel request to the receiver (the sender might send this request because the receiver has returned a negative response). The Cancel request informs the receiver that the current chain is abnormally terminated, that the receiver will not receive further requests in the chain, and that the receiver might want to discard the chain requests it has already received.

The actual unit of work that the chain represents is determined entirely by the PLU and SLU. When the session is established, the PLU and the SLU determine what chaining protocols are to be used.

Figure 51 on page 203 illustrates a possible use of chaining. In this example, an LU submits an inquiry to the application program. The application program can obtain various pieces of information from data files and send them to the LU as each becomes available. By chaining the output requests, the application program has a convenient way of telling the LU whether any given piece of data represents the beginning, middle, or end of a reply to an inquiry.

Figure 52 on page 204 and Figure 53 on page 205 show in more detail the use of chaining illustrated by Figure 51 on page 203. Chaining is also shown in Figure 114 on page 701.

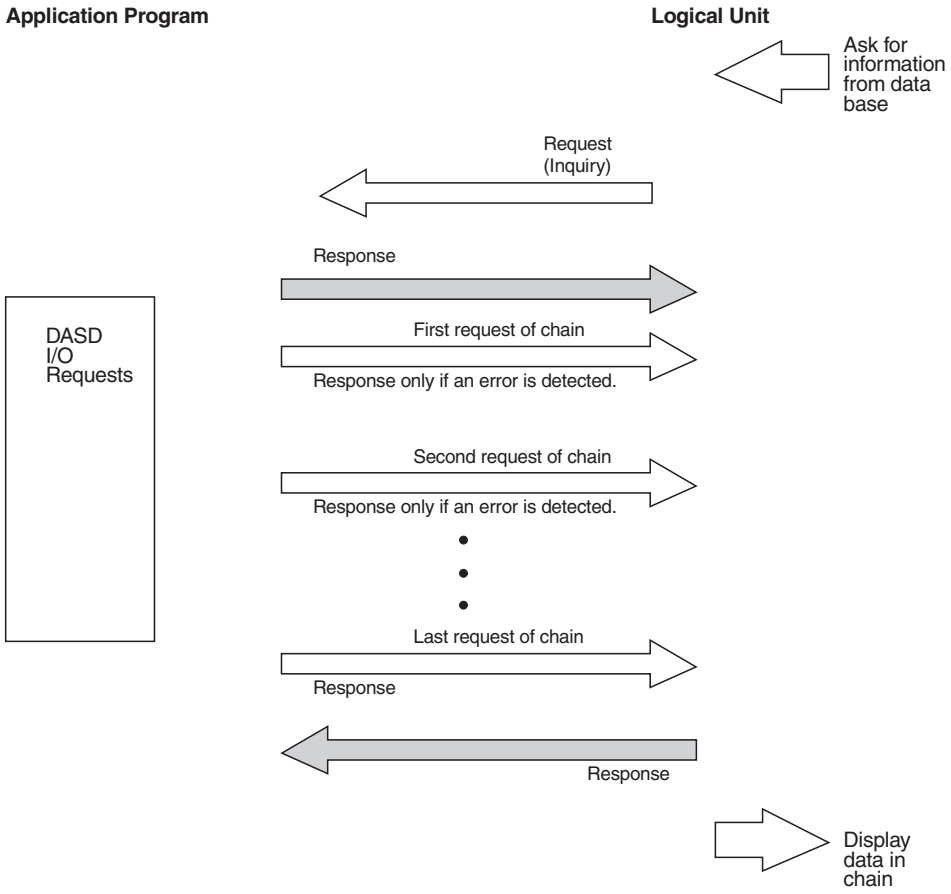


Figure 51. Example of request chaining

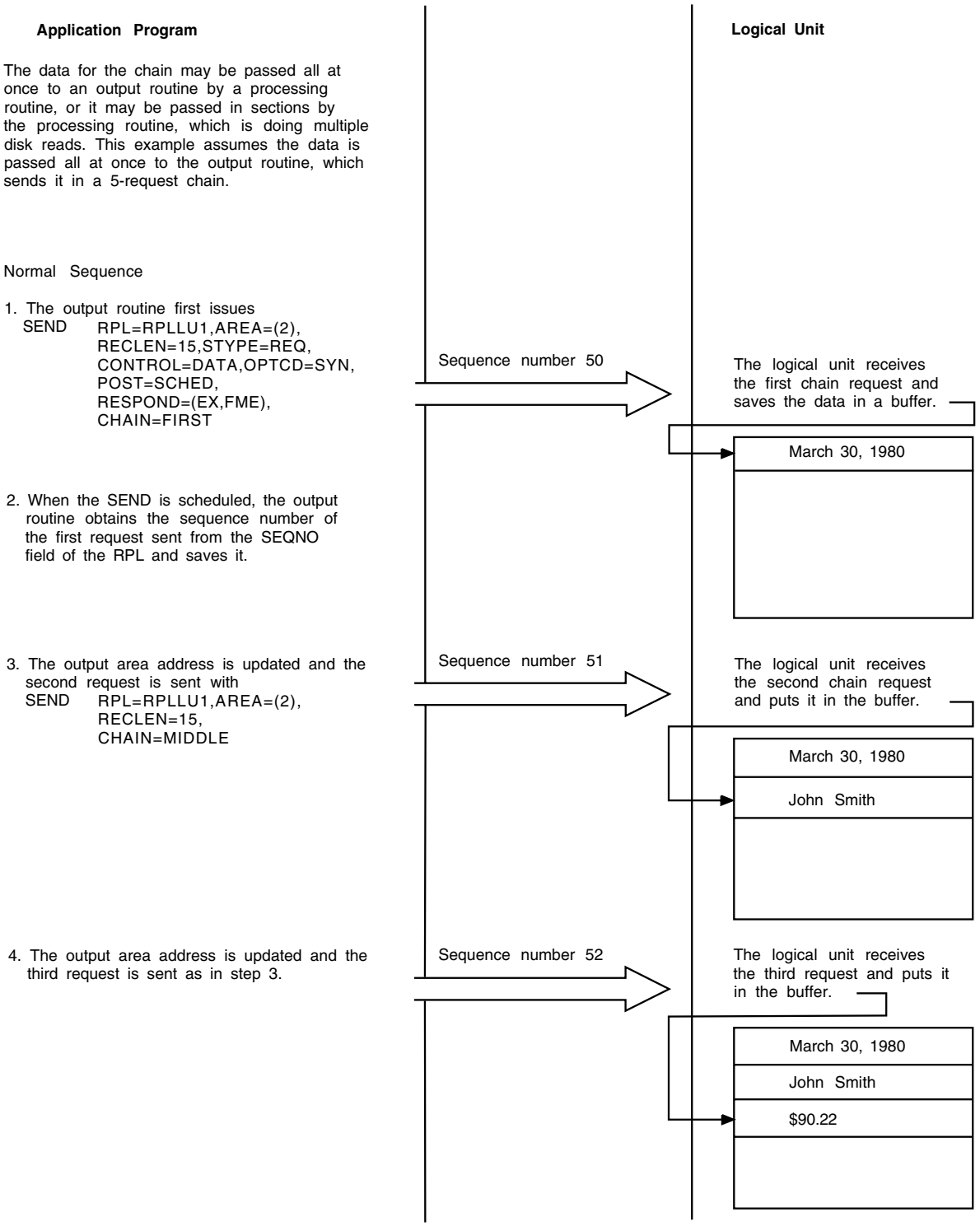


Figure 52. Example of sending a chain of requests to an LU that is buffering data (Part 1 of 2)

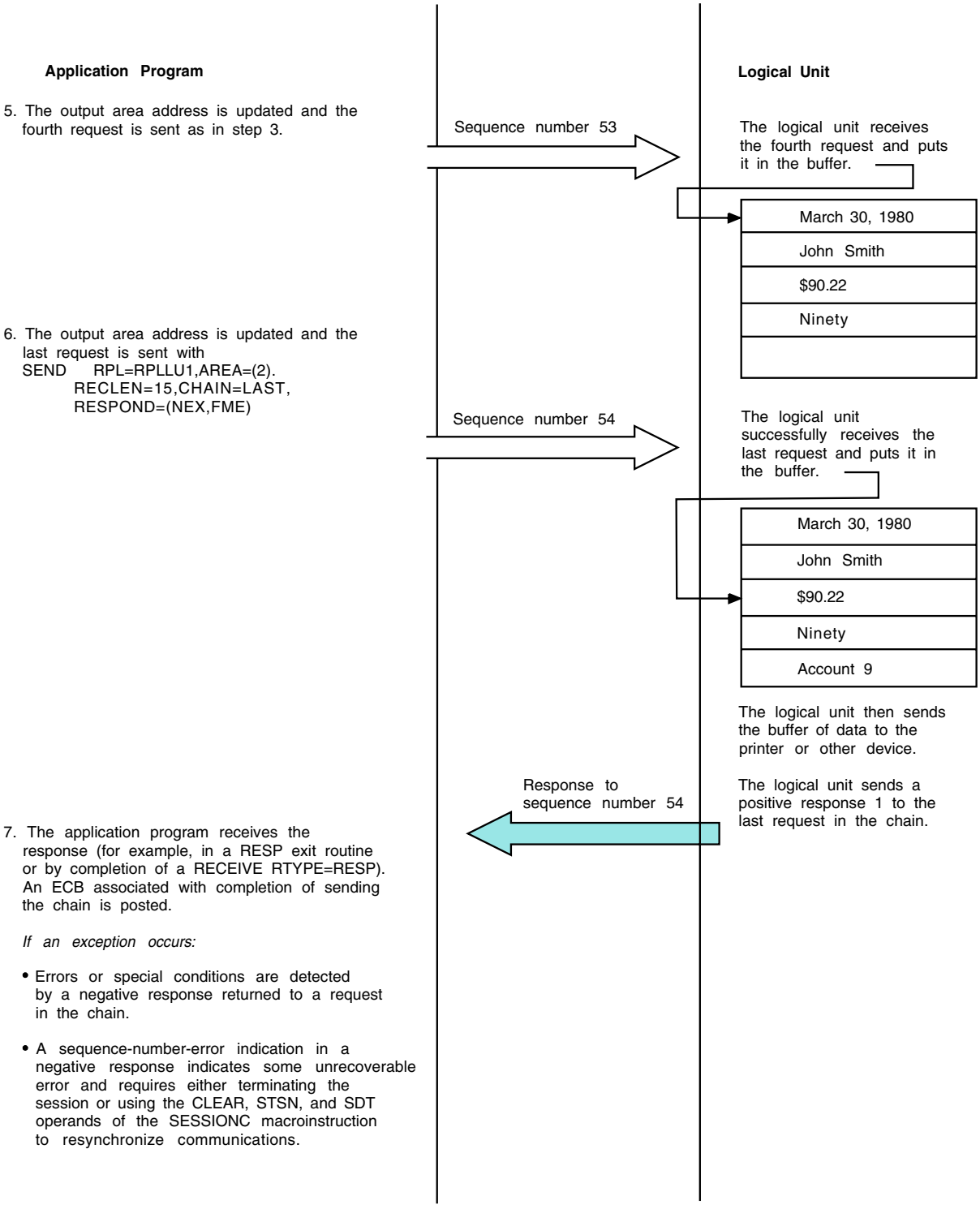


Figure 53. Example of sending a chain of requests to an LU that is buffering data (Part 2 of 2)

Request and response modes: The session parameters sent as part of session establishment contain combinations of protocol bits which establish certain request and response modes. The bits indicate, among other things, whether chaining is permitted, how often a response is asked for, and in what order the responses must be returned.

The sender (application program or LU) may be in either of two modes:

Immediate request mode
Using this mode, the sender can send a series of requests constituting one or more complete chains and ask for a definite response only in the last request in the series. After asking for a definite response, the sender does not send another normal-flow request until it receives the definite response. Thus, if the sender is sending a series of single-request chains, only the last request asks for a definite response; the other requests ask for an exception response only. If the sender is sending a multiple-request chain, only the last request in the chain asks for a definite response. If the sender is sending multiple chains, only the last request in the last chain asks for a definite response; all preceding requests ask for an exception response only.

Delayed request mode
Using this mode, the sender can insert requests asking for definite responses to the series of requests it is sending; it is not required to wait for any of those responses. This mode can be used to send multiple chains, with a definite response requested in the last request of each chain (all other requests in each chain would request an exception response only), and the sender can send any number of chains before stopping to wait for responses.

The receiver (application program or LU) can be in either of two modes:

Immediate response mode
The request receiver sends responses in the same order as the request sender asked for them. Thus, when the request sender receives a response to a request, it can infer that the request receiver has received and completed processing all preceding requests, and that no negative responses are forthcoming for those preceding requests.

Delayed response mode
The request receiver need not return responses in the same order as they were asked for. A response for one request can be delayed beyond the response for a subsequent request. There is one restriction, however, on the receiver. The receiver must send responses for normal-flow requests preceding a Chase request before it sends the response to the Chase request.

Quiescing:

SNA provides a set of data-flow-control requests that the application program can use to ask an LU to stop sending normal-flow requests (data requests and normal-flow data-flow-control requests) to the program. An LU can also ask that the VTAM application program stop sending normal-flow requests on a session.

One use of this facility is to ensure that, at a given time, only one session partner can send normal-flow requests. (This use of the quiesce requests is described later as “quiesce protocol.”)

Another use of quiescing is to stop the other end of the session from sending because of a temporary condition or problem. This action is usually needed when the sender is sending a long chain or a series of chains and the receiver wants the transmission to be stopped temporarily. Often, the receiver needs to halt the transmissions because the receiver is running out of buffer space in which to store the incoming data. Another reason is to stop the incoming requests long enough to allow the receiver to send an informational request of its own.

To understand how quiescing works, consider the situation in which the receiver is running out of buffer space. Assume that this condition develops at the LU while the application program is in the middle of sending a chain to the LU. To tell the application program that it should stop sending data, the LU sends a ***Quiesce at End of Chain*** (QEC) request to the application program. The exact meaning of that request must have been worked out between the LU and the application program before the programs were coded. The request might mean "stop sending immediately and do not complete the chain normally" or it might mean "stop sending after you complete the current chain normally.". If it means "stop sending immediately," the application program can send a Cancel request or a special request (ending the chain) to tell the LU to discard the beginning of the chain. If the QEC request means "complete the chain normally before stopping," the application program continues sending requests until the chain is completed. In either case, the application program signals its compliance with the QEC request by sending a ***Quiesce Complete*** (QC) request to the LU. The LU then continues processing the previously received requests (perhaps by printing them or by writing them to disk storage).

When buffers are available to hold more incoming data, the LU sends a Release-Quiesce (RELQ) request to the application program. Upon receipt of that request, the application program recommences sending (either at the beginning of the aborted chain or at the beginning of a new chain, depending upon the agreed-upon protocol). Figure 54 on page 208 and Figure 55 on page 209 illustrate the use of quiescing to prevent buffer overflow.

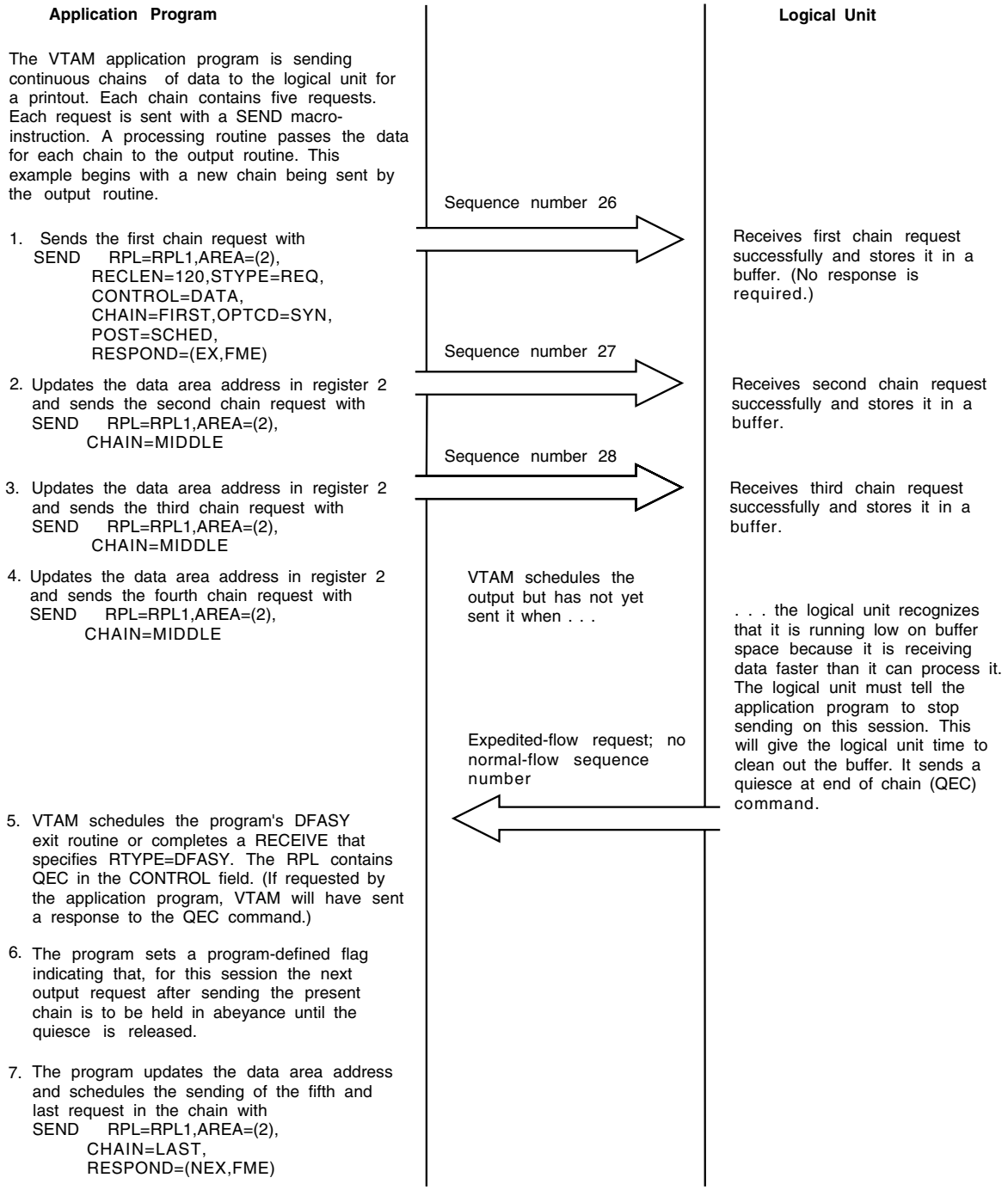


Figure 54. Example of an LU quiescing an application program in order to interrupt continuous sending (Part 1 of 2)

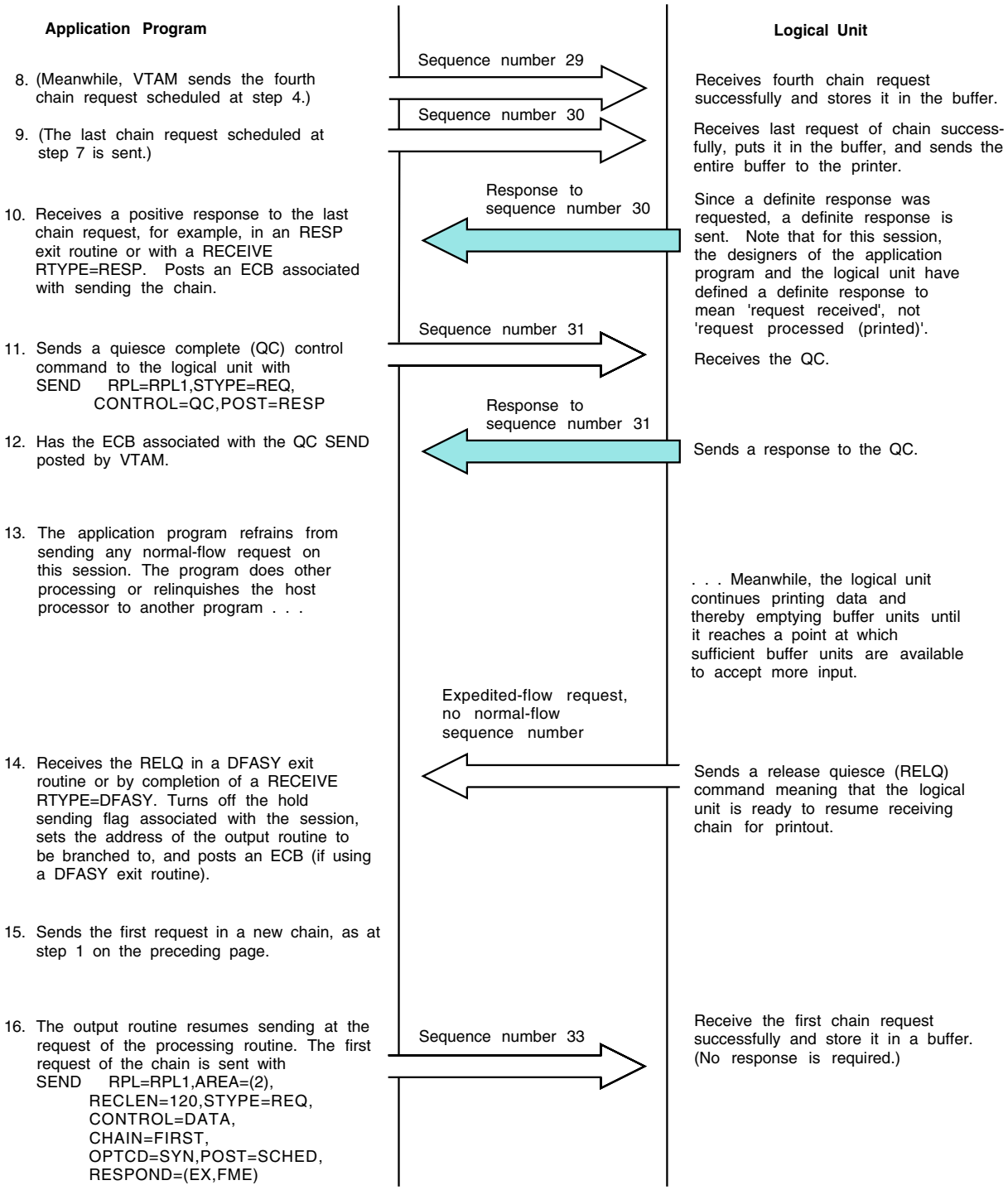


Figure 55. Example of an LU quiescing an application program in order to interrupt continuous sending (Part 2 of 2)

Protocols for ensuring orderly communications:

Certain types of LUs are limited in their communication with each other to specific directions of traffic flow. Some LUs can both send and receive, but can operate only in one direction at a time (half-duplex). Others can send and receive simultaneously (full duplex). These characteristics affect the selection of session parameters, which are sent by the PLU to the SLU when the session is established (see "Establishing parameters for sessions" on page 122). Other factors that affect

the selection of the session parameters are (1) the type of communication that takes place (interactive versus batch, for example) and (2) particular conventions that are agreed upon between programmers before the host application program and the LU program are written.

SNA provides several protocols that enable the application program and the LU to coordinate and control the direction of flow and their exchanges of requests and responses. None of these protocols are enforced by VTAM; VTAM sends the data-flow-control requests and indicators specified by the sender without checking them and without comparing them to the current status of communications. It is the responsibility of the application program and the LU to abide by the communication rules (the session parameters) they agreed upon when the session was established.

Quiesce protocol: As described in the preceding section, the quiesce requests can be used to temporarily stop the sender from sending when the receiver encounters a problem or special condition. Another use of the quiesce requests is to ensure that, at any one time, only one session partner can send normal-flow requests. This second use of the quiesce requests is called quiesce protocol.

In this protocol, one session partner controls the direction of flow by using the quiesce request to "turn off" normal-flow transmission by the other session partner. For example, assume that the application program is to control the direction of flow. When the application program has not quiesced the session, the LU is free to send normal-flow requests on the session. When the application program wants to start sending, it informs the LU by transmitting the Quiesce at End of Chain request on the expedited flow. On receipt of that request, the LU knows that it must stop sending normal-flow requests on the session when it completes sending the current chain. The LU also knows that the next normal-flow transmission comes from the application program. The application program then starts sending normal-flow requests and continues until it sends the Release Quiesce request to the LU. On receipt of that request, the LU knows that it can again start sending normal-flow requests on the session. In this way, the application program alternately grants the LU permission to send (by transmitting the Release Quiesce request) and stops the LU from sending (by transmitting the Quiesce at End of Chain request). The direction of flow can similarly be controlled by the LU.

Quiesce state applies only to normal-flow traffic. While a session partner is in quiesce state, it can send responses and expedited-flow requests on the session. Figure 56 on page 211 shows an example of quiesce protocol. See also Figure 115 on page 702.

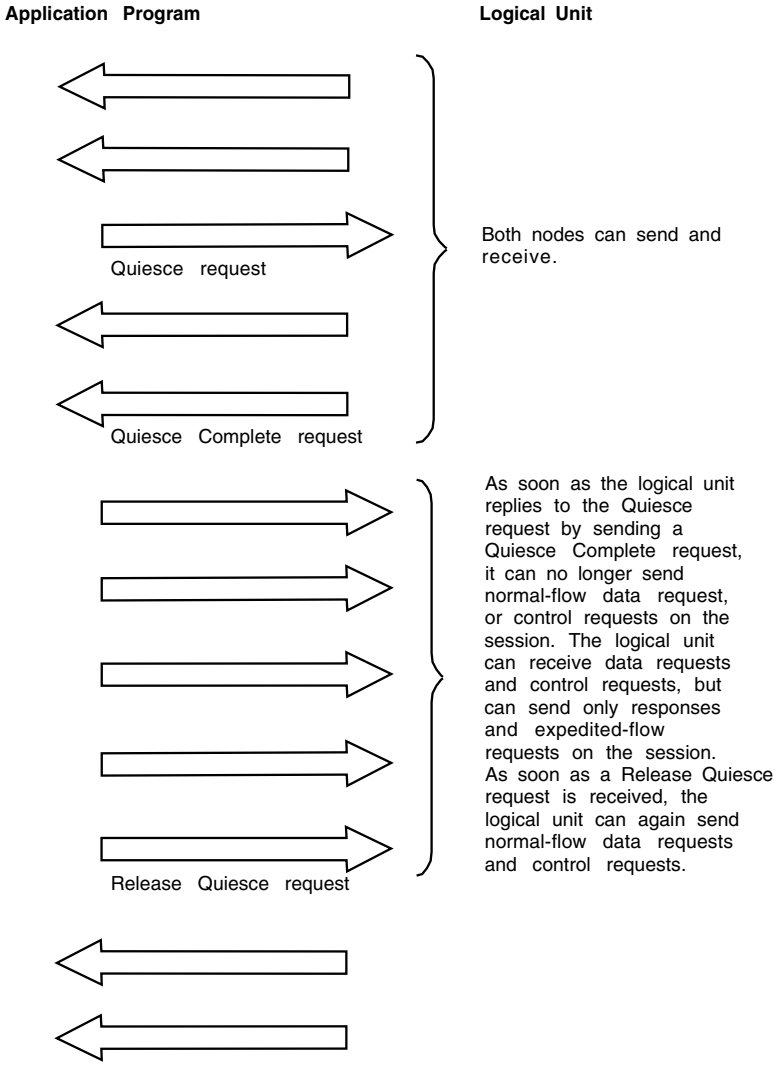


Figure 56. Quiesce protocol

Half-duplex protocols: There are two forms of half-duplex communication: half-duplex flip-flop communication and half-duplex contention communication. In half-duplex flip-flop communication, one session partner is designated in the session parameters as the first to send a normal-flow request after a session is established; thereafter, the program and the LU notify each other, in turn, by sending a change-direction indicator whenever the other side can begin sending normal-flow requests. In half-duplex contention communication, after the session has been established, the application program and the LU can both attempt to start sending a normal-flow request at the same time (called contention). The one that is allowed to proceed is the one that was designated in the session parameters as the one that would always win in a contention situation. Similarly, in contention communication, when either session partner finishes sending a chain of normal-flow requests, both session partners can attempt at the same time to start sending a new request. Again, the winner of the contention is the one designated as such in the session parameters.

One bit in the common protocol portion of the session parameters controls priority for initial sending in half-duplex communication. One setting of the bit indicates that the PLU has priority for sending; that is, in half-duplex flip-flop communication, the PLU is the first to send a normal-flow request in the session, or is the first to send a normal-flow request after a Clear request. Another bit in the common protocol portion of the session parameters governs whether the PLU or the SLU wins in a half-duplex contention race in which both try to send at the same time.

Change-direction protocol must be used in half-duplex flip-flop communication; the protocol can optionally be used in half-duplex contention communication.

Change-direction protocol works like this: The LU that is the first to send continues sending normal-flow requests until it reaches the end of the data it wants to send. In the last request of the last chain it sends, the sender turns on the change-direction indicator (CD). (A VTAM application program does this by issuing SEND CHNGDIR=CMD.) Upon receipt of this CD request, the other LU then sends normal-flow requests until it relinquishes its ability to send by including the CD in the last request of a chain.

Communication continues to alternate in this fashion indefinitely, as shown in Figure 57 on page 213.

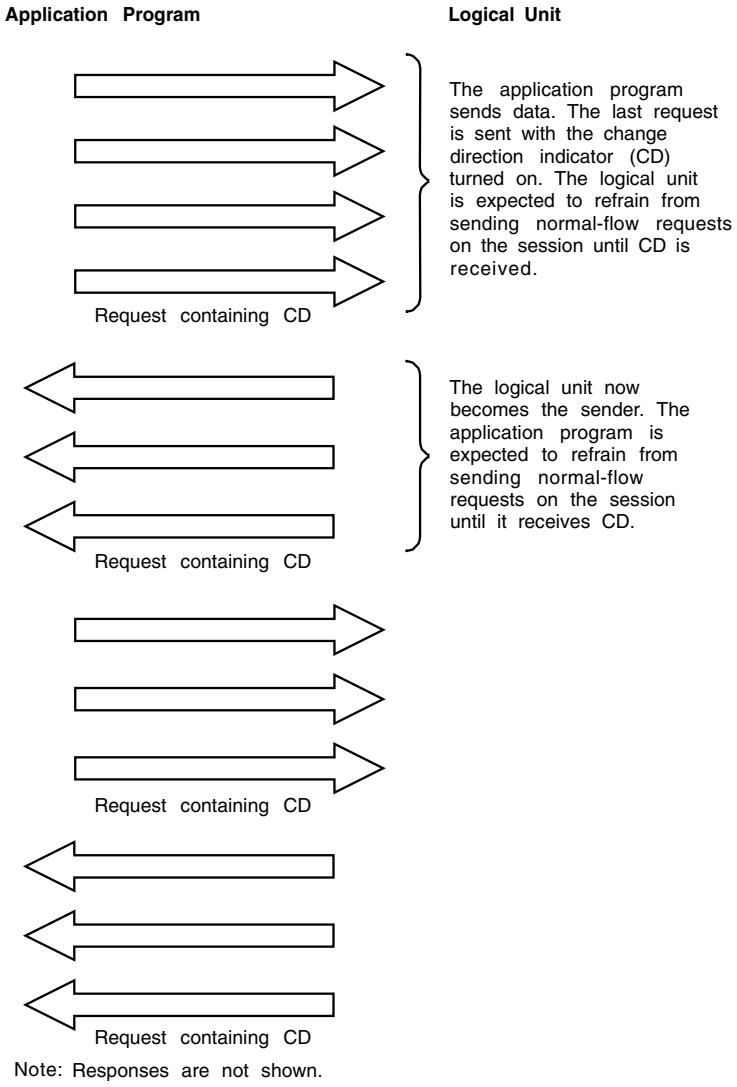


Figure 57. Change-direction protocol

While the receiver is awaiting CD, it can transmit to its session partner a Signal data-flow-control request with a Signal code that requests that CD be sent.

The LU that is awaiting CD (like the LU that has been quiesced in quiesce protocol) is prohibited only from sending normal-flow request traffic. It is free to send responses and expedited-flow requests.

As mentioned previously, VTAM does not enforce the change-direction protocol. Should the side waiting for CD begin sending data anyway, VTAM does not prevent the transmission. Compliance with the change-direction protocol is entirely the responsibility of the application program and the LU.

Bracket protocols: A bracket is any "uninterruptible" unit of work that an application program and an LU have been programmed to accomplish. A bracket can consist of any combination of data requests and data replies, ranging from a single request in one direction to an elaborate exchange of requests and replies. (Reply is used here to mean a data request sent in answer to a previously received data request.) But, no matter how simple or complex the series of requests and

replies can be, the characteristic that makes them all part of the same bracket is that they all pertain to the same unit of work.

A database inquiry transaction is a typical example of a bracket. In such a transaction, the LU sends an inquiry to the host processor asking for some piece or body of information stored in the database. For example, an insurance agent at a terminal asks the processor to provide information on all insurance policies issued to a particular client. In answer to the inquiry, the application program in the host processor sends a single request or a series of requests containing the requested information. At this point, the bracket might end. Or, as the result of one of the replies, the LU might ask for further details. In that case, the bracket does not end until the application program has acquired the details from the database and sent them to the LU.

Bracket protocols are used when one, or both, of the ends of the session cannot begin processing a new unit of work on the session until the current one has been completed. For example, it can be used if the LU or application program cannot start handling a new inquiry on the session until the replies to the current inquiry have been completed. Bracket protocols provide a way of ensuring that a new unit of work is not started until the preceding one has been finished.

The application program and LU that are using bracket protocols indicate for each chain whether that chain is the beginning, middle, or end of the bracket. This allows the receiving LU to determine whether a new bracket can be started. A begin bracket (BB) indicator is turned on in the first request of the first chain in a bracket. The end bracket (EB) indicator is turned on in the first request of the last chain in the bracket; alternatively, the conditional end bracket (CEB) indicator is turned on in the last request in the last chain in a bracket. (CEB can be specified only if FM profile 19 is used in the session.) A VTAM application program turns on these indicators by issuing SEND BRACKET=BB, SEND BRACKET=EB, and SEND BRACKET=CEB, respectively.

When a session is established, bits in the session parameters determine who wins bracket contention when both sides want to begin a bracket simultaneously, who can end a bracket, and whether bracket termination is conditional (the session partner sending EB does not consider the bracket ended until it receives a positive response to the chain that includes EB) or unconditional (termination occurs when the chain containing EB is sent). Figure 58 on page 215 shows an example of bracket protocol.

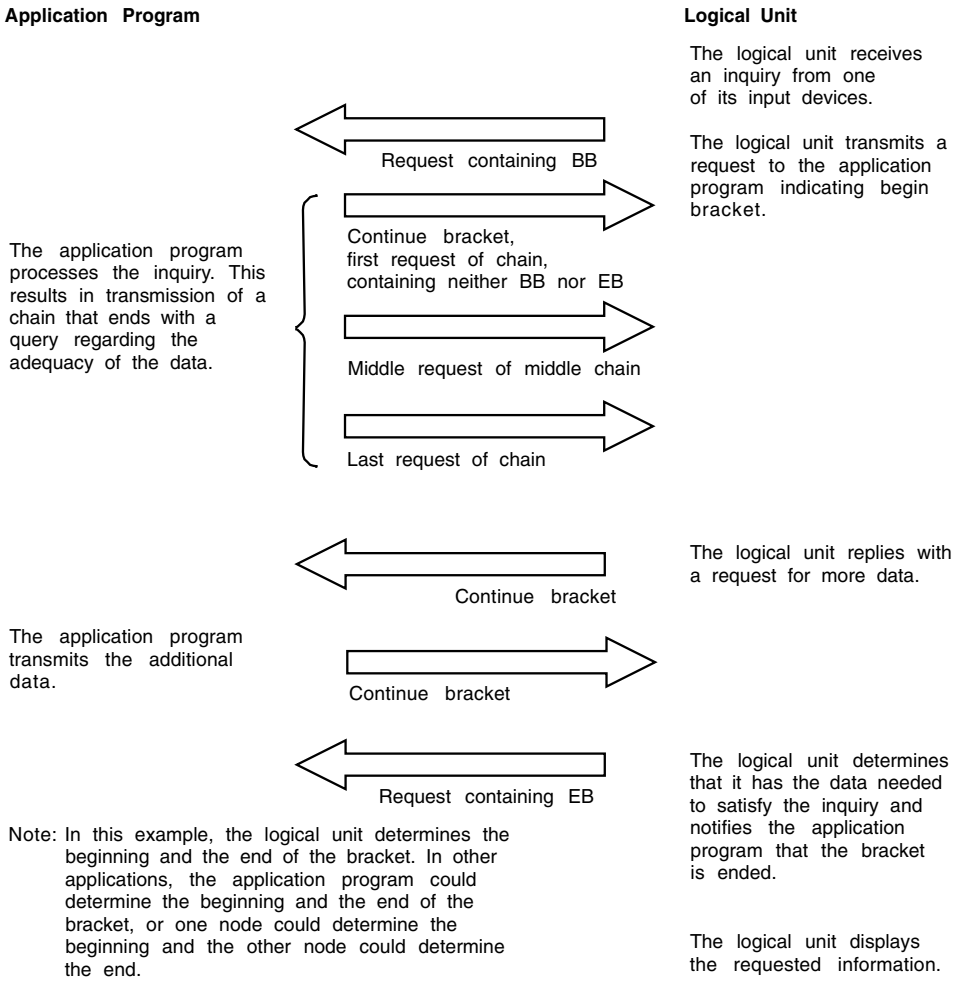


Figure 58. Bracket protocol

One bracket-related bit in the session parameters determines the winner of bracket contention by assigning the role of first speaker to one LU and the role of bidder to the other participant. The first speaker is the LU that is given the ability to begin a bracket without asking permission from the other LU in the session. The bidder is the LU that must ask for and receive permission from the first speaker to begin a bracket. The bit in the session parameters designates whether the PLU or the SLU is to be the first speaker; the other LU is automatically the bidder.

When a bracket is ended, the first speaker can start a new bracket if needed. The bidder, however, must ask permission to begin a bracket. The bidder can do this in either of two ways:

- The bidder can ask permission by sending a Bid data-flow-control request to the first speaker. A positive response to the Bid request indicates that the first speaker has granted permission. A negative response indicates that permission is denied. The negative response, however, can be accompanied by sense data that indicates whether the first speaker will or will not later grant the permission by sending a Ready to Receive data-flow-control request. On receipt of that request, the bidder can begin a bracket.
- The bidder can ask permission by starting to send a chain in which the first request contains BB. The response indicates whether the bidder can continue with the bracket, with a positive response indicating that it can continue and a

negative response indicating that the attempt was rejected. The negative response, however, can be accompanied by sense data that indicates whether the first speaker does or does not later send the Ready to Receive request. There are restrictions on attempting to begin a bracket by starting to send a chain with BB:

- If the bidder is sending only a single chain, the first request in the chain must specify BB and EB.
- If the bidder is sending multiple chains, the first request in the first chain must specify BB and the bidder must ask for a definite response to the first chain. If the bidder gets a negative response, it knows that its bracket-initiation request was rejected and that it must terminate the chain (either by sending the Cancel request or by sending a request marked last in chain).

Like quiesce and change-direction protocol, bracket protocol is not enforced by VTAM. It must be adhered to by the participants.

Special use of RESPOND=QRESP with bracket protocol:

Consider the following situation. The application program (which is the bidder) and the LU (the first speaker) are in a session involving half-duplex contention and the use of brackets. At this point, assume that they are within a bracket. They have agreed in the session parameters to conditional bracket termination; that is, a bracket is not terminated until the sender of EB gets a response to the chain containing EB.

The application program then sends a chain containing EB. Simultaneously, the LU sends a data request that was meant to be within the current bracket. If RESPOND=NQRESP is used for the EB chain in VTAM, it is possible for the application program to get the response to the EB chain before it gets the data request. If that happens, the application program fails to know that the LU meant for the data request to be within the bracket.

To avoid this problem, the application program should specify NIB PROC=ORDRESP and send the EB chain with SEND RESPOND=QRESP. That parameter causes VTAM to treat the response as if it were a normal-flow request. Because the LU must send the response to the request specifying EB after it sends the in-bracket request, and because the response is treated as a normal-flow request, the application program gets the in-bracket request and the response in that order, which is the correct order.

Because the response to the EB request is treated as a normal-flow request, it does not cause scheduling of an RESP exit routine, nor can it cause completion of RECEIVE RTYPE=RESP. The SEND macroinstructions used to send the chain containing EB and RESPOND=QRESP must specify POST=SCHED. The response itself (because it is treated as a normal-flow request) must be obtained with RECEIVE RTYPE=DFSYN (not RTYPE=RESP).

For more information on the QRESP and NQRESP parameters in the RESPOND operand, see “Controlling the handling of normal-flow responses” on page 161.

The Chase request:

The Chase data-flow-control request can be used by a PLU or an SLU at any point in its processing to ensure that it has received all responses from the session partner. When the session partner receives the Chase request, it must send any unsent responses to previous data requests or normal-flow data-flow-control

requests before it sends the response to the Chase request. Thus, when the Chase sender receives the response to the Chase request, the sender knows there are no outstanding responses for the session partner on that session.

The Chase request is frequently used before a session-termination request. For example, an SLU that has received a Shutdown request might issue a Chase request to get any outstanding responses from the PLU before issuing the Shutdown Complete request. For an example of this Shutdown Complete request, see Figure 122 on page 709. The Chase request can also be issued by a PLU before it issues a CLSDST macroinstruction.

Using the Chase request can cause a problem. When a response cannot be passed immediately to the application program, it is placed on a queue to await presentation to the program. If the application program sends a Chase request with POST=RESP, the SEND operation is posted complete as soon as the response to the Chase request is received by VTAM. If any responses were previously queued for the program, it can no longer be prepared to process them, having interpreted the response to the Chase request as an indication that all responses were received.

To avoid this problem, the session should be established with PROC=ORDRESP specified in the NIB, and the Chase request should be sent with a macroinstruction that specifies POST=SCHED:

```
SEND  RPL=RPL1,STYPE=REQ,CONTROL=CHASE,POST=SCHED,            C
      RESPOND=(FME,NEX)
```

The response to the Chase is thus handled in order with respect to other normal-flow responses. If any outstanding responses might have the QRESP indicator on, then the Chase must also be sent with RESPOND=(NEX,FME,QRESP) to ensure that the Chase response is received in order with other responses having QRESP on.

For more information on the QRESP and NQRESP parameters, see “Controlling the handling of normal-flow responses” on page 161.

Function management headers: The function management (FM) header option is specified through the RPL or SEND macroinstruction. Specifying OPTCD=FMHDR indicates that a user-defined or SNA-defined FM header is included in a data request sent to an LU. This option indicates to VTAM how the format bit in the request header (RH) of a specific data request or response is to be set. If FMHDR is coded, the format bit is set on in the RH and sent to the receiver of the request or response.

Similarly, if the format bit is on in the RH of a received request or response (indicating the presence of an FM header), it causes FMHDR to be set in the RPL used for the exit or RECEIVE operation. OPTCD=FMHDR is set in the RPL by VTAM whenever a data-flow-control request or data-flow-control response is sent or received. FMHDR can be tested with the TESTCB macroinstruction or by using the IFGRPL DSECT.

When the session is established, the application program and LU determine whether FM headers can be used for data requests. This is specified in the BIND session parameters. SNA protocols do not use the FM header indicator on data responses for LU-LU sessions. As with other data-flow-control protocols described in this chapter, VTAM does not enforce the way indicators are used.

Additional SNA protocol information:

In addition to the protocols described earlier in this chapter, the following protocols can be specified in the session parameters:

- Whether an LU can remove redundant characters (for example, blanks) before data is transmitted (compression)
- Who has error recovery responsibility
- Whether an alternate character code is acceptable (for example, ASCII instead of EBCDIC)

Sending and receiving enciphered data requests:

Sessions using cryptography can use selective, required, or private cryptography. VTAM determines the level of cryptography by examining the cryptographic requirements:

- Established by the VTAM operator in a MODIFY command
- In the logon mode table entry
- In the BIND request operands
- Specified by the ENCR parameters in the NIBs used by the PLU and the SLU to establish the session

See Table 20 on page 134 and Table 21 on page 135 for the relationships between the various methods of specifying the cryptography requirements.

For selective cryptographic sessions, data requests are enciphered on the basis of the CRYPT operand of the SEND macroinstruction. If CRYPT=YES is specified, VTAM enciphers the data request sent to the other LU of the session, and sets the request header RH-enciphered data bit on for the request. When VTAM receives an enciphered data request (one with the RH-enciphered data bit set), VTAM sets CRYPT=YES in the RPL specified by the RECEIVE macroinstruction and deciphers the data before passing it to the application program. If a SEND macroinstruction specifies selective data encryption, but the session does not support encryption, the macroinstruction will fail.

For required cryptographic sessions or conditional sessions (with the SLU capable of cryptography), the CRYPT operand is ignored; on SEND, all data requests are enciphered, and the RH-enciphered data bit is set for each one. VTAM sets CRYPT to YES in the RECEIVE RPL for each received data request and deciphers each request before passing it to the application program.

For a session that is not using selective or required cryptography, if CRYPT=YES is specified in a SEND macroinstruction, VTAM assumes that the sender is using a private (user-defined) form of enciphering and VTAM sends the data request to the other LU of the session without further enciphering; the RH-enciphered data bit is set. Similarly, on receipt of a request having the RH-enciphered data bit set, VTAM sets CRYPT=YES in the RECEIVE RPL. VTAM does not decipher such requests.

Note: This same CRYPT=YES bit is used by VTAM for selective or required cryptographic sessions. Therefore, if an application program also uses it for private cryptographic protocols, misleading results can occur. For example, the bit can be set because of a required level of session cryptography, yet the application program can interpret this as having been set because of private cryptographic protocols. Thus, the bit should not be used for private protocols if conflicts can occur with session-level cryptography.

Chapter 7. Using exit routines

This chapter discusses how exit routines work, presents some of their advantages and disadvantages, and describes procedures to follow in using them. Additional detailed considerations related to exit routines are discussed in Chapter 5, "Establishing and terminating sessions with logical units," on page 81, Chapter 6, "Communicating with logical units," on page 151, and Chapter 10, "Operating system facilities," on page 295. An overview of exit routines is given in Chapter 2, "VTAM language," on page 19. In particular, "Normal operating system environment for a VTAM application program" on page 30 should be read as background information for this chapter (for example, for definitions of inline exit routine, asynchronous exit routine, and mainline program).

How exit routines work

VTAM provides two general kinds of exit routines for use in a VTAM application program: RPL exit routines and exit-list (EXLST) exit routines. The two kinds of exit routines work somewhat differently, as described in the following sections. Additionally, the TESTCB manipulative macroinstruction provides an exit routine, which is described in "TESTCB—Test the contents of a control block field" on page 565. Other VTAM-supplied exit routines are not executed as part of an application program and are referred to as installation-wide exit routines rather than application program exit routines. Installation-wide exit routines (for example, those relating to session management, authorization and accounting) are described in z/OS Communications Server: SNA Customization.

RPL exit routines

The instructions executed when an RPL-based operation completes are written as a separate routine. This routine, called an RPL exit routine, is specified in the RPL-based macroinstruction that requests the operation. The address of the exit routine specified in the EXIT operand of the macroinstruction is placed in the EXIT field of the RPL. When the requested operation completes, VTAM schedules and eventually causes entry to the RPL exit routine. The RPL exit routine has control in the addressing mode of the application program at the time the request is issued. See Chapter 10, "Operating system facilities," on page 295, for information about addressing mode.

When VTAM gives control to the RPL exit routine, the routine usually cannot be interrupted even though other pending events are completed; the exit routine must return control to VTAM before VTAM can return control to other parts of the application program, including other RPL or EXLST exit routines that VTAM might have scheduled. However, when the following special conditions occur, another part of the program can be given control before the currently running RPL exit routine completes:

- A LERAD or SYNAD exit routine can be entered if an error or special condition occurs for an RPL-based request that either is issued and not accepted in the RPL exit routine or is checked in the RPL exit routine.
- An RPL exit routine can be interrupted to allow the TPEND exit routine to be entered with reason code 8.

© Copyright IBM Corp. 2000, 2013

- An application program can use certain operating system facilities (multiple tasks and SRBs) to allow the application program to have several parts that can run concurrently.

See Chapter 10, “Operating system facilities,” on page 295 for more information about the LERAD, SYNAD, and TPEND exit routines and SRBs.

Designating a routine as an RPL exit routine is an alternative to having VTAM post an ECB when an asynchronous event is completed. A program can use either technique exclusively, or it can use a mixture of ECB-posting and RPL exit routines. Sample Program 1 in Chapter 14, “Logic of a simple application program,” on page 573, shows an example of an RPL exit routine. The same RPL exit routine can be designated by more than one macroinstruction; that is, an RPL exit routine can be established as a common exit routine.

If the application program also uses ECBs, the RPL exit routine can post an ECB related to the associated event so that the mainline program later discovers that an operation has been completed. Because it might be necessary to reuse the RPL associated with the request whose completion caused entry to the exit routine (for example, for reissuance of a RECEIVE request within the exit routine), and because it is a means of causing entry to a LERAD or SYNAD exit routine if an error occurs, a CHECK macroinstruction can be required in the exit routine. If the RPL does not have to be reused in the exit routine, the CHECK macroinstruction can be in the mainline program, perhaps following the discovery of a posted ECB associated with the RPL. Figure 59 on page 221 illustrates the use of an RPL exit routine.

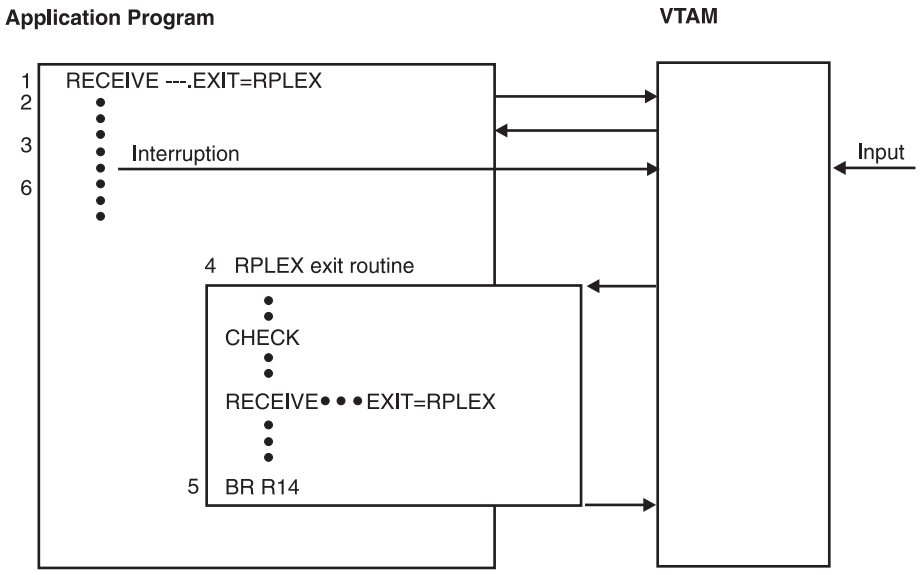


Figure 59. Example of using an RPL exit routine

The following items match the steps indicated in Figure 59:

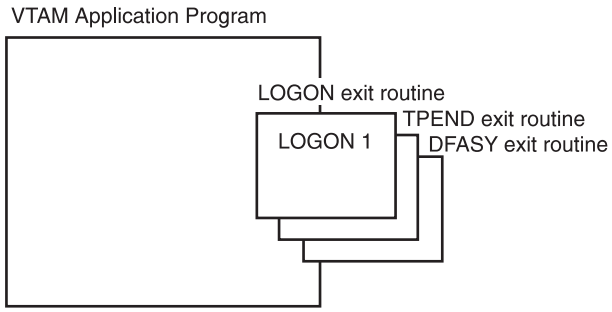
1. The VTAM application program issues an asynchronous RECEIVE request, which the program passes to VTAM. The request specifies the scheduling of the RPLEX exit routine when the operation completes. VTAM accepts the request and returns control to the program at the next sequential instruction.
2. The program continues execution until input arrives.
3. VTAM interrupts the program when input arrives and schedules RPLEX as the next routine. Because an asynchronous exit routine is not currently executing, RPLEX is immediately given control (4).
4. RPLEX executes without any other part of the program gaining control. RPLEX issues a CHECK macroinstruction to mark the RPL inactive. RPLEX then issues a RECEIVE macroinstruction to read input again. It is an asynchronous request specifying that RPLEX be scheduled when the operation completes. (If more input arrives and the operation completes, VTAM schedules RPLEX, but it is not reentered until after it finishes the current invocation and returns control to VTAM.)
5. Once RPLEX completes its job, control returns to VTAM.
6. If input to satisfy the receive in RPLEX has not yet arrived, VTAM returns control to the mainline program interrupted at 3. If input has arrived, the RECEIVE in RPLEX completes, and control is given to RPLEX.

EXLST exit routines

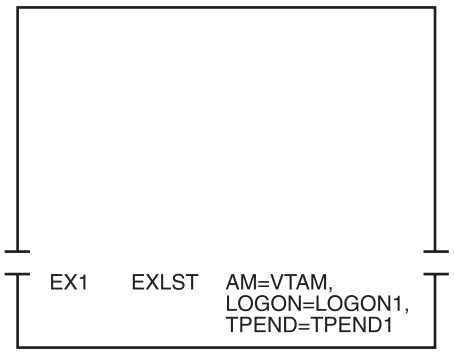
This type of exit routine differs from the RPL exit routine in that it is a special-purpose exit routine. The special purpose is understood by both the VTAM application program and VTAM. Instead of being specified in a particular RPL-based macroinstruction request, the identity of an EXLST exit routine is established only when the exit list in which its name is specified is identified to VTAM, either when the program is opened or, for certain types of exit routines, when a session is established. In general, EXLST exit routines are special-purpose exit routines, entered only when a somewhat unusual event occurs, such as the VTAM operator's issuance of a HALT command to shut down the network.

Here is how EXLST exit routines work:

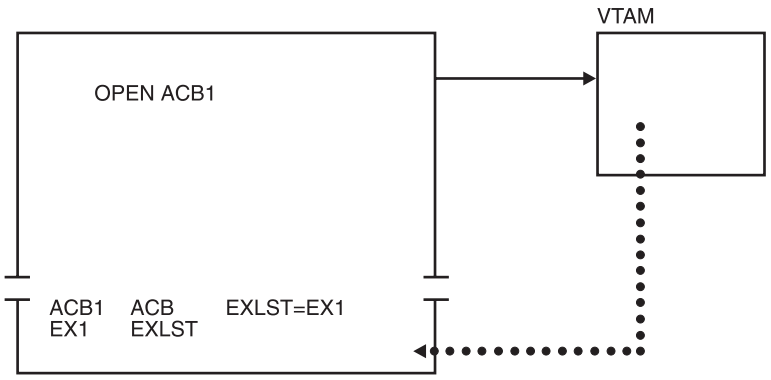
1. A VTAM application program contains a number of exit routines written for different purposes (for example, a LOGON exit routine and a TPEND exit routine).



2. The program names the special-purpose exit routines and puts their names in an exit list. The exit list is created with the EXLST macroinstruction. Each exit-routine name is specified with an appropriate VTAM-provided keyword, such as LOGON and TPEND.

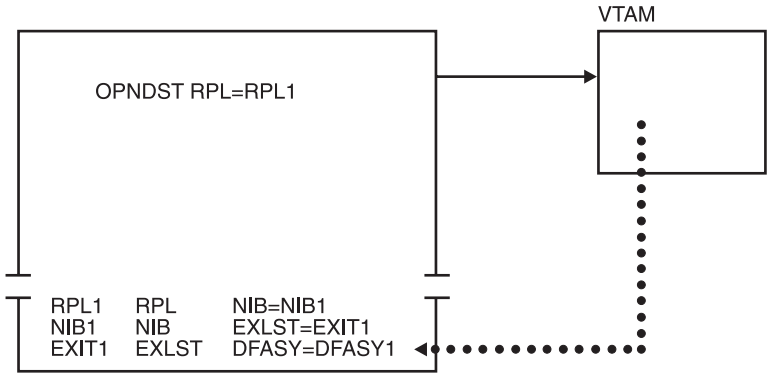


3. This exit list, identified by the name of the EXLST macroinstruction, can be specified in the EXLST operand of the program's ACB. When the ACB is opened, the list of exit routines becomes available to VTAM.



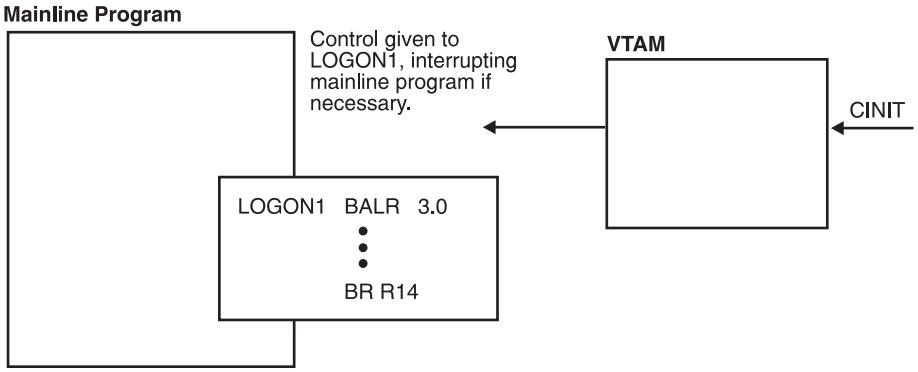
4. Alternatively, certain types of exit routines—DFASY, SCIP, and RESP—can be listed in the EXLST macroinstruction specified in the EXLST operand of the NIB that is used when a session is established. After the session is established (that is, after OPNDST or OPNSEC has been completed), VTAM uses an exit routine identified in the NIB exit list in preference to the corresponding exit routine specified in the ACB exit list. The preference applies only for the session. If an appropriate exit routine is not in the exit list established for the session, VTAM looks in the ACB-specified exit list that is specified when the

ACB is opened. (For more qualifying details, see “Specifying the DFASY, RESP, and SCIP exit routines in an ACB or NIB” on page 229.) An NIB EXLST cannot be altered or freed until the last session using it terminates. An ACB EXLST cannot be altered or freed until the ACB is closed.



5. When an event occurs for which a related EXLST exit routine exists, VTAM schedules the appropriate exit routine, using the exit routine provided in EXLST. As soon as no other asynchronous exit routine is being executed, the EXLST exit routine is given control (if necessary, interrupting the mainline portion of the program). As described in “TPEND exit routine” on page 263, the TPEND exit routine, with reason code 8, is handled in a special way.

Note: The scheduling of the LOGON exit (for CINITs) or SCIP exit (for BINDs) can be delayed by using the SETLOGON macroinstruction.



When the ACB opens, control is given to an asynchronous EXLST exit routine in the addressing mode of the application program. (Asynchronous EXLST exit routines include all EXLST exit routines except SYNAD and LERAD.)

When VTAM gives control to an asynchronous EXLST exit routine, the routine usually cannot be interrupted even though other pending events are completed; the exit routine must return control to VTAM before VTAM can give control to other parts of the application program, including other EXLST or RPL exit routines that VTAM might have scheduled. However, when the following special conditions occur, another asynchronous exit routine can be given control before the currently running EXLST exit routine completes:

- A LERAD or SYNAD exit routine can be entered if an error or special condition occurs for an RPL-based request that either is issued and not accepted in the asynchronous EXLST exit routine or is checked in the asynchronous EXLST exit routine.
- An asynchronous EXLST exit routine can be interrupted to allow the TPEND exit routine to be entered with reason code 8.
- An application program can use certain operating system facilities (multiple tasks, and SRBs) to allow the application program to have several parts that can run concurrently.

See Chapter 10, "Operating system facilities," on page 295 for more information about the LERAD, SYNAD, and TPEND exit routines and SRBs.

The SYNAD and LERAD exit routines, called inline exit routines, obey different rules than the asynchronous EXLST exit routines. SYNAD and LERAD operate under the same operating system scheduling control block (for example, a TCB or SRB) as the part of the program that issued the RPL-based or CHECK macroinstruction whose issuance caused the SYNAD or LERAD exit routine to be invoked. Thus, SYNAD and LERAD operate essentially as extensions to that part of the program, and are subject to exactly the same interruption conditions as that part of the program.

When the application program issues CHECK, the SYNAD and LERAD exit routines are entered in the addressing mode of the application program. For requests with OPTCD=SYN, the addressing mode is in the same mode as the application program that issued the original request.

Summary of exit routines

Table 30 summarizes exit routines by showing the purpose of each type of exit routine and showing how the address of each type is specified to VTAM. Table 31 on page 226 summarizes the parameter list pointed to by register 1 upon invocation of each EXLST exit routine.

Table 30. Summary of exit routines

Type of exit routine	Purpose	How the exit routine's address is specified	Type of exit list in which that routine's name can appear
RPL exit routine	Notify the application program of the completion of an RPL-based macroinstruction.	Code the address in the EXIT operand of an RPL macroinstruction or in the request that uses the RPL.	Not applicable
EXLST exit routines (Each type is listed in the following.)	Special purposes.	Code the names and addresses of the exit routines in an EXLST macroinstruction. The list that is created is then identified in the EXLST operand of an ACB or NIB macroinstruction.	
ATTN **Note:** Scheduled only for applications that use VTAM's API for LU 6.2	Handle LU 6.2 events such as receiving notification of VTAM's negotiation of a CNOS or deactivating an LU 6.2 session.	Code the address in the ATTN operand of the EXLST macroinstruction.	ACB only

Table 30. Summary of exit routines (continued)

Type of exit routine	Purpose	How the exit routine's address is specified	Type of exit list in which that routine's name can appear
DFASY	Receive expedited-flow data-flow-control input (for example, a Quiesce at End of Chain request) without requiring an outstanding RECEIVE RTYPE=DFASY.	Code the address in the DFASY operand of the EXLST macroinstruction.	ACB or NIB
LERAD	Handle logic errors that can occur as the result of an RPL-based macroinstruction.	Code the address in the LERAD operand of the EXLST macroinstruction.	ACB only
LOGON	Handle a request for a session with the application program (a CINIT request in which the application program acts as the primary logical unit (PLU)).	Code the address in the LOGON operand of the EXLST macroinstruction.	ACB only
LOSTERM	Handle the situation of a session being unexpectedly lost to the program, or notify the application program of other unusual conditions that can affect the session.	Code the address in the LOSTERM operand of the EXLST macroinstruction.	ACB only
NSEXIT	Handle a situation in which: 1. A request for a procedure has been positively responded to, but the procedure cannot be completed 2. VTAM has initiated session termination because of a session outage 3. Some other kind of network services request unit is received.	Code the address in the NSEXIT operand of the EXLST macroinstruction.	ACB only
RELREQ	Handle a request from another application program for an LU that is presently in session with the program that contains the RELREQ exit routine.	Code the address in the RELREQ operand of the EXLST macroinstruction.	ACB only
RESP	Receive a response without requiring an outstanding RECEIVE RTYPE=RESP.	Code the address in the RESP operand of the EXLST macroinstruction.	ACB or NIB
SCIP	Receive and process one of the following session-control requests: • Clear • Start Data Traffic (SDT) • Request Recovery (RQR) • Set and Test Sequence Number (STSN) • Bind Session (BIND) • Unbind Session (UNBIND).	Code the address in the SCIP operand of the EXLST macroinstruction.	ACB or NIB
SYNAD	Handle a physical error or special condition that occurs as the result of an RPL-based macroinstruction.	Code the address in the SYNAD operand of the EXLST macroinstruction.	ACB only

Table 30. Summary of exit routines (continued)

Type of exit routine	Purpose	How the exit routine's address is specified	Type of exit list in which that routine's name can appear
TPEND	Notify the application program when the VTAM operator halts VTAM or deactivates the application program, when VTAM terminates abnormally, or when an alternate application takes over the sessions of an application that has enabled persistence.	Code the address in the TPEND operand of the EXLST macroinstruction.	ACB only

Table 31. Parameter list for the EXLST exit routines

Exit routine	Register 1 parameter list						
	1st word	2nd word	3rd word	4th word	5th word	6th word	7th word
ATTN [5]	ACB address	Reserved	Reserved	Event for which exit routine is being driven	Address of read-only RPL	Reserved	Address of network identifier parameter list [8]
DFASY	ACB address	CID	USERFLD data [4]	Reserved	Address of read-only RPL	Reserved	Reserved
LERAD	None (Register 1 contains the RPL address for the request that failed.)						
LOGON	ACB address	Address of the SLU's symbolic name [6]	USERFLD data or zeros [2]	Length of logon message	Address of read-only RPL (RPL contains the address of the CINIT RU)	CID	Address of network identifier parameter list [8]
LOSTERM (session notification)	ACB address	CID	USERFLD data [4]	Reason code	Reserved	Reserved	Reserved
LOSTERM (cross memory macro request failure notification)	ACB address	Address of the user RPL	Reserved	Reason code	Reserved	Reserved	Reserved
NSEXIT (for CLEANUP RU)	ACB address	CID	USERFLD data [4]	Reserved	Address of read-only RPL (RPL contains the address of the CLEANUP RU)	Reserved	Reserved

Table 31. Parameter list for the EXLST exit routines (continued)

Exit routine	Register 1 parameter list						
	1st word	**2nd word**	**3rd word**	**4th word**	**5th word**	**6th word**	**7th word**
NSEXIT (for Notify RU)	ACB address	Reserved	USERFLD data [3]	Reserved	Address of read-only RPL (RPL contains the address of the Notify RU)	Reserved	Address of network identifier parameter list [8]
NSEXIT (for NSPE RU)	ACB address	Reserved	Reserved	Reserved	Address of read-only RPL (RPL contains the address of the NSPE RU)	Reserved	Reserved
RELREQ	ACB address	Address of the SLU's symbolic name	Reserved	Reserved	Reserved	Reserved	Address of network identifier parameter list [8]
RESP	ACB address	CID	USERFLD data [4]	Reserved	Address of read-only RPL	Reserved	Reserved
SCIP (for BIND RU)	ACB address	CID	USERFLD data or zeros [1]	Address of the session parameters	Address of read-only RPL (RPL contains the address of the BIND RU)	Address of the PLU's symbolic name [7]	Address of network identifier parameter list [8]
SCIP (for other than BIND RU)	ACB address	CID	USERFLD data [4]	Reserved	Address of read-only RPL (for UNBIND, RPL contains the address of UNBIND RU)	Reserved	Reserved
SYNAD	None (Register 1 contains the RPL address for the request that failed.)						
TPEND	ACB address	Reason code	Reserved	Reserved	Reserved	Reserved	Reserved

Table 31. Parameter list for the EXLST exit routines (continued)

Exit routine	Register 1 parameter list						
	1st word	2nd word	3rd word	4th word	5th word	6th word	7th word

Notes:

1. If the BIND request is a result of a REQSESS macroinstruction, word 3 contains the USERFLD data from the NIB used with REQSESS; otherwise, word 3 contains zeros.
2. If the LOGON exit routine is entered as a result of a SIMLOGON macroinstruction, word 3 contains the USERFLD data from the NIB used with SIMLOGON; otherwise, word 3 contains zeros.
3. Word 3 contains the USERFLD data from the NIB used with REQSESS, SIMLOGON, or CLSDST OPTCD=PASS.
4. Word 3 contains the USERFLD data from the NIB used with OPNDST or OPNSEC.
5. For more information, refer to the z/OS Communications Server: SNA Programmer's LU 6.2 Guide.
6. This name is the LUALIAS name or the 8–byte name of the SLU taken from the CINIT.
7. This name is the LUALIAS name or the name in the NSPLU name field of the BIND.
8. The network identifier parameter list is mapped by the ISTNRIPL DSECT. For more information, see Table 117 on page 762.

Deciding whether and how to use exit routines

In general, the use of exit routines is optional. An RPL exit routine is an alternative to having a routine that is branched to in the mainline program following the posting of an ECB by VTAM. Refer to "Using RPL exit routines" on page 42 for a discussion of these two alternatives. Most EXLST exit routines are optional, though some are designed for common use and should be included in an application program. The LOSTERM, NSEXIT, SCIP, and TPEND exit routines are strongly recommended because, without them, the application program might not be notified of certain important events. The SCIP exit routine is required by any application program that acts as a secondary logical unit (SLU).

The following EXLST exit routines are designed for common use:

- LERAD
- LOGON
- LOSTERM
- NSEXIT
- SCIP
- SYNAD
- TPEND.

The RELREQ exit routine is required only if the application program is to be notified when another logical unit (LU) requests a session with an LU that is in session with the application program and is at its session limit.

These EXLST exit routines are optional alternatives to other facilities:

- DFASY, rather than having to issue RECEIVE RTYPE=DFASY in the mainline program and branching to a related routine on completion
- RESP, rather than having to issue RECEIVE RTYPE=RESP in the mainline program and branching to a related routine on completion.

If an EXLST exit routine is not provided and the event or condition that the exit routine would handle does occur, the application program may never be informed

about the event or condition. In some cases, the application program might learn of the event or condition through return codes or information in an RPL when it is posted complete.

Note: Only exit routines that can be recognized by VTAM can be specified in the EXLST macroinstruction. Non-VTAM exit routines (such as VSAM exit routines) cannot be specified in the macroinstruction.

Specifying the DFASY, RESP, and SCIP exit routines in an ACB or NIB

The DFASY, RESP, and SCIP EXLST exit routines can be in an exit list that is associated either with an ACB (identified by the EXLST operand of an ACB) or with a NIB (identified by the EXLST operand of an NIB). VTAM uses ACB-specified exit routines for all sessions with the program represented by the ACB. VTAM uses NIB-specified exit routines only for each session whose NIB specifies the exit routine when the session is established. With one exception, the DFASY, RESP, and SCIP exit routines identified in an NIB exit list are scheduled instead of corresponding exit routines identified in an ACB exit list. This exception is in the processing of a BIND request; the ACB-specified SCIP exit routine is always scheduled in this case. For details on how VTAM classifies DFASY and RESP input, and decides to schedule a RESP or DFASY (NIB or ACB) exit routine instead of completing RECEIVE, see Figure 39 on page 179 through Figure 42 on page 182. Several sessions can share the same list of DFASY, SCIP, and RESP exit routines, or the list can be unique for each session.

Figure 60 on page 230 shows two sets of NIB-specified exit routine addresses. When input from the session associated with NIB1 arrives, the appropriate EXLST1 exit routine is scheduled. When input from the session associated with NIB2 arrives, VTAM checks EXLST2 for the appropriate exit routine. If no exit routine is specified (which in this example would be true if the input is a response, because EXLST2 has no RESP entry), VTAM satisfies any pending RECEIVE macroinstructions or checks for an ACB-specified exit routine address in EXLSTA. When input from any other session arrives, VTAM uses EXLSTA.

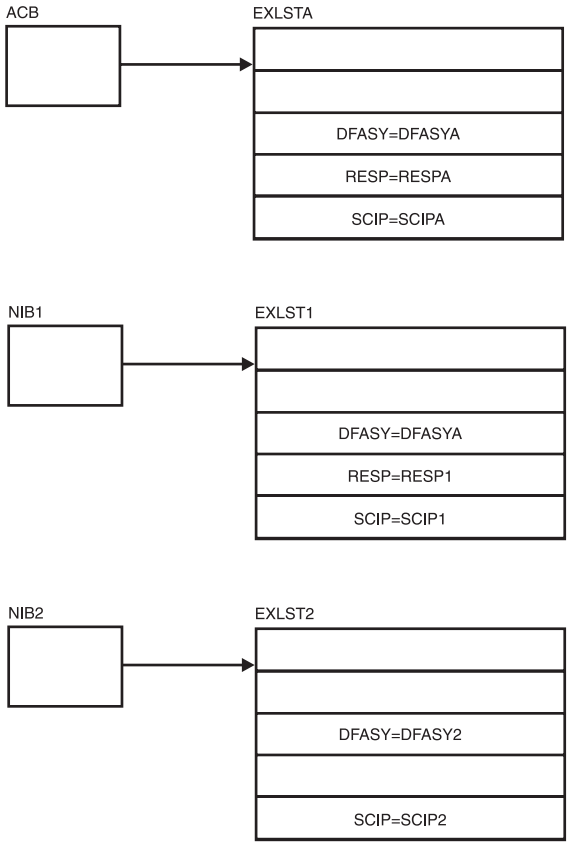


Figure 60. ACB-oriented and NIB-oriented exit routines

Special requirements for LERAD and SYNAD exit routines

The optional LERAD and SYNAD exit routines improve a program's error-handling capabilities. After a macroinstruction that specifies an RPL is issued, one of these two exit routines, if present, is entered if an error occurs. If the exit routine does not exist, VTAM, in any case, provides feedback information in registers 0 and 15 and in appropriate RPL fields. The return code in register 0 enables the next sequential instruction in the program to determine whether a logic error or one of several other general types of errors occurred; the program can itself then branch to an appropriate routine. The chief advantage of using LERAD and SYNAD exit routines is that they provide a convenient way to organize sets of error and special-condition-handling logic that serve all requests in the program.

The same name can be specified for the program's LERAD and SYNAD exit routines. The common exit routine can determine, after it is entered, whether a logic error or some other error or special condition occurred.

"Procedures to follow in writing exit routines" on page 231 discusses re-entrance requirements for LERAD and SYNAD exit routines.

A discussion of the kinds of logic that these routines might contain, as well as a detailed flow of how they are invoked, is provided in Chapter 9, "Handling errors and special conditions," on page 277.

Exit scheduling versus ECB posting

An asynchronous exit routine runs at a higher dispatching priority than the mainline part of the application program. Thus, if an application program uses ECB posting for a macroinstruction (OPTCD=SYN), the program can be notified of an event that causes exit routine scheduling before it recognizes that the macroinstruction has been posted complete. This can happen even though the event causing the exit routine to be scheduled occurs after the macroinstruction is posted complete. Here are a few examples of this condition:

1. SIMLOGON is issued and its ECB is posted complete with (RTNCD,FDB2)=(X'00',X'00') set in the RPL. However, before the application program recognizes this, the LOGON exit routine (CINIT received) or NSEXIT exit routine (Notify or NSPE received) starts to execute.
2. SESSIONC is issued to send SDT; the ECB is posted complete with (RTNCD,FDB2)=(X'00',X'00') set in the RPL. However, before the application program recognizes this, the DFASY exit routine receives a SIGNAL request.
3. OPNDST OPTCD=ACQUIRE is issued and its ECB is posted complete with (RTNCD,FDB2)=(X'00',X'00') in the RPL. However, before the application program recognizes this, the NSEXIT exit routine starts to run because a CLEANUP RU is received, perhaps because of a path outage in the session just established.

Either the application program must be prepared to handle situations like these or else it should be coded to avoid them. One such way to code, in the normal operating system environment, is to use an RPL exit routine instead of an ECB post; then, the RPL exit routine runs before the exit routine that notifies the application program of the second event. Another useful facility is the user correlation field available with SIMLOGON and certain other macroinstructions; this allows the application program to relate certain exit routine events (such as receipt of CINIT or Notify) with the original macroinstruction, even though the macroinstruction completion posting has not yet been recognized.

Procedures to follow in writing exit routines

Figure 61 on page 232 summarizes addressability and save-area requirements for the mainline program and exit routines. In most cases, LERAD and SYNAD exit routines do not have to be reentrant. See Figure 62 on page 233 for those cases. See Figure 63 on page 234 for the situations in which exit routines must be reentrant.

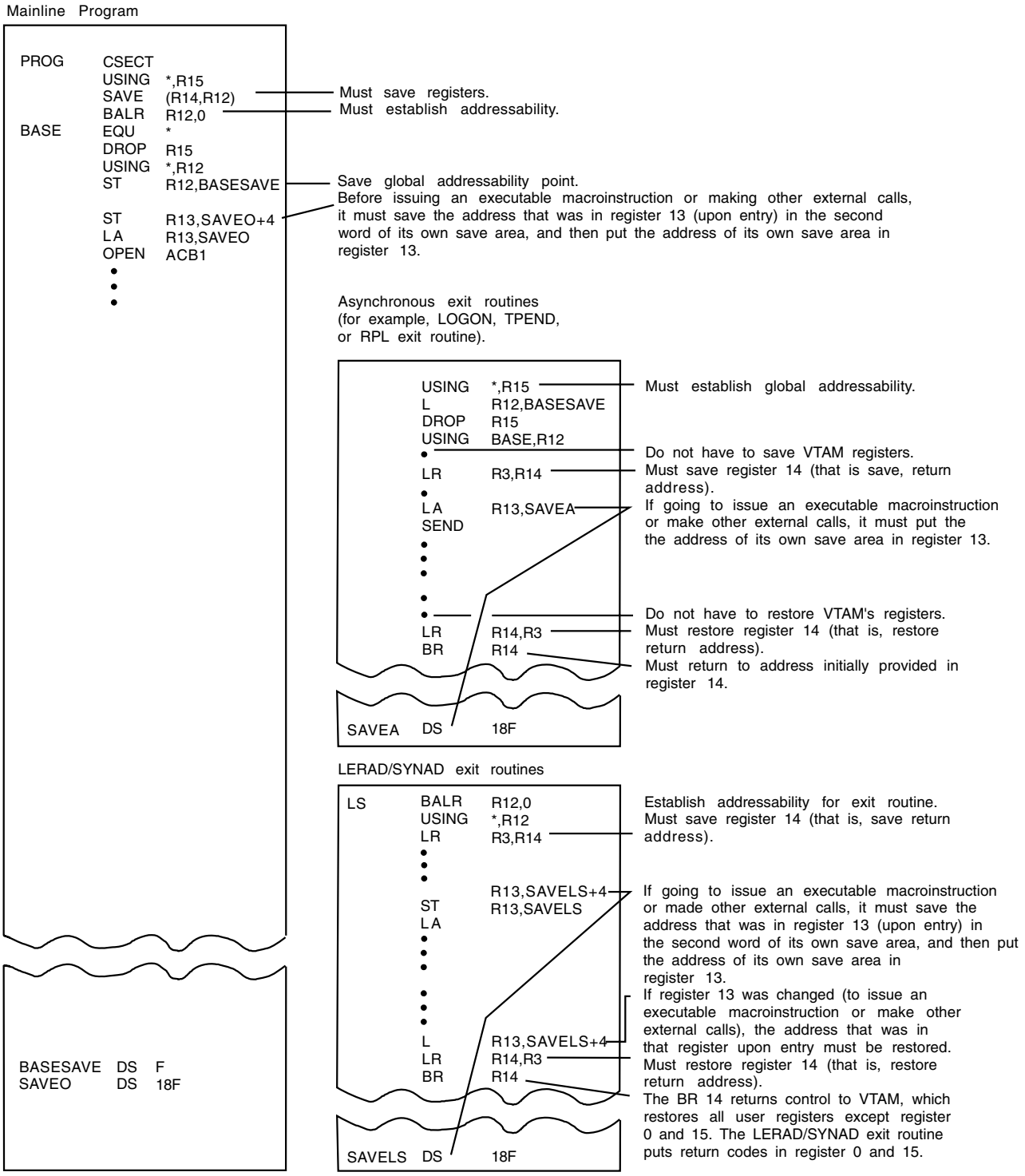
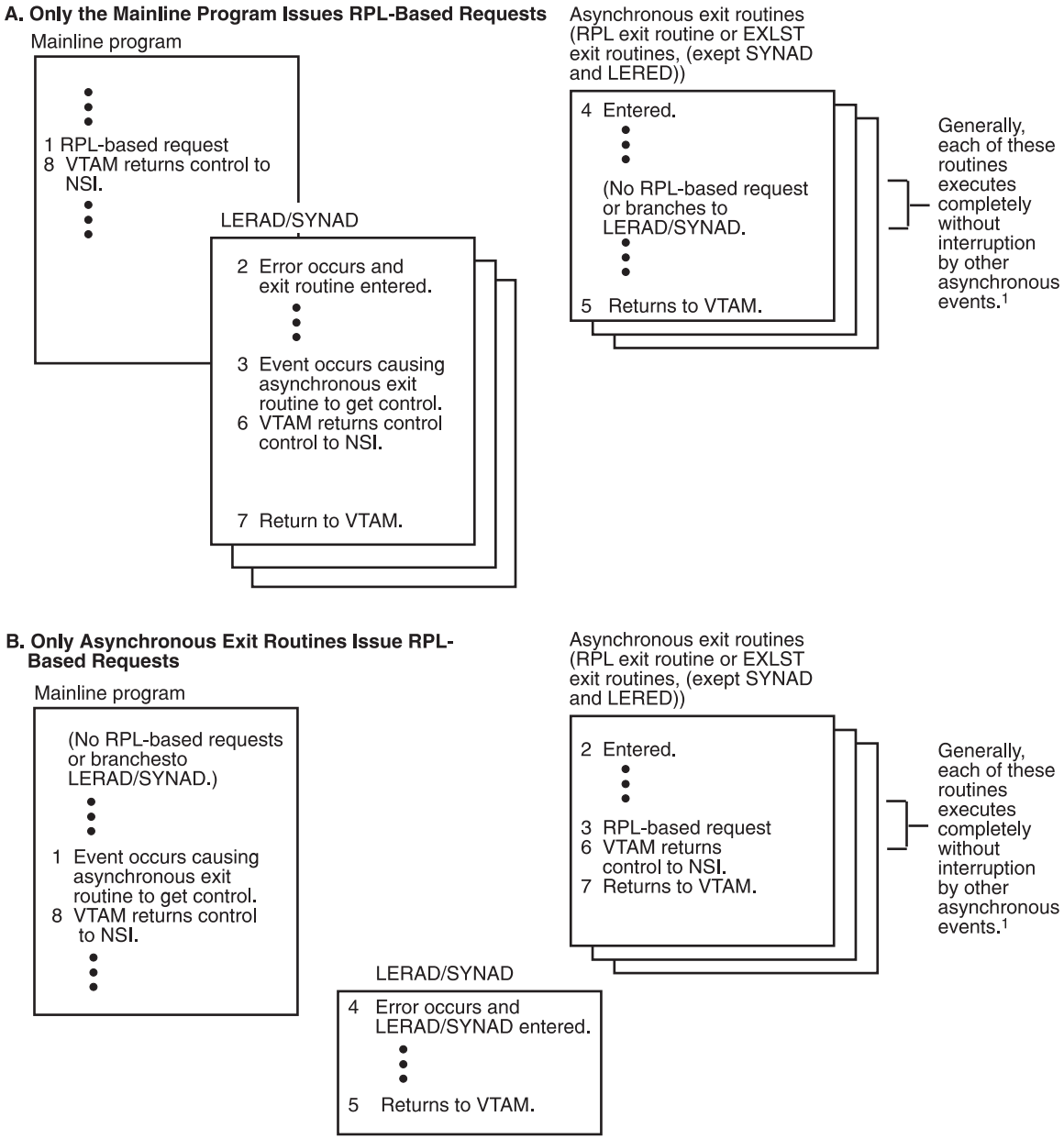


Figure 61. Summary of addressability and save-area requirements for the mainline program



[1] For circumstances in which a second exit routine can run before the first completes, see the section titled "How Exit Routines Work" earlier in this chapter.

Figure 62. Situations in which LERAD and SYNAD exit routines do not have to be reentrant

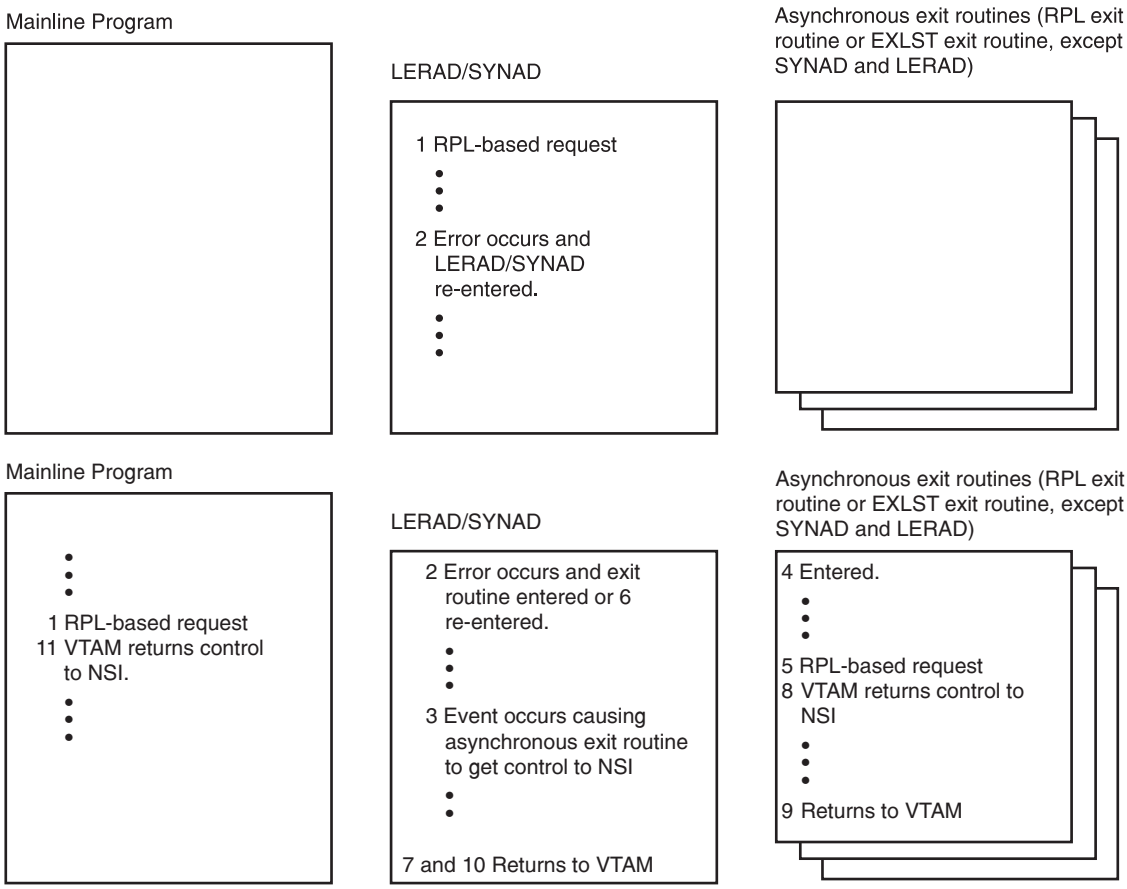


Figure 63. Situations in which LERAD and SYNAD exit routines must be reentrant

Entry procedures

In general, when an exit routine is entered, the following apply:

- Register 1 contains the address of a parameter list. (Table 31 on page 226 summarizes the parameter list for the EXLST exit routines.) This parameter list is freed when the exit routine returns control to VTAM.
- Register 14 contains an address for returning control after the exit routine completes. Unless stated otherwise in the discussions of the specific exit routines, VTAM returns control to the instruction in the application program that is about to be executed when the exit routine interruption occurs.

  If the exit routine is running under an SRB, different conditions apply for return. See “Execution of exit routines” on page 307.
- VTAM's registers do not have to be saved; register 13 does not contain the address of a VTAM save area. However, when a LERAD or SYNAD exit routine is entered, register 13 does contain the address of an 18-word save area supplied by the application program.
- VTAM does not provide a save area for the application program's general registers. If any executable VTAM macroinstruction is issued in the exit routine, the address of the exit routine's own 18-word save area must be in register 13 when the macroinstruction is issued.

Cautions, restrictions and techniques

When writing exit routines, consider these cautions, restrictions, and techniques:

- Establish addressability for each exit routine and for all of the storage that the exit routine uses. The two techniques for addressing control blocks are:
  - Define constants and literals that are within the range of the USING statement by using LTORG.
  - Use A-type address constants for storage that must be shared among the mainline program and exit routines or that cannot economically be duplicated (for example, save areas and the ACB).
- Do not use the OPEN and CLOSE macroinstructions in an exit routine.
- Do not use the MIBConnect or MIBDisconnect functions in an exit routine.
- Although most exit routines cannot be interrupted to be reentered, some exceptions exist (see "How exit routines work" on page 219). A LERAD or SYNAD exit routine, in some cases, must be reentrant (see Figure 63 on page 234). In a reentrant exit routine, storage must be obtained dynamically for control blocks (an RPL, for example) and data.
- Do not reuse save areas still in use by other parts of the application program. For example, the save area used by the mainline program when an RPL-based macroinstruction is issued is in use until VTAM returns to the mainline program; it should not be used by an asynchronous exit routine in the meantime.
- If a VTAM macroinstruction is issued within an exit routine identified in an ACB exit list, the macroinstruction should be issued asynchronously to avoid delays.
- If an RPL-based macroinstruction (such as CLSDST, SEND or EXECRPL) is issued in a LERAD or SYNAD exit routine, a flag should be set to ensure that, in the event of an error or special condition, the LERAD or SYNAD exit routine recognizes that it has been re-entered, thereby avoiding a loop. This flag can be set in the leftmost bit of any register, between register 2 and register 12 inclusively, that is used to point to the RPL when the request is issued. The flag is returned to the SYNAD or LERAD exit routine in register 1 along with the RPL address. For example:

```
O    R2,=80000000 SET RECURSION FLAG
SEND RPL=(R2)
```

- In the normal operating system environment, if the exit routine issues a macroinstruction, and is waiting for completion in the same routine (for example, by using CHECK or WAIT), the mainline program, as well as the exit routine, waits until the requested operation is completed. To avoid such delays, consider using an RPL exit routine for notification of completion.
- Be aware of the addressing mode in which the exit routine is given control. The exit routine must use addresses (either 24- or 31-bit) that are consistent with this addressing mode.
- Chapter 10, "Operating system facilities," on page 295, discusses a number of additional considerations for application programs that make use of certain operating system facilities.

Exit procedures

In the normal operating system environment, an exit routine can branch to any location in the program. When the exit routine is finished, the following conventions must be observed:

- Except for LERAD and SYNAD, exit routines must return control with a BR 14 after register 14 has been restored with the address it contained when the exit routine is entered (an address within VTAM).
- For LERAD and SYNAD exit routines, if the program returns control with a BR 14, it must not issue any macroinstruction that would change the contents of the 18-word save area whose address is in register 13 on entry. Then, when ready to return control, it puts the address of the old save area back in register 13. In other words, when the program returns control with a BR 14, register 13 must be pointing to the same save area it is pointing to at the time the LERAD or SYNAD exit routine is entered.
- A LERAD or SYNAD exit routine can use registers 0 and 15 to pass information to the mainline program.
- Chapter 10, "Operating system facilities," on page 295, discusses a number of additional considerations for application programs that make use of certain special operating system facilities.

DFASY exit routine

The DFASY exit routine provides a way for VTAM to notify an application program that an expedited-flow data-flow-control request has arrived. The requests that can be received by an application program in a DFASY exit are:

- Quiesce at End of Chain (QEC)
- Release Quiesce (RELQ)
- Request Shutdown (RSHUTD)
- Shutdown Complete (SHUTC)
- Shutdown (SHUTD)
- Signal (SIG)
- Stop Bracket Initiation (SBI)

For information on all of these requests, see Appendix C, "Summary of control requests and indicators," on page 681. See "DFSYN, DFASY, and RESP types of RUs" on page 160 for information on the DFSYN, DFASY, and RESP types of RUs.

If a DFASY exit routine is specified in an NIB EXLST or in an ACB EXLST applicable to the session whenever an expedited-flow data-flow-control request arrives, VTAM can schedule that DFASY exit routine. The detailed manner in which VTAM handles DFASY input is shown in Figure 39 on page 179 and Figure 40 on page 180. If an NIB exit routine is specified, it is always scheduled, whereas a specified ACB exit routine is scheduled only if the following conditions exist:

1. No NIB exit routine is specified.
2. The NIB for the session specified DFASYX.
3. The session is in CA mode for DFASY input.
4. RECEIVE OPTCD=SPEC,RTYPE=DFASY is not currently queued.

Using a DFASY exit routine is an alternative to getting each expedited-flow data-flow-control request with RECEIVE RTYPE=DFASY. The advantages and disadvantages of the two alternatives are discussed in "Explicit RECEIVEs and EXLST exit routines" on page 178.

For a DFASY exit routine, information about the request that has been received is available in a read-only RPL provided by VTAM. This RPL resides in read-only VTAM storage and that cannot be used by RPL-based macroinstructions. The following are also true for this RPL:

- The application program uses SHOWCB, TESTCB, or assembler instructions with the IFGRPL DSECT to examine the RPL fields
- 3-byte user RH field is set in the read-only RPL (see "Operation for inbound RUs" on page 198 for more information)
- All feedback fields (except REQ) are set exactly as they would be following RECEIVE RTYPE=DFASY (see Figure 92 on page 522)
- CHECK must not be issued for the RPL
- RPL is freed when control returns to VTAM.

The location of the read-only RPL is provided in the parameter list passed to the exit routine when the routine is scheduled.

Table 32. DFASY exit routine: Registers upon entry

Reg	Contents
0, 2-13	Unpredictable
1	Address of a parameter list that includes the following: Word 1—address of ACB for application program to which the expedited-flow data-flow-control request is sent Word 2—CID of the session Word 3—USERFLD data from NIB that is used to establish the session Words 4, 6—reserved Word 5—address of a VTAM-supplied, read-only RPL
14	VTAM address that is branched to when DFASY exit routine completes processing
15	Address of DFASY exit routine

LERAD exit routine

A LERAD exit routine is identified in an ACB exit list when the application program wants a routine to be automatically invoked when a logic error (in contrast to a physical error) is detected.

Generally, a logic error results when an RPL-based request is made that is inherently contradictory, for example, when attempting to use an CID that is not valid. The SYNAD exit routine handles physical errors (such as hardware malfunctions). The LERAD exit routine, if specified, is entered for recovery action return codes of 20 and 24 (decimal). Chapter 9, "Handling errors and special conditions," on page 277, discusses the use and logic of LERAD exit routines in detail.

When the LERAD exit routine returns control to VTAM, VTAM leaves registers 0 and 15 intact so that the routine can pass information in these registers back to the part of the application program from which LERAD is invoked.

VTAM returns control to the next sequential instruction in the application program following the RPL-based request. Because the routine is executed under the same system scheduling control block as the part of the program that issues the RPL-based or CHECK macroinstruction, LERAD can branch to other parts of the

program. If LERAD is entered from an RPL-based request or if the CHECK is issued in an asynchronous exit routine, the program must eventually branch to register 14. If the routine returns control to the next sequential instruction by branching to the register 14 address, VTAM restores the register from the save area whose address is in register 13.

If the exit routine is running under an SRB, different conditions apply for return. See "Execution of exit routines" on page 307.

Table 33. LERAD exit routine: Registers upon entry

Reg	Contents
0	Recovery action return code (refer to Chapter 9, "Handling errors and special conditions," on page 277).
1	Address of RPL associated with the request. The values in register 0 and register 1 are related in the following ways: • If register 0 is set to 24 (decimal), VTAM cannot place a value in the FDB2 field to specify the reason for the error. This occurs for one of the following reasons: – Issued macroinstruction's RPL is in use. – CHECK is issued for a request whose RPL exit routine has not yet been scheduled. – RPL that is not valid is specified. If the CHECK or RPL-based macroinstruction that caused entry into LERAD exit routine used one of the registers 2–12 inclusively, bit 0 of that register is returned in bit 0 of register 1. You can use this to detect recursive entries of LERAD.
2-12	Unmodified
13	Address of an 18-word save area supplied by the programmer when the macroinstruction that causes the logic error is issued. Be aware of the following: • If the exit routine returns control through register 14, it must not change anything in the save area. If any macroinstruction is issued in the exit routine, register 13 must first be loaded with the address of a new save area. • Before control is returned through register 14, register 13 must be restored with the value it has when the exit routine is invoked.
14	VTAM address that is branched to when LERAD exit routine completes processing
15	Address of LERAD exit routine

LOGON exit routine

An application program can receive a request (through a CINIT request from the SSCP) to establish a session with another LU and act as the PLU of that session. The application program can handle each CINIT either by having VTAM complete a previously issued OPNDST or schedule a LOGON exit routine. The LOGON exit routine technique enables the program to examine the CINIT or make other inquiries of VTAM using INQUIRE before establishing the session with OPNDST OPTCD=ACCEPT. OPNDST OPTCD=ACCEPT to accept the session or CLSDST to reject the session can be issued either in the LOGON exit routine or in another part of the program, after the LOGON exit routine returns to VTAM. If the session is rejected, sense data can be included on the negative response to the CINIT with CLSDST OPTCD=SENSE.

Note: When a CINIT is received, VTAM first checks for an outstanding OPNDST request that has not yet been completed. If there is no appropriate outstanding OPNDST request, VTAM then schedules a LOGON exit routine if one exists and the scheduling is allowed (see the following discussion of MACRF and SETLOGON). Therefore, a CINIT does not cause a LOGON exit routine to be scheduled if there is a pending OPNDST for the session. If no LOGON exit routine exists, and there is no outstanding OPNDST for the session, the CINIT is queued.

Regardless of the mechanism by which the LOGON exit routine is scheduled, the routine is asked to establish a session in which the application program acts as the PLU. The routine's principal task, therefore, is to determine whether it should honor the request and, when it determines that it should, issue OPNDST OPTCD=ACCEPT to establish the session. If the request is not honored, the routine issues CLSDST to reject the CINIT. If neither OPNDST nor CLSDST is issued, the CINIT remains queued, which can prevent the LU from establishing other sessions. If MACRF=LOGON is specified in the ACB and SETLOGON OPTCD=QUIESCE has not been issued, CINITs are queued for the application program regardless of whether a LOGON exit routine is available. A CINIT remains queued until the program issues OPNDST or CLSDST for the session unless a termination RU occurs. For example, a VARY TERM is issued or DACTLU/INOP flows. Queuing a CINIT does not necessarily mean that the CINIT is queued for eventual scheduling of the LOGON exit routine; it merely means that the CINIT is queued for an eventual OPNDST OPTCD=ACCEPT (or CLSDST). The LOGON exit routine is scheduled only if MACRF=LOGON is specified for the ACB and if the application program issued SETLOGON OPTCD=START. For further information about CINIT handling, see Table 6 on page 82.

You can use SETLOGON OPTCD=HOLD and OPTCD=START to synchronize session setup requests. SETLOGON OPTCD=HOLD causes all subsequent CINIT requests to be queued and prevents the scheduling of the LOGON exit for session setup requests. When SETLOGON OPTCD=START is issued after SETLOGON OPTCD=HOLD, VTAM schedules the LOGON exit request. VTAM continues to drive the LOGON exit as usual until the application issues SETLOGON OPTCD=HOLD or SETLOGON OPTCD=QUIESCE. If a LOGON exit routine exists, SETLOGON OPTCD=START must be issued to cause session reallocation. If a LOGON exit routine exists and SETLOGON OPTCD=START is not issued, sessions are not reallocated as part of application activation.

A LOGON exit routine may be entered multiple times if the application supports this function. (Only an LU 6.2 application can support this function.) The application-capabilities vector specifies if the application program's LOGON exit routine can be entered multiple times during a session initiating processing for the same two session partners.

For a LOGON exit routine, information about the request that has been received is available in a read-only RPL provided by VTAM. This RPL resides in read-only VTAM storage and cannot be used by RPL-based macroinstructions. The following are also true for the RPL:

- Read-only RPL (see Figure 92 on page 522) can be examined by IFGRPL DSECT or by SHOWCB or TESTCB
- RPL and the copy of the CINIT RU are freed when the exit routine returns to VTAM; any information needed from these areas must be copied before then.

Certain information can appear in control vectors appended to the CINIT RU.

Vector Description

X'0D' Class-of-service and virtual route list

X'0E' Network-name control vector or network-qualified name of the PLU

X'0E' Network-name control vector or network-qualified name of the SLU

X'15' Network-qualified address pair control vector

X'2C' COS and TPF control vector

X'2D' Mode-name control vector

X'2F' Model-terminal-information control vector

X'59' Session authorization data control vector

X'5F' Extended fully qualified PCID control vector

X'60' Fully qualified PCID control vector

X'64' TCP information control vector

X'66' Data compression control vector

The read-only RPL's AREA field points to the beginning of a read-only copy of the CINIT RU. The RECLEN and AREALEN fields contain the length of the CINIT RU. The logon mode name and class-of-service name used to initiate the session are present in the CINIT. The following should be ignored:

- The session key field containing network addresses
- Any fields in the BIND image field in CINIT beyond the user data field.

Also, the lengths of the password and requester ID fields in CINIT are 0 because VTAM does not support these fields. The read-only RPL's AAREA field points to the beginning of the control vectors attached to the CINIT. Refer to *SNA Formats* for a list of control vectors on CINIT and their formats.

Table 34. LOGON exit routine: Registers upon entry.

Reg	Contents
0, 2-13	Unpredictable
1	Address of a parameter list that includes the following: • Word 1—address of ACB for application program • Word 2—address of 8-byte symbolic name of SLU [1] – Name is placed in NAME field of NIB used for session – Area is freed when exit routine returns to VTAM. • Word 3—If LOGON exit routine is entered by SIMLOGON, this word contains USERFLD data from NIB used by SIMLOGON, otherwise this word is 0 • Word 4—length of user data from the LU (this data can be accessed by INQUIRE OPTCD=LOGONMSG or examined in CINIT RU) • Word 5—address of a VTAM-supplied, read-only RPL • Word 6—CID of session to be established with LU [2] • Word 7—address of the network identifier parameter list [3]
14	VTAM address that is branched to when LOGON exit routine completes processing
15	Address of LOGON exit routine

Table 34. LOGON exit routine: Registers upon entry (continued).

Reg	Contents

Note:

1. This name is the LUALIAS name or the 8-byte name of the SLU taken from the CINIT.
2. The application program should use the CID to identify the session for a particular logical unit.
3. The network identifier parameter list is mapped by the ISTNRIPL DSECT. For more information, see Table 117 on page 762.

TSO/VTAM Katakana and double-byte character set (DBCS) support

TSO/VTAM's LOGON exit has been modified to support the LANG operand of the MODEENT macroinstruction.

If the BIND indicates that the Query command is not supported, TSO/VTAM determines which characters are valid by examining the language byte.

If the BIND indicates that the Query command is supported, TSO/VTAM determines which characters are valid according to the language specifications in the reply to the Query command.

LOSTERM exit routine

VTAM can schedule a LOSTERM exit routine when a session with an application program is terminated or potentially disrupted, or when a conditional terminate request for a session is received. Alternatively, for some of these conditions, an SCIP exit routine is scheduled with UNBIND, or an NSEXIT exit routine is scheduled with CLEANUP. See “Session outage notification” on page 110 for details. The application program might issue CLSDST to end the session. (If the application program fails to issue CLSDST, the LU with which the application program is, or was, in session might be unavailable for a session with any other application program. This occurs if the LU is at its session limit as a result of this session.)

If a session outage occurs, VTAM posts any outstanding requests associated with the affected session with an appropriate return code. If there are no outstanding requests, whenever the program makes the next request, it is posted with an appropriate return code.

A LOSTERM exit routine is especially recommended for an application program that does not issue specific-mode communication requests for its sessions, but is driven instead by input arriving as the result of RECEIVE macroinstructions issued in any-mode. Use of the exit routine is also recommended for an application program when there is the possibility that the LU can fill VTAM's buffers (obtained from application program storage) faster than the application program is emptying them with RECEIVE macroinstructions.

An example of a coded LOSTERM exit routine is shown in Chapter 15, “Sample code of a simple application program,” on page 579.

Table 35. LOSTERM exit routine: Registers upon entry

Reg	Contents
0, 2-13	Unpredictable
1	Address of a parameter list that includes the following: Word 1—address of ACB of application program whose session has been affected Word 2—address of RPL if reason code 44; for all other reason codes, CID of the session Word 3—reserved if reason code 44; for all other reason codes, USERFLD data from NIB that is used to establish the session Word 4—reason code value indicating why LOSTERM is entered (unless indicated otherwise, the reason code is reported only to the PLU application) Words 5, 6—reserved
14	VTAM address that is branched to when LOSTERM exit routine completes processing
15	Address of LOSTERM exit routine

LOSTERM reason codes

The following section describes the different reason codes that are found in register 1.

Reason code
Meaning

0 (X'00')
Reserved.

4 (X'04')
Reserved.

8 (X'08')
Reserved.

12 (X'0C')
Session has been terminated; immediate recovery is unlikely. The application program **must** issue CLSDST if it has not already done so. The cause of the session termination might preclude the session from being reestablished immediately; for example, the VTAM operator might have deactivated the LU or the LU might have had an unrecoverable failure (for example, an abend).

For a list of possible causes of reason code 12, see "Session outage notification (SON) codes on UNBIND" on page 93.

As discussed in Chapter 5, "Establishing and terminating sessions with logical units," on page 81, some of the types of session outages that cause it are reported instead through an SCIP exit routine (if SONSCIP=YES on the APPL definition statement) or otherwise through an NSEXIT routine (if one exists).

16 (X'10')
The session has been terminated. This reason code is reported immediately after, and only after, reason code 24 (unless CLSDST has been issued). The application program can try to reinitiate the session, for example, by issuing SIMLOGON or OPNDST OPTCD=ACQUIRE; however, the application program **must** issue CLSDST if it has not already done so.

If the LU has a controlling APPL, that is, if the original session is a result of a LOGAPPL parameter on a definition statement, or of a VARY LOGON command, VTAM attempts to reestablish the session by scheduling the LOGON exit.

Note: Once CLSDST has been issued, session initiation is subject to the normal rules. Therefore, if another LU has a queued session with the LU, the new session requested by the application program whose LOSTERM exit routine is invoked cannot be immediately established.

As discussed in Chapter 5, "Establishing and terminating sessions with logical units," on page 81, many of the types of session outages that cause it are reported instead through an SCIP exit routine (if SONSCIP=YES on the APPL definition statement) or otherwise through an NSEXIT exit routine (if one exists).

20 (X'14')
A CTERM Forced request has been received. Perhaps the LU issued a Terminate Forced request, using TERMSESS OPTCD=UNCOND, for example. For a list of possible causes of reason code 20, see "Session outage notification (SON) codes on UNBIND" on page 93. The application program **must** issue CLSDST. When this completes, the application program can attempt to reinitiate the session.

24 (X'18')
The session has been terminated. The LOSTERM exit routine is immediately rescheduled with reason code 16 for this session. (See the preceding description of reason code 16.) The application program can issue CLSDST at this time, which in turn might cancel the execution of the LOSTERM exit routine with reason code 16. The session outage can instead be reported through an SCIP or NSEXIT exit routine as described under reason code 16.

28 (X'1C')
Reserved.

32 (X'20')
A CTERM Conditional request has been received. Perhaps the LU issued a Terminate Orderly request, using TERMSESS OPTCD=COND, for example. For a list of possible causes of reason code 32, see "Session outage notification (SON) codes on UNBIND" on page 93. The application program can issue CLSDST at any time. The interpretation of the receipt of CTERM Conditional is not defined by SNA or VTAM.

36 (X'24')
Received request or response units for this session have been discarded because of a lack of buffer space. The session has not been terminated, but session data recovery procedures are required (for example, CLEAR, SDT, and STSN as described in Chapter 6, "Communicating with logical units," on page 151). This reason code can be reported to either a PLU or SLU application program. For further details, see the description of (RTNCD,FDB2)=(X'10',X'0F') in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

44 (X'2C')
A cross-memory macroinstruction request failure has occurred. VTAM attempted to SUSPEND or RESUME the execution thread of the requestor, but the operation was not successful, possibly because the cross-memory address space has failed. The user RPL (pointed to by Word 2 of the exit

parameter list) contains return code and feedback information describing the failures, but VTAM cannot perform normal completion notification. The RPL is now available for reuse.

48 (X'30')
A CLEANUP RU has been received. No NSEXIT is available to issue the CLSDST macroinstruction. VTAM will close the session.

Note: For any of the LOSTERM reason codes that require or recommend a CLSDST macroinstruction, do not issue a second CLSDST if one has been issued for the same session, but possibly for a different reason.

NSEXIT exit routine

The NSEXIT exit routine is entered whenever certain network services RUs arrive for an application program. Network services RUs are sent to the application program on the SSCP-to-application program LU session. (If the program is a communication network management program, any network services RUs having a management-services category are sent to the application program as data. For more information, refer to Chapter 12, "Coding for the communication network management interface," on page 337. In this case, the application program must issue a RECEIVE macroinstruction; the NSEXIT exit routine is not scheduled for management services requests.) The action taken by the exit routine depends on the type of network services request unit received by the program.

An application program can receive any of three types of network services request units:

- The program receives a ***Clean Up Session*** (CLEANUP) ***request unit*** (RU) when a session of which the application program is a part is terminated.
- The program receives a ***Notify request unit*** (containing control vector hex 3) if, after the program has issued a session-initiation request (other than through OPNDST OPTCD=ACQUIRE) with a ***nonzero NIB USERFLD*** and the request has been responded to positively by the SSCP, something happens that makes it impossible to ensure that the session is set up successfully. The program also receives a Notify RU (containing control vector hex 3) after the session initiated by CLSDST OPTCD=PASS is established if PARMS=(THRDPTY=NOTIFY) is specified.
- The program receives a ***Network Services Procedure Error*** (NSPE) ***request unit*** if, after the program has issued a session-initiation request with a ***zero NIB USERFLD*** and the request has been responded to positively by the SSCP, something happens that makes it impossible to ensure that the session is set up successfully. (See "Network services procedure error or Notify" on page 245 for more information.)

When the exit routine is entered, VTAM provides it with the address of a read-only RPL in the fifth word of the parameter list. The fields of this RPL that are filled in by VTAM are shown in Figure 181 on page 851. The AREA field of the RPL contains the address of the RU that is received, and the RECLEN field (as well as the AREALEN field) of the RPL specifies the number of bytes in the RU. The exit routine examines the RU to determine which type of network services RU is received; this determines what action it should take. The following are also true for this RPL:

- The application program uses SHOWCB, TESTCB, or IFGRPL DSECT to examine the RPL fields (see Figure 92 on page 522).

- The 3-byte user RH field is set (see "Operation for inbound RUs" on page 198 for more information).
- CHECK must not be issued for the RPL.
- RPL and RU pointed to by AREA are freed when control returns to VTAM.

In the event that other types of network services request units are passed to the NSEXIT exit routine in the future, the exit routine must test the 3-byte network services header to determine the particular type of request received and take action for each type. The exit routine must also take particular action when it receives a request unit other than one of the types it expects to receive. If the exit routine receives an RU other than an NSPE, Notify, or CLEANUP RU, the exit routine should set register 0 to 0 and register 15 to 4 and then return control to VTAM. Otherwise, it must set register 0 to 0 and register 15 to 0 before returning. This allows VTAM to determine whether the application program understands the network services request and to take appropriate action for that RU if the application program does not understand it. (VTAM does not take any special action if values other than 0 are returned in registers 0 and 15.) Similarly, certain other fields within the network services request units must be checked; if the values of these fields are not understood, register 0 must be set to 0 and register 15 must be set to 4. See Figure 64 on page 247, Table 36 on page 247 to Table 40 on page 251, and Figure 65 on page 252 for the fields that must be checked.

Network services procedure error or Notify

As indicated in the preceding section, an NSPE or Notify RU can arrive at an application program when, after having received a positive response to a session-initiation request, the program is awaiting the next event in the session-establishment procedure. Here are some examples of conditions that cause an NSPE or Notify RU to be generated and delivered to an application program:

- An application program issues REQSESS, and the macroinstruction is completed successfully (indicating that SSCP returned a positive response to the request). The PLU then rejects the resulting CINIT (for example, by issuing CLSDST) which causes an NSPE or Notify RU to be sent to the application program that issued REQSESS.
- An application program issues REQSESS, and the macroinstruction is completed successfully. The PLU is then abnormally terminated before it can process the CINIT that resulted from the REQSESS. VTAM sends an NSPE or Notify request unit to the application program that issued REQSESS.
- An application program issues SIMLOGON for an LU, and the macroinstruction is completed successfully (indicating that a CINIT for a session with the LU is created by VTAM and sent to the application program that issued the macroinstruction). Before the CINIT can be processed, the VTAM operator deactivates the LU. This causes VTAM to send the application program an NSPE or Notify request unit.
- An application program issues SIMLOGON, and the LOGON exit routine is entered. If CLSDST is issued, an NSPE or Notify results.
- Application program A issues CLSDST OPTCD=PASS to pass an LU to application program B, and the macroinstruction completes successfully. When application program B processes the resulting CINIT, it either (1) rejects the CINIT by issuing CLSDST or (2) issues OPNDST to the LU, but the LU rejects the BIND request by sending a request-rejected response. In either case, VTAM sends an NSPE or Notify request unit to application program A. The request unit signals application program A that, even though the CLSDST OPTCD=PASS is posted complete, the session that was requested cannot be established.

- Application program A issues CLSDST OPTCD=PASS to pass an LU to application program B, and specifies PARMS=(THRDPTY=NOTIFY). Application program B issues OPNDST OPTCD=ACCEPT to accept the resulting CINIT and send BIND to the LU; OPNDST is successful. VTAM sends a Notify request unit to application program A to indicate that the session is established successfully.
- An application program issues OPNDST OPTCD=ACQUIRE. Before the session is completely established, the VTAM operator issues VARY INACT,FORCE for the SLU. NSPE is sent to the application program.

The format of the NSPE request unit is shown in Figure 64 on page 247. The format of the Notify request is shown in tables from Table 36 on page 247 to Table 40 on page 251.

If parallel sessions are initiated, the network-name pair in NSPE and Notify is insufficient to determine which Initiate failed. Notify, however, contains a user request correlation field that is transformed by VTAM into a user correlator, and passed in word 3 of the exit routine parameter list. This can be used to isolate the failing Initiate. NSPE does not contain a user request correlation field. Thus, if it is critical to determine the failing request, SIMLOGON, rather than OPNDST OPTCD=ACQUIRE, should be used by a PLU because Notify cannot be requested for OPNDST, but can be requested for SIMLOGON. For further details, see "SIMLOGON macroinstruction" on page 87.

In general, if an NSPE or Notify is received indicating that VTAM cannot ensure the establishment of a requested session, either the session has not been established or it is in the process of being terminated. Therefore, if the application program issues TERMSESS or CLSDST referencing that session, a return code indicating that the session does not exist is usually (but not always) returned.

In some situations, such as the first two conditions described in the preceding discussion, the application program can issue another session-initiation request immediately, or can wait and issue the request at a later time. Even if no other action is taken, the NSEXIT exit routine should set registers 0 and 15 to 0 before returning control to VTAM.

Byte	Contents	Description
0-2	X'010604'	Network service header. (Must be checked.)

Note: The remainder of this RU is in one of two formats, comprehensive or condensed, depending on the setting of bit 7 of the reason byte (byte 3). The implementation determines when each format is used.

Comprehensive Format

3 — Reason code; (See also sense data below.)

Note: In the comprehensive format, the reason byte is coded for two different uses.

If bit 4 is 0, then the reason byte is coded for a setup procedure error.
If bit 4 is 1, then the reason byte is coded for a takedown procedure error.

Setup Procedure Error

0123 4567
1... .... An error occurred sending CINT to the PLU.
.1.. .... An error occurred sending BIND to the SLU.
..1 .... Session establishment was rejected at the PLU.
...1 .... Session establishment was rejected at the SLU.
.... 0... A session setup procedure error occurred.
.... .0.. Reserved
.... ..1. Session establishment was rejected at the SSCP.
....1 This RU is in the comprehensive format.

Takedown Procedure Error

0123 4567
1... .... An error occurred sending CTERM to the PLU.
.1.. .... An error occurred sending UNBIND to the SLU.
..1 .... Session takedown was rejected at the PLU.
...1 .... Session takedown was rejected at the SLU.
.... 0... A session takedown procedure error occurred.
.... .0.. Session takedown was rejected at the SSCP.
.... ..1. The bit combination of 11 for bits 4 and 6 is set aside for implementation internal use and is not otherwise defined.
....1 This RU is in the comprehensive format.

4-7 — Sense Data

The sense data, if applicable, is from the step in the program that caused the session setup or takedown failure. For the meaning of the sense data, see *SNA Formats.*

8 X'06' — Session key for network name pair (must be checked if the following field is examined)

9-n+ — Identification of the LUs involved in failed procedure, as follows:

+This value represents a variable-length field.

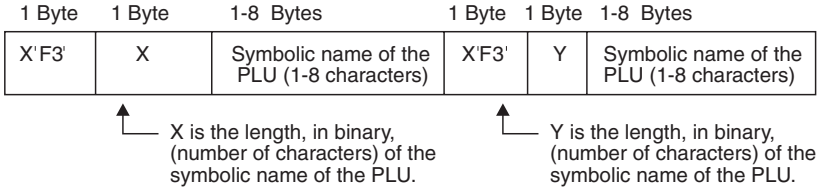


Figure 64. Format of a Network Services Procedure Error request unit

Table 36. Format of a Notify request unit (Part 1 of 5)

Byte	Contents	Description
0-2	X'810620'	Network service header (must be checked) for SSCP-LU and for LU-SSCP.
3-n*		One NOTIFY vector described in the following
3 4-n*		Control vector key (must be checked) vector data

Note: The body of a NOTIFY Request Unit consists of one of the following NOTIFY vectors.

ILU/TLU or third-party SSCP notification NOTIFY vector (X'03').

- ILU/TLU notification informs the sender of an INIT or TERM request of the status of the session.
- Third-party notification informs a third-party SSCP (the SSCP whose LU issued an INIT-OTHER) of the status of the setup procedure.

3 X'03' Control vector key (must be checked)

Table 36. Format of a Notify request unit (Part 1 of 5) (continued)

Byte	Contents	Description
4		Status (must be checked) The following lists the currently supported values.
	X'00'	SSCP (OLU) and SSCP (DLU) not logically connected. No session (or session setup path if rerouting is required) exists between them.
	X'01'	Session terminated
	X'02'	Session set up (In the same-domain case, +RSP (SESSST) has been sent, or, in the cross-domain case, +RSP (CDSESSST) has been sent or received).
	X'03'	Procedure error
5-12		PCID, a unique value used as a session identifier
13		Reason (defined for a status field value of X'03' only)

Note: The reason byte is coded for two different uses:

If bit 4 is 0, the reason byte is coded for a setup procedure error.

If bit 4 is 1, the reason byte is coded for a takedown procedure error.

Setup procedure error

```
0123  4567
```

1... ....
An error occurred sending CINIT to the PLU.

.1.. ....
An error occurred sending BIND to the SLU.

..1. ....
Session establishment was rejected at the PLU.

...1 ....
Session establishment was rejected at the SLU.

.... 0...
A session setup procedure error occurred.

.... .X..
Reserved.

.... ..1.
Session establishment was rejected at the SSCP.

....X
Reserved.

Table 36. Format of a Notify request unit (Part 1 of 5) (continued)

Byte	Contents	Description
		Takedown procedure error **0123 4567** **1... ....** An error occurred sending CTERM to the PLU. **.1.. ....** An error occurred sending UNBIND to the SLU. **..1. ....** Session takedown was rejected at the PLU. **...1 ....** Session takedown was rejected at the SLU. **.... 1...** A session takedown procedure error occurred. **.... .1..** Session takedown was rejected at the SSCP. **.... ..0.** The bit combination of 11 for bits 4 and 6 is set aside for implementation internal use and is not otherwise defined. **.... ...X** Reserved.
14-17		Sense data (defined for a status value of X'03' only) The sense data, if applicable, is from the step in the procedure that caused the session setup or takedown failure. For the meaning of the sense data, refer to *SNA Formats*

*This value represents a variable-length field.

Table 37. Format of a Notify request unit (Part 2 of 5)

Byte	Contents	Description
18-m		Session key for network name pair (Must be checked if the following field is examined.)
One of the following keys is used:		
	X'06'	Network name pair (The first name of the pair is a PLU, OLU, LU1; the second name of the pair is an SLU, DLU, or LU2.)
	X'07'	Network address pair (PLU and SLU, respectively)
	X'0C'	User request correlation (URC) field **Note:** This session key is applicable within a NOTIFY only for SSCP-LU; it is the URC carried in the session key field (rather than the value from the URC field) in TERM, and differs from the URC in bytes m+1 through n below.
	X'15'	Network-qualified address pair (PLU and SLU, respectively)
	X'1C'	Network-qualified name pair (The first name of the pair is a PLU, OLU, or LU1, the second name of the pair is an SLU, DLU, or LU2.)
m+1-n*		User request correlation (URC) field
m+1		Length, in binary, (number of characters) of the URC name
m+2-n*		URC name The LU-defined identifier specified in an INIT or TERM request (It is used to correlate the NOTIFY to the initiating request.) **Note:** The URC length is 0 for SSCP-SSCP.

*This value represents a variable-length field.

Resource available NOTIFY vector (X'06') (replaces NOTIFY vector key X'01'). It is used to inform the current users (LUs) or actively controlling SSCPs of a resource (LU) that another LU wishes to use the resource. It is sent by an SSCP that supports NOTIFY NS key X'06', as specified in the CDRM (X'06') control vector, to an SSCP with the same capabilities or to an LU in its domain.

Table 38. Format of a Notify request unit (Part 3 of 5)

Byte	Contents	Description
3	X'06'	Control vector key (must be checked)
4-n*		Three X'19' control vectors are used to identify the LUs involved in the resource request: X'19' resource identifier control vector identifying the current session partner of the requested LU and the target LU X'19' resource identifier control vector identifying the requested LU X'19' resource identifier control vector identifying the requesting LU. **Note:** If the length of one of the resource identifier control vectors is 0, the indicated LU name is unavailable.

*This value represents a variable-length field.

LU-LU session services capabilities NOTIFY vector (X'0C'). It is used to inform the SSCP having an active session with the sending LU of the current LU-LU session services capabilities of that LU.

Table 39. Format of a Notify request unit (Part 4 of 5)

Byte	Contents	Description
3	X'0C'	Control vector key (must be checked)
4		Vector length
5-n*		Vector data
5		Bits 0-3, PLU capability (reserved for Type 2.1 nodes): **0123 4567** **0000 ....** PLU capability is inhibited. Sessions can be neither queued nor started. **0001 ....** PLU capability is disabled. Sessions can be queued or started. **0010 ....** Reserved. **0011 ....** PLU capability is enabled. Sessions can be queued or started. Bits 4-7, SLU capability: **0123 4567** **.... 0000** SLU capability is inhibited. Sessions can be neither queued nor started. **.... 0001** SLU capability is disabled. Sessions can be queued or started. **.... 0010** Reserved **.... 0011** SLU capability is enabled. Sessions can be queued or started.
6-7		LU-LU session limit (A value of 0 means that no session limit is specified.)

Table 39. Format of a Notify request unit (Part 4 of 5) (continued)

Byte	Contents	Description
8-9		LU-LU session count: the number of LU-LU sessions for this LU that are not reset, and for which SESSEND is sent to the SSCP
10		Flags **0123 4567** **1... ....** Parallel sessions capability. Parallel sessions supported **.1.. ....** Retired **..1. ....** SESSST capability in RSP(ACTLU) (reserved in NOTIFY). SESSST RU is sent if for an SLU **...1 ....** XRF session activation (X'27') control vector support in RSP(ACTLU) (reserved in NOTIFY and for peripheral nodes). XRF session activation (X'27') control vector supported on BIND. **.... 1...** Peripheral node extended BIND support indicator (used between a dependent LU and its BF, but otherwise reserved). Dependent LU does support receipt of extended BINDs. **.... .1..** Network-qualified names support indicator in bytes k+2-m and p+2-r. A BIND received by this LU can contain network-qualified LU names in bytes k+2-m and p+2-r. **.... ..1.** Subarea node extended BIND support indicator (used between a subarea LU or BF and its SSCP, but otherwise reserved). Subarea LU or BF(LU) does support sending and receiving extended BIND. **.... ...1** Boundary function (BF) network address pair (X'15') session key support (reserved for peripheral nodes). BF does support session key X'25'. **Note:** Boundary function support for session key X'25' cannot be changed after RSP(ACTLU); in NOTIFY, the sender sets bit 7 to 0, which is then ignored by the receiver.
11		Vector extension
12-18		Retired

LU-LU session services capabilities NOTIFY vector (X'0C'). It is used to inform the SSCP having an active session with the sending LU of the current LU-LU session services capabilities of that LU.

Table 40. Format of a Notify request unit (Part 5 of 5)

Byte	Contents	Description
19		Additional capabilities: **0123 4567** **1... ....** Receipt of unrecognized control vectors on CINIT supported. **.111 ....** Reserved **.x11 ....** 1 **..xx xxxx** Reserved

Cleanup session

For certain types of session outages, see “Session outage notification (SON) codes on UNBIND” on page 93. VTAM sends the application program a CLEANUP request unit.

Arrival of the CLEANUP request unit at an application program causes the program's NSEXIT exit routine to be scheduled (if one exists). Because VTAM has completely terminated the session, the exit routine does not take any action to end the session (that is, it does not issue a CLSDST or TERMSESS macroinstruction). The exit routine can clean up application program control blocks for the session. The exit routine can also attempt to reestablish the session, and the attempt can be successful if the cause of the session outage has been repaired or bypassed, and the desired LU is available. The format of the CLEANUP request unit is shown in Figure 65.

Byte	*Contents*	*Description*
0-2	X'810629'	Network service header. (Must be checked.)
3		Ru format 0123 4567 0000 .... Format 0 (No other format is defined.) .... xxxx Reserved
4		Reserved
5		Reason code indicating how the session is taken down 0123 4567 x... .... Retired .x.. xxxx 0 - takedown is normal 1 - takedown is normal ..xx xxxx Reserved
6		Session key for network name pair (Must be checked if the following field is examined.) One of the following session keys is used
	X'06'	Uninterpreted name pair
	X'07'	Network address pair (retired)
	X'13'	Network-qualified address pair.
7-n+		Identification of the LUs involved in the cleanup procedure, as follows:

+This value represents a variable-length field.

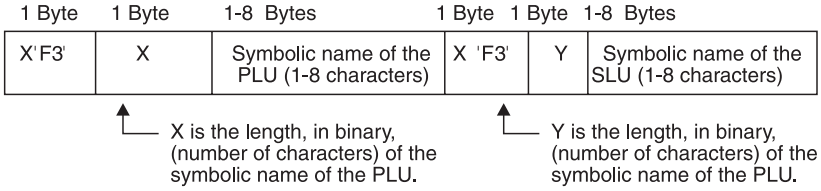


Figure 65. Format of a Cleanup Session request unit

If a PLU application program does not have an SCIP exit routine or has SONSCIP=NO coded on its APPL definition statement, any UNBIND it receives is converted to a CLEANUP for processing by the NSEXIT exit routine.

When the application program involved in a disrupted session is acting as the PLU and does not have an NSEXIT exit routine, that program's LOSTERM exit routine (if one exists) can be scheduled to report loss of the session. See “Session outage notification” on page 110. An application program acting as an SLU must have an NSEXIT exit routine or else it receives no notification of sessions being cleaned up.

Even if no other action is taken, the NSEXIT exit routine should set registers 0 and 15 to 0 before returning to VTAM.

Table 41. NSEXIT exit routine: Registers upon entry

Reg	Contents
0, 2-13	Unpredictable
1	Address of a parameter list that includes the following: • Word 1—address of ACB for application program to which the network services RU is sent • Word 2—contents depend on the type of RU: – CLEANUP RU - CID of the session – NSPE or Notify (control vector hex 3) RU - reserved. • Word 3—contents depend on the type of RU: – CLEANUP RU - USERFLD data of NIB when the session referred to by the CID is established (using OPNDST or OPNSEC) – NSPE - reserved – Notify RU - USERFLD field contents of NIB associated with the issued SIMLOGON, REQSESS, or CLSDST OPTCD=PASS. If another exit has already been scheduled reporting this failure, this word can be zero. • Words 4, 6—reserved • Word 5—address of a VTAM-supplied, read-only RPL that resides in read-only VTAM storage and that cannot be used by RPL-based macroinstructions. • Word 7—contents depend on the type of RU: – CLEANUP or NSPE RU - reserved – NOTIFY RU - address of network identifier parameter list [1]
14	VTAM address that is branched to when NSEXIT exit routine completes processing
15	Address of NSEXIT exit routine

Note:

1. The network identifier parameter list is mapped by the ISTNRIPL DSECT. For more information, see Table 117 on page 762.

RELREQ exit routine

VTAM schedules the RELREQ exit routine when one application program requests a session with an LU that is in session with another application program, and the LU is at its session limit. The requesting program requests a session with SIMLOGON OPTCD=(RELRQ,Q). As a result, VTAM schedules the RELREQ exit routine of the PLU application program currently in session with the LU. The PLU can either ignore the request (that is, remain in session with the LU and make the requesting program wait) or can take action to end the session.

An application program LU can have a session limit of one or have no session limit. If there is no session limit for an application acting as SLU in an active session, subsequent session initiation requests result in additional active sessions with the SLU. Therefore, it is not possible to use the RELREQ exit in this case. If however, the application acting as SLU in an active session has a session limit of one, subsequent session initiation requests can be queued, and the RELREQ exit can be used. An application has a session limit of one if SESSLIM=YES is coded on the APPL definition statement.

If the application program decides to end the session, it might want to determine whether there are any pending (incomplete) communication requests for the session, and wait for those communication operations to complete. To end the

session, the application program issues CLSDST OPTCD=RELEASE. After execution of CLSDST, the LU is made available to the PLU that has the oldest queued session for the LU. If there are still one or more queued sessions for the LU when the PLU has established its session, the RELREQ exit routine of the new PLU is driven (one time only).

If the exit routine decides to ignore the RELREQ request, it takes no action and continues communication on the session. The queued session (from the SIMLOGON OPTCD=Q from the other application program) remains queued until the LU is available or until the queued session is purged (for example, by a VARY TERM for the SLU).

If an application program does not have a RELREQ exit routine, the program cannot be notified of another program's request. The RELREQ exit routine is scheduled at least once for each SLU for which a queued session (specifying RELRQ notification) exists; multiple notifications are possible, but not guaranteed. This means that the PLU to which the LU is made available (when the LU is released) can be different from the PLU whose SIMLOGON caused the current entry to the RELREQ exit routine of the controlling application program.

If the application program shares printers with other IBM subsystems (for example, CICS and IMS) or other application programs, the RELREQ exit routine can be used to end the printer session without operator intervention. See "Program structure recommendations" on page 33 for more information about printer sharing.

Table 42. RELREQ exit routine: Registers upon entry

Reg	Contents
0, 2-13	Unpredictable
1	Address of a parameter list that includes the following: Word 1—address of ACB for application program with which the LU is currently in session Word 2—address of an 8-byte symbolic name of the requested LU (the name is padded on the right with blanks, if necessary) Word 3-6—reserved Word 7—address of the network identifier parameter list
14	VTAM address that is branched to when RELREQ exit routine completes processing
15	Address of RELREQ exit routine

RESP exit routine

The RESP exit routine provides a way for VTAM to notify an application program when a response to a normal-flow request (data or data-flow-control) has arrived.

The RESP exit routine is one of three ways an application program can be notified of receipt of a normal-flow response (other than a DFSYN response). The other two ways are:

- Specifying POST=RESP in the macroinstruction used to send the normal-flow request. If this is done, the macroinstruction is not completed until the response is received.
- Maintaining an active RECEIVE RTYPE=RESP that is completed when a normal-flow response is received.

Using an RESP exit routine is an alternative to getting each normal-flow response with a RECEIVE RTYPE=RESP macroinstruction. The advantages and disadvantages of the two alternatives are discussed in "Explicit RECEIVEs and EXLST exit routines" on page 178.

If an RESP exit routine is specified in an NIB or ACB EXLST applicable to the session, whenever a normal-flow response arrives, VTAM can schedule that RESP exit routine. The detailed manner in which VTAM handles RESP input is shown in Figure 39 on page 179 and Figure 41 on page 181. Certain normal-flow responses are considered DFSYN input instead of RESP input and therefore do not schedule an RESP exit. For normal-flow responses other than DFSYN responses, if a NIB RESP exit routine is specified, it is always scheduled; however, a specified ACB exit routine is scheduled only if:

- No NIB exit routine is specified.
- The NIB for the session specified RESPX.
- The session is in CA mode for RESP input.
- No RECEIVE OPTCD=SPEC,RTYPE=RESP macroinstruction is queued.

For an RESP exit routine, information on the normal-flow response that has been received is available in a read-only RPL provided by VTAM. This RPL resides in read-only VTAM storage and cannot be used by RPL-based macroinstructions. The following are also true for this RPL:

- The application program uses SHOWCB, TESTCB, or IFGRPL DSECT to examine the RPL fields.
- The 3-byte user RH field is set (see "Operation for inbound RUs" on page 198 for more information).
- All feedback fields (except REQ) are set exactly as they would be following RECEIVE RTYPE=RESP.
- CHECK must not be issued for the RPL.
- RPL is freed when control returns to VTAM.

The location of the read-only RPL is provided in the parameter list passed to the exit routine when the routine is scheduled.

Note: Be aware that an RESP exit routine, which is scheduled for execution before the application program issues CLSDST, cannot be executed until after the CLSDST completes.

Table 43. RESP exit routine: Registers upon entry

Reg	Contents
0, 2-13	Unpredictable
1	Address of a parameter list that includes the following: Word 1—address of ACB of application program to which the response is sent Word 2—CID of the session Word 3—USERFLD contents from NIB that is used to establish the session Word 4—reserved Word 5—address of a VTAM-supplied, read-only RPL Words 6, 7—reserved
14	VTAM address that is branched to when RESP exit routine completes processing
15	Address of RESP exit routine

RPL exit routine

An RPL exit routine is entered after an RPL-based operation completes if the RPL or the macroinstruction using it specified the exit routine address in the EXIT operand.

Table 44. RPL exit routine: Registers upon entry

Reg	Contents
0, 2-13	Unpredictable
1	Address of RPL whose RPL operation has just completed
14	VTAM address that is branched to when RPL exit routine completes processing
15	Address of RPL exit routine

SCIP exit routine

The SCIP exit routine is entered when an application program receives any of the following session-control requests by an application program:

- CLEAR
- Start Data Traffic (SDT)
- Request Recovery (RQR)
- Set and Test Sequence Numbers (STSN)
- BIND
- UNBIND.

For CLEAR, RQR, and UNBIND, VTAM automatically sends a response before the request is presented to the exit routine. For STSN and BIND, the application program must send its own response. For SDT, either the application program or VTAM sends the response, depending on the SDT operand specified on the OPNSEC's NIB. For all these requests except UNBIND, if the application program has no SCIP exit routine, VTAM automatically sends a negative response; for UNBIND, VTAM sends a positive response.

Four of the requests—CLEAR, SDT, STSN, and BIND—are sent only from the PLU to the SLU. Thus, in an application program, those four requests can be received and processed only in an SCIP exit routine at the SLU. RQR is sent only from the SLU to the PLU. Thus, that request is received and processed only in an SCIP exit routine at the PLU. UNBIND can be sent from either the PLU or the SLU. The SCIP exit routine at the PLU is entered on receipt of UNBIND only if the APPL definition statement for the PLU application program is coded with SONSCIP=YES.

You can use SETLOGON OPTCD=HOLD and OPTCD=START to synchronize session setup requests. SETLOGON OPTCD=HOLD causes all BIND requests to be queued and prevents the scheduling of the SCIP exit for session setup requests. When SETLOGON OPTCD=START is issued after SETLOGON OPTCD=HOLD, VTAM schedules the SCIP exit for each queued BIND request. VTAM continues to drive the SCIP exit as usual until the application issues SETLOGON OPTCD=HOLD or SETLOGON OPTCD=QUIESCE.

Note: The SCIP exit is driven for all requests other than BIND regardless of the use of SETLOGON OPTCD=HOLD.

For an SCIP exit routine, information on the request that has been received is available in a read-only RPL provided by VTAM. The RPL resides in read-only VTAM storage and cannot be used by RPL-based macroinstructions. The following are also true for the RPL:

- The application program uses SHOWCB, TESTCB, or IFGRPL DSECT to examine the RPL fields (see Figure 92 on page 522).
- The 3-byte user RH field is set (see "Operation for inbound RUs" on page 198 for more information).
- The CONTROL field identifies the particular session-control request that caused scheduling of the SCIP exit routine.
- If SCIP is entered as result of BIND or UNBIND request, AREA contains the address of the BIND or UNBIND RU, and RECLEN and AREALEN fields contain the length of the RU (refer to *SNA Formats* for a description of the contents of the BIND or UNBIND RU).
- CHECK must not be issued for the RPL.

The read-only RPL and any RU pointed to by the AREA field are freed when the exit routine returns to VTAM; any information needed from these areas must be copied before then.

The location of the read-only RPL is provided in the parameter list passed to the exit routine when the routine is scheduled.

Clear

The Clear request is sent by the PLU when the flow of data requests, data-flow-control requests, and responses is to be stopped, either because the PLU is terminating the session or because the PLU wants to take some recovery action. Clear can be sent only if the transmission services profile for the session indicates that Clear is supported.

For more information on Clear, see "Controlling flow" on page 164. For the role of Clear in various sequences, see Appendix D, "Request and response exchanges for typical communication operations," on page 693.

Start Data Traffic (SDT)

When required by the transmission services profile in the session parameters, Start Data Traffic (SDT) is sent from the PLU to the SLU at the beginning of the session and within a session after successful sequence-number resynchronization has occurred. In both cases, SDT informs the SLU that the flow of data requests, data-flow-control requests, and responses can be started (or resumed).

For more information on SDT, see "Controlling flow" on page 164. For the role of SDT in various sequences, see Appendix D, "Request and response exchanges for typical communication operations," on page 693.

Request Recovery (RQR)

The SLU sends Request Recovery to the PLU to request sequence number resynchronization or other recovery operations. In most cases, RQR is sent when the SLU discovers (1) a discrepancy in the sequence numbers of incoming requests or (2) a discrepancy between the sequence numbers it assigned to outgoing requests and the sequence numbers of the responses it is receiving to those

requests. It might also send RQR if it loses or is forced to discard some incoming requests before it can process them. (For example, input buffers are too full to hold the incoming requests.)

For a description of the procedure used to resynchronize sequence numbers, see "Controlling flow" on page 164. For examples of the use of Request Recovery, see Figure 119 on page 706, Figure 132 on page 719, and Figure 133 on page 720.

Set and Test Sequence Numbers (STSN)

Set and Test Sequence Numbers is used by the PLU to resynchronize sequence numbers. For more information on STSN, see:

- "Controlling flow" on page 164
- "SESSIONC—Send a session-control request or response" on page 532
- Figure 109 on page 696 and Figure 110 on page 697
- Figure 119 on page 706
- Figure 132 on page 719 and Figure 133 on page 720

BIND

BIND is sent from the PLU to the SLU during session establishment. BIND indicates that the PLU wants to start a session with the SLU, and the request contains the session parameters that the PLU proposes to use for the session. The application program can issue OPNSEC to accept the session or SESSIONC CONTROL=BIND or TERMSESS OPTCD=UNBIND to reject the session. These macroinstructions can be issued either in the SCIP exit routine or in another part of the program after the SCIP exit routine returns to VTAM. For more information on BIND, see Chapter 5, "Establishing and terminating sessions with logical units," on page 81, especially "BIND request" on page 91, and "BIND response" on page 97. See Appendix D, "Request and response exchanges for typical communication operations," on page 693, for examples of BIND in various sequences, and Appendix F, "Specifying a session parameter," on page 793, for a description of the session parameters in BIND.

Certain information can appear in control vectors appended to the BIND RU:

Vector	Description
X'0E'	Network-name control vector. Network-qualified name of the PLU.
X'0E'	Network-name control vector. Network-qualified name of the SLU.
X'27'	XRF session-activation control vector.
X'2B'	Route selection-control vector.
X'2C'	COS and TPF control vector.
X'2D'	Mode-name control vector.
X'5F'	Extended fully qualified PCID control vector.
X'60'	Fully qualified PCID control vector.
X'66'	Data compression control vector.

Refer to *SNA Formats* for a list of control vectors on BIND and their formats.

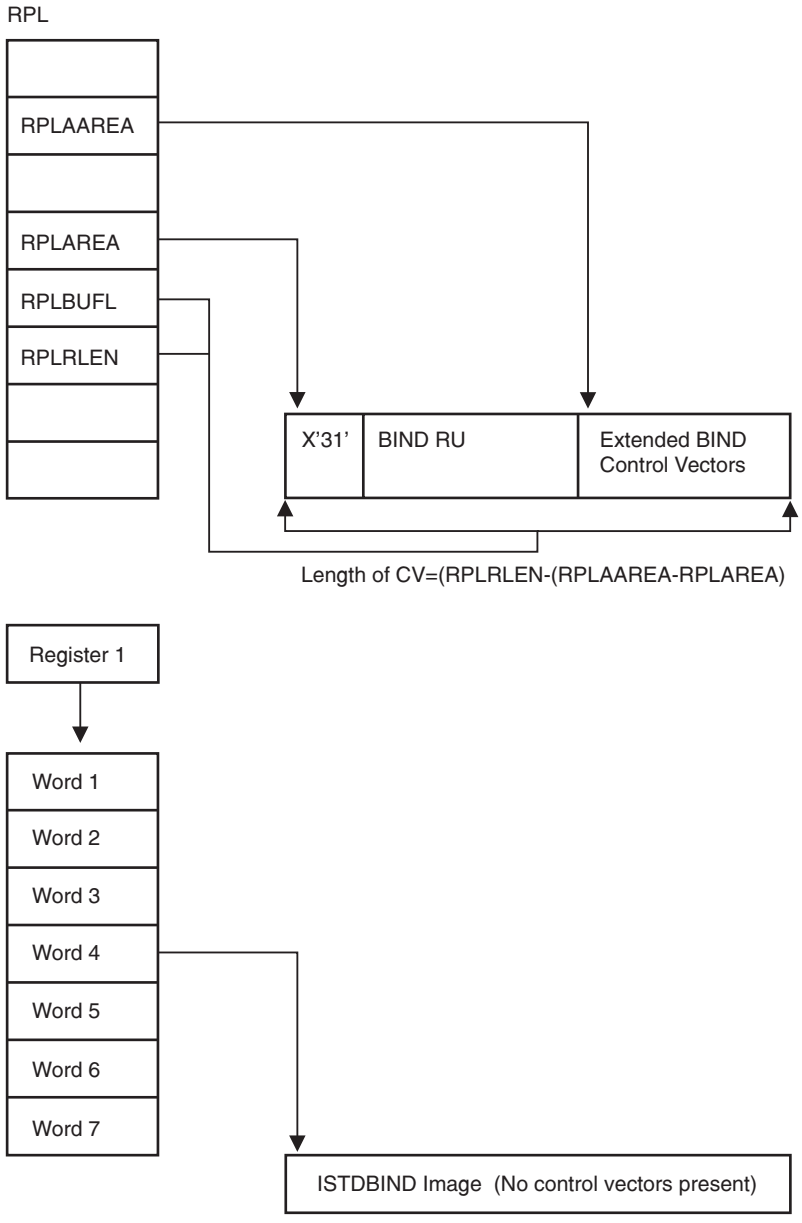
The read-only RPL's AAREA field points to the beginning of the control vectors attached to the BIND. The AAREA field is zero if there are no attached control vectors.

The length of the control vectors is determined as follows:

Length of CV=RPLRLEN-(RPLAAREA-RPLAREA)

The RU pointed to by word 4 of the parameter list, and mapped by ISTDBIND, is not the same RU as the one pointed to by RPLAREA (see Figure 66). Differences exist in the:

- PLU name field
- Crypto fields
- User data.



NOTE: There are two different BIND Images passed to the SCIP Exit.

Figure 66. BIND information presented to SCIP exit

UNBIND

UNBIND is sent by an LU, either a PLU or an SLU, to its session partner in an LU-LU session to end that session. Additionally, UNBIND can be sent on an LU-LU session on behalf of an LU by other network components; this is typically done to notify each LU in the session that a session outage has occurred (for example, because of a network or LU failure).

UNBIND received by an SLU application program is always presented through an SCIP exit routine. UNBIND received by a PLU is presented through an SCIP exit routine only if such an exit routine exists and if SONSCIP=YES is coded on the APPL definition statement for that application program; otherwise, either an NSEXIT exit routine is scheduled with CLEANUP, or a LOSTERM exit routine is scheduled with a reason code. See "Session outage notification" on page 110 for details.

If an SCIP exit routine is scheduled, the AREA field in the read-only RPL provides the address of an UNBIND RU in read-only storage. This RU contains a 1-byte request code and a 1-byte type code. Additional data may be available for certain type codes. Many of the currently defined UNBIND SON (UNBIND type) codes follow. Additional codes are listed in *SNA Formats*. If an application program receives an UNBIND SON code that it does not recognize, it should act as if UNBIND SON type 1 had been received. VTAM terminates the session and attempts to notify the application program for **any** UNBIND SON code.

UNBIND

Type code meaning

1 (X'01')

Normal end of session. (For example, the PLU application program issued CLSDST OPTCD=RELEASE for this session.)

2 (X'02')

BIND forthcoming. (For example, the PLU application program issued CLSDST OPTCD=PASS for this session.) The LU receiving the UNBIND should retain resources associated with this session because a BIND arrives shortly to establish another session which can use those resources.

7 (X'07')

Virtual route inoperative. The virtual route used by this session has become inoperative, possibly because of a link or network node failure. You can try to reinitiate the session (for example, by using SIMLOGON or REQSESS) using the same class of service or a different class of service (class of service is specified indirectly through a logon mode name); this attempts to reestablish the session on a virtual route within that class of service. The virtual route used can be the same as the original virtual route (if it has recovered) or can be another virtual route within the chosen class of service.

8 (X'08')

Route extension inoperative. The route extension used by this session has become inoperative. (The route extension is the connection, including the link, between a peripheral node associated with one end of the session and the subarea node supplying boundary function for that peripheral node.) The session cannot be reestablished until the route extension has been made operative again. You can attempt to reinitiate the session by using SIMLOGON OPTCD=(Q,ASY). This fails if the SSCP has detected a

permanent route extension failure. If a recovery procedure is under way to reestablish contact with the peripheral node, SIMLOGON cannot be posted complete for a long time; it is posted complete with (RTNCD,FDB2)=(X'00',X'00') if the recovery procedure succeeds; it is posted complete with an error return code if a permanent error is detected.

9 (X'09')
Hierarchical reset. The current LU-LU session is being terminated because an SSCP established an SSCP-LU (or SSCP-PU) session with the other LU in this session (or the PU associated with that LU) and that LU (or PU) could not accept the SSCP session without terminating all of its associated LU-LU sessions. An immediate attempt to reinitiate the LU-LU session (for example, by using SIMLOGON) is likely to succeed.

10 (X'0A')
SSCP gone. The current LU-LU session is being terminated because the SSCP-PU or SSCP-LU session associated with the other LU in the session has been intentionally or unintentionally terminated. For example, a VTAM operator has used the VARY command to deactivate that PU or LU, or the virtual route used by the SSCP sessions with the PU and LU has failed and the node providing boundary function support for the PU and LU has been coded to end LU-LU sessions when this occurs. An immediate attempt to reinitiate the LU-LU session is unlikely to succeed because the other LU is currently unavailable.

11 (X'0B')
Virtual route deactivated. The current LU-LU session is being terminated because the virtual route used by that session has been intentionally deactivated. You can try to reinitiate the session (for example, by using SIMLOGON or REQSESS) using the same class of service or a different class of service (class of service is specified indirectly through a logon mode name); this attempts to reestablish the session on a virtual route within that class of service. The virtual route used can be the same as the original virtual route (if it can be reactivated), or a new virtual route can be used.

12 (X'0C')
Unrecoverable LU failure. The current LU-LU session is being terminated because of a permanent failure of one of the LUs involved in the session. An attempt to reinitiate the session probably fails.

14 (X'0E')
Recoverable LU failure. The current LU-LU session is being terminated because of a temporary failure of one of the LUs involved in the session. An attempt to reinitiate the session might be successful.

15 (X'0F')
Cleanup. The current LU-LU session is being terminated because one of the LUs in the session is being cleaned up, usually because an SSCP issued a CLEANUP request to that LU. An attempt to reinitiate the session might succeed.

254 (X'FE')
Session protocol or user-supplied sense code that is not valid. One of the LUs in the session has detected a protocol violation. Four bytes of sense data following the UNBIND SON code indicate the specific violation. Depending on the violation, an attempt to reestablish the session might succeed.

Code X'FE' can also be coded in an application program to indicate that 4 bytes of user-defined sense data are supplied in the UNBIND.

Certain information can appear in control vectors appended to the UNBIND RU:

Vector	Description
X'35'	Extended-sense data control vector.
X'60'	Fully qualified PCID control vector.

For the role of the UNBIND request in various sequences, see Appendix D, "Request and response exchanges for typical communication operations," on page 693.

Table 45. SCIP exit routine: Registers upon entry

Reg	Contents
0, 2-13	Unpredictable
1	Address of a parameter list that includes the following: • Word 1—address of ACB for application program to which the session-control request is sent • Word 2—CID of the session [1] • Word 3—USERFLD data of NIB at time the session is initiated or established; be aware of the following: – If session has been established, word 3 contains USERFLD field contents from NIB used by OPNDST or OPNSEC – For receipt of BIND request resulting from the application program issuing a REQSESS, word 3 contains USERFLD contents from NIB used by REQSESS, else, word 3 is 0 for BIND – For receipt of UNBIND, word 3 can contain the user field data for the session if it still exists, else, word 3 will be 0. This user field data was originally obtained from the USERFLD contents in the NIB. • Word 4—contents depend on the type of request: – BIND request - contains beginning address of the session parameters (see Appendix F, "Specifying a session parameter," on page 793); the area is freed when both of the following have occurred: - Exit routine has returned to VTAM - OPNSEC, SESSIONC, or TERMSESS has been issued for session • Other types of requests—contains no meaningful data • Word 5—address of a VTAM-supplied, read-only RPL • Word 6—address of the PLU's symbolic name [2] (BIND RU only) • Word 7—address of the network identifier parameter list [3] (BIND RU only)
14	VTAM address that is branched to when SCIP exit routine completes processing
15	Address of SCIP exit routine

Notes:

1. The application program should use the CID to identify the session for a particular logical unit.
2. This name is the LUALIAS name, USERVAR name, or the name in the NSPLU name field of the BIND.
3. The network identifier parameter list is mapped by the ISTNRIPL DSECT. For more information, see Table 117 on page 762.

SYNAD exit routine

A SYNAD exit routine is identified in an ACB exit list when a routine is to be automatically invoked when a physical error is detected. A physical error is an unrecoverable input or output error or other unusual condition that occurs during an operation. The SYNAD exit routine, if specified, is entered for all recovery action return codes of 4, 8, 12, and 16 (decimal). Chapter 9, "Handling errors and special conditions," on page 277, discusses the use and logic of SYNAD exit routines in detail.

When the SYNAD exit routine returns control to VTAM, VTAM leaves registers 0 and 15 intact; this enables the routine to pass information in those registers back to the part of the application program from which SYNAD is invoked. VTAM returns control to the next sequential instruction in the application program following the RPL-based request (or CHECK). Because the routine is executed under the same system scheduling control block as the part of the program that issued the RPL-based or CHECK macroinstruction, SYNAD can branch to other parts of the program. If SYNAD is entered from an RPL-based request, or if the CHECK is issued in an asynchronous exit routine, the program must eventually branch to register 14. If the routine returns control to the next sequential instruction by branching to the register 14 address, VTAM restores the register from the save area whose address is in register 13.

If the exit routine is running under an SRB, different conditions apply for return. See "Execution of exit routines" on page 307 for more information.

Table 46. SYNAD exit routine: Registers upon entry

Reg	Contents
0	Recovery action return code (refer to Chapter 9, "Handling errors and special conditions," on page 277).
1	Address of RPL associated with the request If the CHECK or RPL-based macroinstruction that caused entry into SYNAD exit routine used one of the registers 2–12 inclusively, bit 0 of that register is returned in bit 0 of register 1. You can use this to detect recursive entries of SYNAD.
2-12	Unmodified
13	Address of an 18-word save area supplied by the programmer when the macroinstruction that caused the error is issued. Be aware of the following: • If the exit routine is going to return control through register 14, it must not change anything in the save area. If any macroinstruction is issued in the exit routine, register 13 must first be loaded with the address of a new save area. • Before control is returned through register 14, register 13 must be restored with the value it had when the exit routine is invoked.
14	VTAM address that is branched to when SYNAD exit routine completes processing
15	Address of SYNAD exit routine

TPEND exit routine

VTAM invokes the TPEND exit routine when any of the following occurs:

- VTAM operator issues a HALT command.
- VTAM halts itself in an orderly fashion because of an internal problem.
- VTAM terminates abnormally.

- The operator issues a VARY INACT command for the application program.
- VTAM schedules the TPEND exit in response to an OPEN ACB from an alternate application that wants to take over the sessions from the original application.
- VTAM receives notification of a multinode persistent session (MNPS) forced takeover request for this application.

A reason code in the second word of the parameter list passed to the exit routine indicates the reason for entry to the exit routine.

Reason code
Meaning

0 indicates a standard HALT command (a HALT command without the QUICK or CANCEL operand). The program is allowed to continue communications on existing sessions or CMIP connections, but the program should end those communications in an orderly fashion as soon as it can. No new sessions can be established. For CMIP application programs, the MIBConnect function cannot be issued. The exit routine should have the mainline program issue a CLOSE macroinstruction or for CMIP application programs, call the MIBDisconnect function. (A CLOSE macroinstruction or for CMIP application programs, a MIBDisconnect function, cannot be issued in an exit routine.)

4 indicates a a HALT QUICK command, when VTAM is halting itself, or when the operator issues a VARY INACT command for the application program. The pending RPL-based operations are canceled. For reason code 4 (as for reason code 0), the application program should issue the CLOSE macroinstruction or for CMIP application programs, the MIBDisconnect function, in the mainline program.

8 indicates a HALT CANCEL command or VTAM abnormal termination. The pending operations are interrupted (without being marked as completed or canceled), and no VTAM macroinstruction except CLOSE is accepted. For CMIP application programs, only the MIBDisconnect function is accepted. The TPEND exit routine should return to the mainline program for immediate issuance of the CLOSE macroinstruction or for CMIP application progams, the MIBDisconnect function, without any attempt to issue other VTAM requests.

Note: The TPEND exit routine with reason code 8 can interrupt any other exit routine.

12 indicates that VTAM has scheduled the TPEND exit in response to an OPEN ACB from an alternate application that wants to take over the sessions from the original application. VTAM drives the TPEND exit and suspends the sessions. See “Restoring sessions pending recovery” on page 137 for information on how to restore sessions following an application program failure and recovery.

This reason code also indicates that an active application program with the same network name has enabled persistence on another VTAM when this VTAM connects to the multinode persistent sessions coupling facility structure. Specifically, this occurs when a persistence-enabled application program is open and VTAM is not connected to the coupling facility. Later, when VTAM connects to the coupling facility, it detects an open application program with the same name that is already enabled for persistence. See “Opening the ACB during recovery from a node failure” on page 70 for more information.

This reason code might also indicate that VTAM has scheduled the TPEND exit in response to an OPEN ACB from an alternate application on a different node in the sysplex that is requesting MNPS forced takeover processing. VTAM drives the TPEND exit to inform the application that VTAM is generating a CLOSE ACB for the application while the application is in persistence enabled state. This mechanism allows the sessions to be maintained long enough for the MNPS takeover to be performed by the other application instance.

Note: CMIP application programs do not receive this return code.

See "TPEND exit routine is entered" on page 74 for more information on actions taken when the TPEND exit routine is invoked.

Table 47. TPEND exit routine: Registers upon entry

Reg	Contents
0, 2-13	Unpredictable
1	Address of a parameter list that includes the following: Word 1—address of ACB of application program being shut down Word 2—reason code for shutdown Words 3-7—reserved.
14	VTAM address that is branched to when TPEND exit routine completes processing
15	Address of TPEND exit routine

Chapter 8. Setting and testing control blocks and macro global variables

The VTAM application program places values in control blocks (ACB, EXLST, NIB, ASDP, and RPL) used by VTAM when the application program requests VTAM to perform actions on its behalf. When the request is accepted or completed, VTAM places values in the ACB, NIB, or the RPL control block as appropriate. The application program tests these values to determine the outcome of the request. This chapter discusses the ways in which the VTAM application program sets and tests these control block values.

Also discussed is certain VTAM information that is made available at assembly time in the form of macro global variables.

Setting and testing control block values

About this task

- Control block values can be set by:
  - Defining them in an ACB, EXLST, NIB, or RPL macroinstruction
  - Specifying values in operands of RPL-based macroinstructions
  - Using the manipulative macroinstructions GENCB and MODCB for certain fields
  - Using the INQUIRE OPTCD=TERMS macroinstruction to generate an NIB or list of NIBs
  - Using the DSECT-creating macroinstructions and assembler instructions to move values into specified fields.
- Control block values can be tested by:
  - Using the manipulative macroinstructions TESTCB and SHOWCB for certain fields
  - Using the DSECT-creating macroinstructions and assembler instructions to test values in specified fields.

Using manipulative macroinstructions

The macroinstructions that manipulate application program control blocks are:

- GENCB
- MODCB
- SHOWCB
- TESTCB

The advantages of these macroinstructions are that they:

- Provide in one instruction what would require several assembler language instructions
- Allow symbolic references to be made to control blocks and their fields without having to be concerned with their relative storage locations
- Can create control blocks in storage obtained dynamically, thereby allowing the application program to be reentrant.

© Copyright IBM Corp. 2000, 2013

The disadvantages of these macroinstructions are that they:

- Do not support newer parameters
- Are not efficient and expend CPU cycles.

For details about syntax and return codes for manipulative macroinstructions, refer to:

- Appendix I, "Return codes for manipulative macroinstructions," on page 855
- Appendix J, "Summary of operand specifications," on page 857
- Appendix K, "Forms of the manipulative macroinstruction," on page 865.

GENCB macroinstruction

GENCB builds and initializes an NIB, ACB, RPL, or EXLST. To use GENCB, this information is specified:

- The kind of control block built: ACB, NIB, RPL, or EXLST.
- The fields initialized and the values set in each field. For example, to build an RPL and initialize the OPTCD field to SYN, specify:

  ```
  GENCB BLK=RPL,AM=VTAM,OPTCD=SYN
  ```

- The number of copies of the control block built. Each copy initializes with the same values and can later modify with the onset of particular requests.
- Where the control block is built. The program defines an area where VTAM builds the control block. If an area is not specified, VTAM gets the storage from the system dynamically. VTAM returns the address of the created control block in register 1 and the length in register 0.

For application programs, any control block allocated using the GENCB macroinstruction resides in 24-bit storage.

Example 1

Build an ACB dynamically; initialize the EXLST field.

```
GENCB BLK=ACB,AM=VTAM,EXLST=MYLST
```

When GENCB is completed, register 1 contains the address of the new ACB; register 0 contains its length.

Example 2

Build 50 copies of an NIB dynamically. Initialize the processing options field to DFASYX and RESPX so that expedited-flow messages and response input on the sessions established with the NIBs causes a DFASY or RESP exit routine to be scheduled.

```
GENCB BLK=NIB,AM=VTAM,PROC=(DFASYX,RESPX),                    C
      COPIES=50
```

The program can calculate the length of each NIB by dividing the length in register 0 by the number of copies, and use this value to refer to each NIB from an RPL.

Example 3

A storage management technique is used whereby, for each LU, the program obtains a storage area to contain an RPL, an LU work area, and a data area. It issues GETMAIN to get storage for the entire area. Then, using a DSECT to map the area, it issues a GENCB to build an RPL.

The WAREA operand of the GENCB macroinstruction uses an S-type constant to point to a real storage area using the DSECT as a map:

```
         GETMAIN R,LV=LEN
         LR    2,1
         USING TWA,2
         GENCB BLK=RPL,AM=VTAM,WAREA=(S,XRPL),              C
         LENGTH=XRPLLEN
         .
         .
         .
TWA      DSECT
CHAIN    DS    F                  CHAINING POINTER
XRPL     DS    0F
         IFGRPL AM=VTAM,DSECT=NO
XRPLEND  EQU   *
WORKAREA DS    10F                LOGICAL UNIT WORK-AREA
DATA     DS    CL100              DATA AREA
END      EQU   *
LEN      EQU   END-TWA            LENGTH OF ELEMENT
XRPLLEN  EQU   XRPLEND-XRPL       LENGTH OF RPL
```

The SHOWCB macroinstruction can be used to examine the fields of this RPL:

```
          SHOWCB      AREA=SHOWAREA,RPL=(S,XRPL),           C
               FIELDS=FDBK2,LENGTH=4,AM=VTAM
          .
          .
          .
SHOWAREA  DS   F
```

MODCB macroinstruction

MODCB modifies the contents of an existing ACB, NIB, RPL, or EXLST. To use MODCB, this information is specified:

- The access method (VTAM)
- The kind of control block to be modified
- The symbolic name of the control block or a register that contains the address of the control block
- The fields to be modified.

A common use of MODCB is to modify an NIB during execution of a LOGON exit routine. Here are some examples:

Example 1

A LOGON exit routine has been entered, and it is necessary to put the symbolic name of the SLU into the NIB prior to session establishment. A pointer to the symbolic name of the SLU is in the parameter list pointed to by register 1 when the exit routine is entered. Because the NIB NAME field must have the symbolic name itself and not its address, the programmer codes:

```
         L     R4,4(R1)           POINT TO THE SYMBOLIC NAME
*                                 OF THE LOGICAL UNIT
         MODCB AM=VTAM,NIB=NIB1,NAME=(*,0(R4))   PUT IN NIB
```

Example 2

The entry for the LOGON exit routine in an exit list (labeled EX1) is to be changed to point to a routine named LOGON1:

```
MODCB AM=VTAM,EXLST=EX1,LOGON=LOGON1
```

Example 3

A pool of 50 RPLs has been created using GENCB. The address of that pool is in register 6. Later, to modify the OPTCD field in the first RPL, this macroinstruction is issued:

```
MODCB AM=VTAM,RPL=(6),OPTCD=SYN
```

To modify the second RPL in the same way, the program divides the number of copies (50) into the total length (contained in register 0) to obtain the length of one RPL (assume that register 4 contains the length of one RPL):

```
ALR  6,4                GET TO NEXT RPL
MODCB AM=VTAM,RPL=(6),OPTCD=SYN
```

Two restrictions govern the use of MODCB.

- An open ACB or an RPL or an NIB for a request that is being processed cannot be modified.
- New entries cannot be added to an EXLST; only addresses of existing entries can be changed.

SHOWCB macroinstruction

SHOWCB copies the values of selected fields in an ACB, NIB, RPL, or EXLST into a designated area. In using SHOWCB, this information is specified:

- The access method (VTAM)
- The kind of control block: ACB, NIB, RPL, or EXLST
- The symbolic name of the particular control block or a register that contains the address of the control block
- The fields to be copied. For example, FDBK and FDBK2 fields in an RPL, the CID field in an NIB, or the ERROR field in an ACB can be specified. The fields must be in the same control block.
- The name and length of a storage area in which VTAM places the contents of the named fields. This area must begin on a fullword boundary.

Example 1

Extract the 4-byte CID from an NIB whose address is in register 2; put the CID in an area defined as CIDAREA:

```
SHOWCB AM=VTAM,AREA=CIDAREA,NIB=(2),FIELDS=CID,              C
       LENGTH=4
```

Example 2

Extract the 4-byte address of the application program identification from the ACB labeled ACB1 and put it into an area labeled MYID:

```
SHOWCB AM=VTAM,AREA=MYID,ACB=ACB1,FIELDS=APPLID,             C
       LENGTH=4
```

Example 3

Extract the contents of the RTNCD and FDBK2 fields from the RPL whose address is in register 7 and store the contents contiguously at HERE:

```
SHOWCB AM=VTAM,AREA=HERE,RPL=(7),FIELDS=(RTNCD,FDB2),        C
       LENGTH=2
```

Example 4

This example shows how to build an ACB by using the SHOWCB macroinstruction to determine the length of the ACB. Using this method, the program issues a SHOWCB to obtain the 4-byte length of the ACB in an area labeled LENAREA. It then stores the length of the ACB in register 10 and issues a storage macroinstruction for the required amount of storage. Finally, it issues a GENCB to build the ACB at the address contained in register 5.

```
         SHOWCB AM=VTAM,FIELDS=ACBLEN,AREA=LENAREA,LENGTH=4
         L     10,LENAREA
         GETMAIN R,LV=(10)
         LR    5,1
         GENCB AM=VTAM,BLK=ACB,WAREA=(5),LENGTH=(10)
LENAREA  DS    F
```

TESTCB macroinstruction

TESTCB tests the value of a specific field in an ACB, NIB, RPL, or EXLST. In using TESTCB, this information is specified:

- The access method (VTAM)
- The kind of control block: ACB, NIB, RPL, or EXLST
- The name of the control block or register that contains the address of the control block
- The keyword for the field to be tested
- The value against which the field is to be tested
- Optionally, the name of a routine to be given control if VTAM cannot compare the two values.

To test the results of a TESTCB, the TESTCB macroinstruction can be followed with a branching instruction such as BE or BNE.

Some common uses of TESTCB are to test the ACB error flags when an ACB does not open properly and to test the FDBK2 field in the RPL after a data-transfer operation.

Example

If an ECB within the RPL is specified for posting, TESTCB can be used to determine which RPL has had its request completed. Use TESTCB to test the IO field of each RPL:

```
TEST     TESTCB AM=VTAM,RPL=(8),IO=COMPLETE
         BE    OUT
```

Using INQUIRE OPTCD=TERMS to generate NIBs

The INQUIRE OPTCD=TERMS macroinstruction can be used to build a single NIB or a list of NIBs. When the INQUIRE macroinstruction is issued, the RPL must point to an NIB whose NAME field contains the name of a resource known to VTAM. An NIB is built for each dependent LU belonging to the set defined by the name in the NIB. For example, if the name of a PU is in the NIB, NIBs are built for all the dependent LUs belonging to that PU; the independent LUs are no longer considered to be subordinate to the PU and are not returned in the NIB list.

When the macroinstruction is issued, the AREA and AREALEN fields in the RPL must designate the location and length of the work area where the NIBs are to be built. Before the macroinstruction is issued, the work area must be set to binary zeros.

If the application program wants the NIBs to be built in dynamically-allocated storage (storage obtained by the application program during execution), INQUIRE should be issued twice. For the first INQUIRE, set AREALEN to 0. This INQUIRE is completed with (RTNCD,FDB2)=(X'00',X'05') (insufficient length), and RECLEN indicates the required length. Obtain the required storage and then issue INQUIRE again with AREALEN set to the proper length.

After the macroinstruction is completed, the RECLEN field in the RPL contains the total length (number of bytes) for all NIBs generated by the macroinstruction. The NAME field of each NIB contains the symbolic name of the LU for which the NIB was generated, and each NIB contains the device characteristics for the LU it represents.

The LISTEND indicator is YES in the last NIB generated; all preceding generated NIBs are LISTEND=NO. Using the symbolic names and device characteristics, the application program can set PROC options and the MODE field in each NIB, and it can set other fields to desired values. The NIBs are then ready to be used for session establishment.

For further details, see "INQUIRE—Obtain logical unit information or application program status" on page 412.

Using DSECT-creating assembler instructions and macroinstructions

VTAM provides macroinstructions that generate a map of the fields and possible field values for each of the application program control blocks. Each macroinstruction generates a DSECT instruction (this can be optionally suppressed for some control blocks, as discussed in the following), a DS instruction for each field, and EQU instructions for certain predefined values. These macroinstructions and associated assembler instructions can be used as an alternative to or in combination with the manipulative macroinstructions. The DSECT-creating macroinstructions and assembler instructions that use the generated labels require execution of fewer instructions than using manipulative macroinstructions.

Defining the DSECTs

Appendix E, "Control block formats and DSECTs," on page 739, and Appendix F, "Specifying a session parameter," on page 793, show the DSECT fields and equated values generated by the macroinstructions. These are the DSECT-creating macroinstructions:

Table 48. DSECT—Creating macroinstructions

Control block/data area	DSECT name and operands	Appendix
ACB	IFGACB AM=VTAM[,DSECT=YES|NO]	E
EXLST	IFGEXLST AM=VTAM[,DSECT=YES|NO]	E
RPL	IFGRPL AM=VTAM[,DSECT=YES|NO]	E

Table 48. DSECT—Creating macroinstructions (continued)

Control block/data area	DSECT name and operands	Appendix
Device characteristics field in NIB	ISTDVCHR [DSECT=YES|NO]	E
Processing options field in NIB	ISTDPROC [DSECT=YES|NO]	E
Reason codes for RTNCD-FDB2-FDBK fields in RPL	ISTUSFBC	E
NIB	ISTDNIB	E
Session parameter	ISTDBIND	F
Program operator message and command header	ISTDPOHD	L
Request/response header	ISTRH	E
Buffer-list entry	ISTBLENT	E
Application-supplied dial parameters	ISTASDP	E
Access-method-support vector list	ISTAMSVL	E
Resource-information vector list	ISTRIVL	E
Application-ACB vector list	ISTVACBV	E

In coding these macroinstructions, follow these rules:

- Specify AM=VTAM as an operand for IFGACB, IFGRPL, and IFGEXLST. This distinguishes these DSECTs from similar DSECTs generated for VSAM.
- Code DSECT=NO as an operand if the DSECT assembler instruction is to be suppressed for IFGACB, IFGEXLST, IFGRPL, ISTDPROC, or ISTDVCHR. This allows the fields generated by these macroinstructions to be concatenated to the current CSECT or DSECT. See Example 3 in "GENCB macroinstruction" on page 268.
- Follow the macroinstruction with a CSECT assembler instruction or another DSECT assembler instruction unless it is the last instruction before ending the program or unless the map is to be extended intentionally. (A terminal work area could be mapped following IFGRPL, for example.)
- Ensure that no labels that are generated or reserved for future use are used elsewhere in the program.

  Other areas of the program use the following labels. Therefore, the application program should not use them:

```
ACB     DEV     NIB     RPL
ASD     EXL     POH     RSV
BIN     IFG     PRO     USF
BLE     IST     RHASD
```

Using the DSECTs

Having used the DSECT-creating macroinstructions to define one or more control-block maps, the program can obtain storage for a control block from an assembled pool or dynamically from the system. The address of this storage should be placed in a register and specified in a USING statement before setting or testing values using the statements generated by the macroinstruction. For example, suppose the address of an RPL is in register 5 and that, after a request containing data has been received, it is necessary to determine the value of the RECLEN field of the RPL.

If IFGRPL AM=VTAM is coded, the statement:

```
USING IFGRPL,5
```

allows assembler language instructions to refer to the label RPLRLEN to obtain the record length. Further information is in the introductory section of Appendix E, "Control block formats and DSECTs," on page 739.

ISTGLBAL macroinstruction

A VTAM macroinstruction available for use in making assembly-time decisions is ISTGLBAL. ISTGLBAL declares and sets a group of macro global variables that describe the VTAM product associated with the macroinstruction definition library containing the ISTGLBAL macroinstruction. Once declared and set, the variables can be interrogated during assembly of the application program to determine whether specific functions or options of VTAM are supported.

ISTGLBAL can be invoked directly, or by either IFGRPL or IFGACB as an inner macroinstruction.

The three types of macro global variables set by ISTGLBAL are:

- Release-level macro global variable (&ISTGLRL)
- Component-ID macro global variable (&ISTGLCI)
- Function-list macro global variables (&ISTGL*xy*where *x* and *y* denote a specific variable).

Details on the macro global variables are provided in this section. A description of the ISTGLBAL macroinstruction is in "ISTGLBAL" on page 639.

Release-level and component-ID macro global variables

When ISTGLBAL, IFGRPL AM=VTAM, or IFGACB AM=VTAM is assembled as part of an application program module, the &ISTGLRL macro global variable is declared and set with a character string that specifies the release level of the library containing ISTGLBAL. The &ISTGLCI macro global variable is declared and set with a character string that specifies the component identifier of VTAM. Later in the assembly of that module, these GBLC variables can be accessed by other macroinstructions and by open code.

The character string with which &ISTGLRL is set indicates the following:

Table 49. Release-level macro global variables for VTAM V6R1.2

Byte no.	Contents	Code indication
0	0	Product (0=VTAM)
1	6	Version 6
2	1	Release 1
3	2	Modification Level 2

The following example shows how the variable might be used in a macroinstruction defined by an application program for V6R1.2:

```
         MACRO
         WHATVTAM
         GBLC  &ISTGLRL
         AIF   ('&ISTGLRL' EQ '0612').LABEL1
VTAMLEVL DC    C'THIS IS NOT V6R1.2 OF VTAM'
```

```
         MEXIT
.LABEL1  ANOP
VTAMLEVL DC    C'THIS IS V6R1.2 OF VTAM'
         MEND
```

If the V6R1.2 level of either ISTGLBAL, IFGRPL, or IFGACB is coded, the WHATVTAM macroinstruction causes the generation of the first DC statement at assembly time. If you use an earlier release of ISTGLBAL, IFGRPL, or IFGACB, or if none of these three macros are issued in the assembly, the second DC statement is generated, indicating that the level of VTAM used is not VTAM V6R1.2.

Function-list macro global variables

ISTGLBAL, IFGRPL AM=VTAM, and IFGACB AM=VTAM also declare and set the list of GBLB global variables to indicate specific functions supported by your release. These variables have names of the form ISTGL*xy*, where *x* specifies a byte in the function-list vector and *y* specifies a bit in that byte. See "The access-method-support vector list" on page 62 for information about how the bits in the function-list vector are initialized for the base operating system and for features supported by VTAM. The corresponding macro global variables are set to the same values when ISTGLBAL, IFGRPL AM=VTAM, or IFGACB AM=VTAM is issued.

If a variable has a value of 1, the corresponding function is present; a value of 0 means the function is not present. The information to be contained in the function-list vector can, therefore, be determined at assembly time or at execution time.

To examine the information, the source code for the application program must first issue ISTGLBAL, IFGRPL AM=VTAM, or IFGACB AM=VTAM, and then declare the variable to be tested with a GBLB statement naming the variable. (See "Release-level and component-ID macro global variables" on page 274 for an example of how a GBLC variable can be used; a similar macroinstruction or open code could be written to test one or more of the GBLB variables.)

Chapter 9. Handling errors and special conditions

This chapter discusses how to analyze information for errors and special conditions and what to do, in general, when the error or special condition is identified. Identifying and acting upon errors and special conditions are discussed separately for:

- OPEN or CLOSE macroinstructions
- Manipulative macroinstructions
- RPL-based requests (for example, SEND, RECEIVE, and CHECK).

Note: For a description of the software error analysis that is operating-system-dependent, refer to "Error handling" on page 322.

The information that VTAM returns to the application program is organized so that a minimum amount of checking is required. For most macroinstructions, if register 15 contains 0 on return to the program, no further checking is necessary; the program proceeds normally. If register 15 contains a value other than 0, checking proceeds until the condition is defined and appropriate action is taken. For certain RPL-based macroinstructions, register 0 must be examined for a special condition even if register 15 contains 0.

OPEN and CLOSE errors and special conditions

Most ERROR settings indicate an error in program logic or some failure to match the name of an ACB as specified in the program with its name as specified during VTAM definition. A dump, program termination, and debugging are required. If multiple ACBs are being opened and only some have been opened successfully, it might be possible to continue with the programs whose ACBs are opened.

Register 15 should be tested after the OPEN or CLOSE macroinstruction. If the return code in register 15 is 0, all ACBs have been opened or closed as requested. If the return code does not equal 0, one or more ACBs were not properly opened or closed. When this occurs, the TESTCB macroinstruction can be used to test the OFLAGS field of each ACB to see if it is open:

```
TESTCB      AM=VTAM,ACB=(3),OFLAGS=OPEN
```

In this example, the address of an ACB is in register 3. If an OPEN macroinstruction failed (except for the error that occurs when the ACB is opened prior to this OPEN), the failing ACB does not have OFLAGS=OPEN, and the ACB is still closed. If a CLOSE macroinstruction failed (except for the error that occurs when the ACB is closed prior to this CLOSE), the failing ACB has OFLAGS=OPEN because the ACB is not closed. Once the failing ACB has been located, the SHOWCB macroinstruction can be used to look at the error bits in the ERROR field of the ACB. Of course, neither the OFLAGS nor the ERROR field can be examined if OPEN originally pointed to an area that did not contain an ACB.

See Chapter 13, "Conventions and descriptions of VTAM macroinstructions," on page 371, for the description of the format and contents of the ERROR field in the OPEN and CLOSE macroinstruction. Figure 67 on page 278 shows how OPEN/CLOSE error and special-condition information is organized.

© Copyright IBM Corp. 2000, 2013

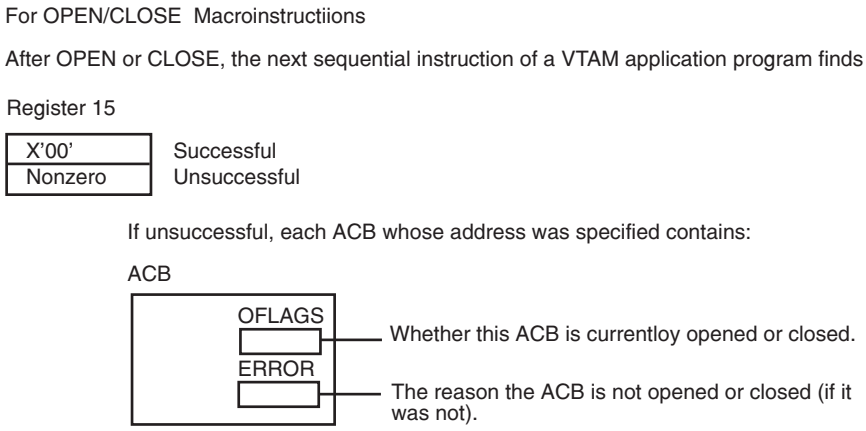


Figure 67. How OPEN/CLOSE error and special-condition information is organized

Manipulative macroinstruction errors and special conditions

Table 50. Manipulative macroinstructions: Return code values and meanings

Reg	Contents	Meaning
0	X'08'	GENCB can fail because of insufficient storage; request can be reissued later
15	X'00'	Manipulative macroinstruction operation is successful
15	X'04'	Specific error in register 0 (examining return code for SHOWCB or TESTCB does not require examining a control block). Appendix I, "Return codes for manipulative macroinstructions," on page 855 shows the possible register 0 settings and their meanings.
15	X'08'	Attempt is made to use execute form of the macroinstruction to enter a new item in parameter list being modified (only existing items can be modified)

Figure 68 on page 279 shows how manipulative macroinstruction error and special-condition information is organized. All other manipulative macroinstruction errors and special conditions are due to faulty logic and require program termination and debugging.

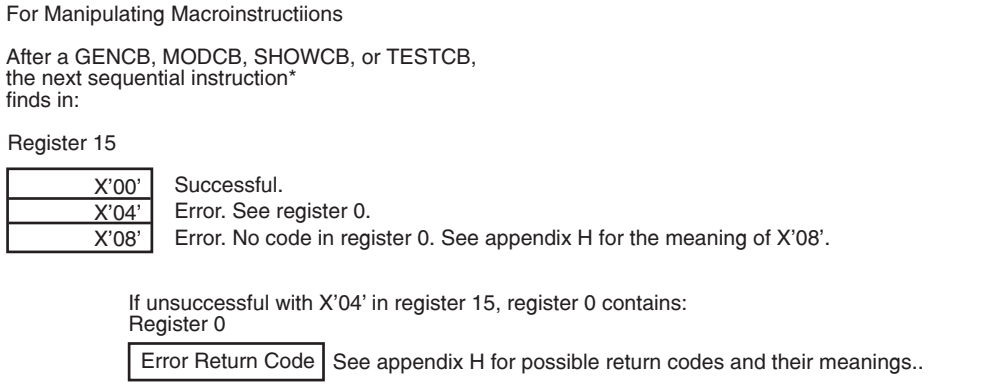


Figure 68. How manipulative-macroinstruction error and special-condition information is organized

RPL-based macroinstruction errors and special conditions

There are two kinds of RPL-based operations: synchronous and asynchronous. Chapter 3, "Organizing an application program," on page 33, provides an overview of these modes of operation. For synchronous RPL-based operations, a single macroinstruction is issued. On return to the VTAM application program, error or special-condition information about the requested operation is available.

For asynchronous RPL-based operations, two RPL-based macroinstructions are required: a request macroinstruction and a CHECK (completion) macroinstruction. Error and special-condition information can be returned at two different stages: as a result of the request for the operation being accepted or not accepted and, if the request is accepted, as a result of the operation completing successfully or unsuccessfully.

Following an RPL-based macroinstruction, information is available to the application program about the acceptability of the request or about the completion of the operation. This information can be provided by VTAM; or, if an error or special condition was detected and VTAM invoked the program's LERAD or SYNAD exit routine, register information is provided by the LERAD or SYNAD exit routine. The information consists of a return code in register 15 (termed a general return code), in some cases a return code in register 0 (termed either a recovery action return code or a conditional completion return code), and information in the RPL (termed feedback information). Feedback information includes the RPL RTNCD field, which contains the recovery action return code, and the RPL FDB2 field, which contains either a conditional completion return code or a specific error return code; other RPL fields can be set depending upon the individual RPL-based macroinstruction. Table 51 on page 280, Figure 69 on page 281, and Table 52 on page 280 show how this information is organized.

Table 51. Summary of register and RPL feedback return codes following an RPL-based request

Register 15 (General return code)	Meaning	Register 0	RPL feedback information		RTNCD
			FDB2	LERAD/SYNAD entered (if specified)	
X'00'	Normal acceptance, normal or conditional completion	Conditional completion return code (X'00' means normal completion)	Recovery action return code (=X'00')	Conditional completion return code	No
X'04'	Request not accepted or request completed abnormally	Recovery action return code (or user-defined value)	Recovery action return code	Specific error return code	Yes
X'20'	ACB not open	RPL request code	RPL not changed	RPL not changed	No

Notes:

1. If a LERAD or SYNAD exit routine is specified, it can exit with user-defined values in registers 15 and 0. These are then the registers returned to the part of the program that issues the synchronous RPL-based request or CHECK macroinstruction.
2. If the RPL is not valid (recovery action return code = X'18'), there is no RPL feedback information.

Table 52. Recovery action return codes and their general meanings

Recovery action return code (in RPL RTNCD field)	LERAD or SYNAD exit scheduled	Type of completion	Programmer action
X'00'	No exit scheduled	Normal or conditional	Continue normal or conditional processing.
X'04'	SYNAD	Exceptional condition	Analyze RPL to choose logic path.
X'08'	SYNAD	Retriable completion	Use EXECRPL macroinstruction to retry if desired.
X'0C'	SYNAD	Data integrity damage	Execute user program error recovery coding.
X'10'	SYNAD	Environment error	Call for external intervention.
X'14'	LERAD	Logic error	Dump program status and continue or abend.
X'18' (in register 0 but not in RTNCD field)	LERAD	Logic error; RPL not valid	Dump program status and continue or abend. Do not reuse this RPL.
Others	LERAD or SYNAD	RPL overwritten	Dump program status and continue or abend. Do not reuse this RPL.

See Table 53 on page 286 and Table 54 on page 288 for more information about completion types.

For RPL-Based Macroinstructions

After a SEND, RECEIVE, CHECK, or other RPL-based macroinstruction, the next sequential instruction finds in:

Register 15

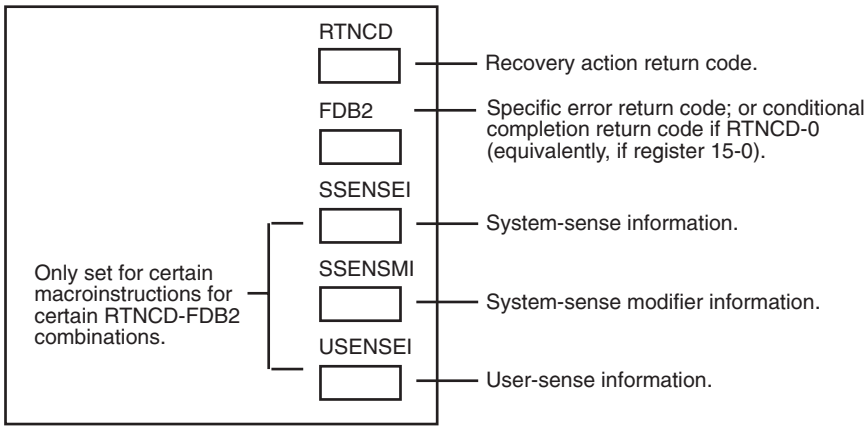
X'00'	The request was successful; or, for a synchronous request (including CHECK), the operation was successful.
some other value	The request or the operation was not successful.

Depending on the request and whether it is successful, it might be necessary to test:

Register 0*

If register 15 is X'00', register 0 indicates (with a conditional completion return code) for certain macroinstructions whether success was conditional. If register 15 is not X'00', register 0 can contain a return code from a LERAD or SYNAD exit routine or, if there is no LERAD/SYNAD exit routine, register 0 can contain a recovery action return code (generally the code from the RTNCD field of the RPL.)

If a request or operation was unsuccessful or conditionally successful, these RPL fields can be examined (in either the issuing routine or in a LERAD or SYNAD exit routine) to determine the cause of the exeption.



In addition, other RPL fields that contain feedback information (such as SEQNO, CHAIN, and CHNGDIR), normally used following the completion of a requested operation, can be used in determining jow to handle an error or special condition.

RPL fields are described under the RPL and other macroinstruction descriptions and are summarized in Appendix A.

Possible RPL RTNCD, FDB2, and sense information settings and their meanings are in Appendix B.

For Arrival of a Logical Unit (LU) Status Request

After receiving input with a RECEIVE specifying RTYPE = (DFSYN, NRESP) and CONTROL = LUS, the RPL inbound sense fields (SSENSEI, SSENSMI, and USENSEI) contain error or special condition information from the LU.

*Register 0 is of interest in these circumstances:

For certain macroinstructions with certain options set (see Appendix B), if register 15 contains X'00', success might be conditional. Register 0 should be examined to see if there is a condition (and what it is).

If an error occurred for an RPL-based macroinstruction and no LERAD or SYNAD exit routine is available, register 15 contains X'04', and register 0 contains a recovery action return code.

If a LERAD or SYNAD exit routine is available, it can set register 15 to X'00' to indicate "Error corrected-request or operation successful." If the error is not corrected, register 15 should be a value other than 0 and a return code be passed from LERAD or SYNAD in register 0.

If the ACB was not opened, register 0 contains the RPL request code.

Figure 69. How RPL-based macroinstruction error and special-condition information is organized.

For recovery action return codes, see Table 52 on page 280. If an error or special condition occurred, it can be analyzed and handled in either the mainline program or exit routine where the RPL-based request was issued, or in a designated LERAD or SYNAD exit routine.

It is convenient to use LERAD and SYNAD exit routines to handle error and special conditions for all RPL-based requests in a program. A LERAD or SYNAD exit routine can be entered for an asynchronous operation at two different times: when the initial request for the operation fails and, if the request is accepted, after the operation fails. The LERAD or SYNAD exit routine can set register 15 to 0 so that the request-issuing part of the program is not aware that an error or special condition occurred and continues normally. If the request-issuing part of the program must be made aware that an error or special condition occurred, register 15 can be set to a value other than 0 and a user-specified return code placed in register 0 (in this case, register 15 can also be used as a return code register between the LERAD or SYNAD exit routines and the issuing part of the program).

If a LERAD or SYNAD exit routine is not used, errors and special conditions can be handled in the inline coding that follows the request or CHECK, or it can be processed in a common subroutine.

Figure 70 on page 283, Figure 71 on page 284, and Figure 72 on page 285 show the relationship between the requesting part of the program and LERAD and SYNAD exit routines, if present. These figures can also be used to code the instructions that check the results of requests in the requesting parts of the program. Table 53 on page 286 and Table 54 on page 288 list in detail the completion conditions for RPL-based macroinstructions.

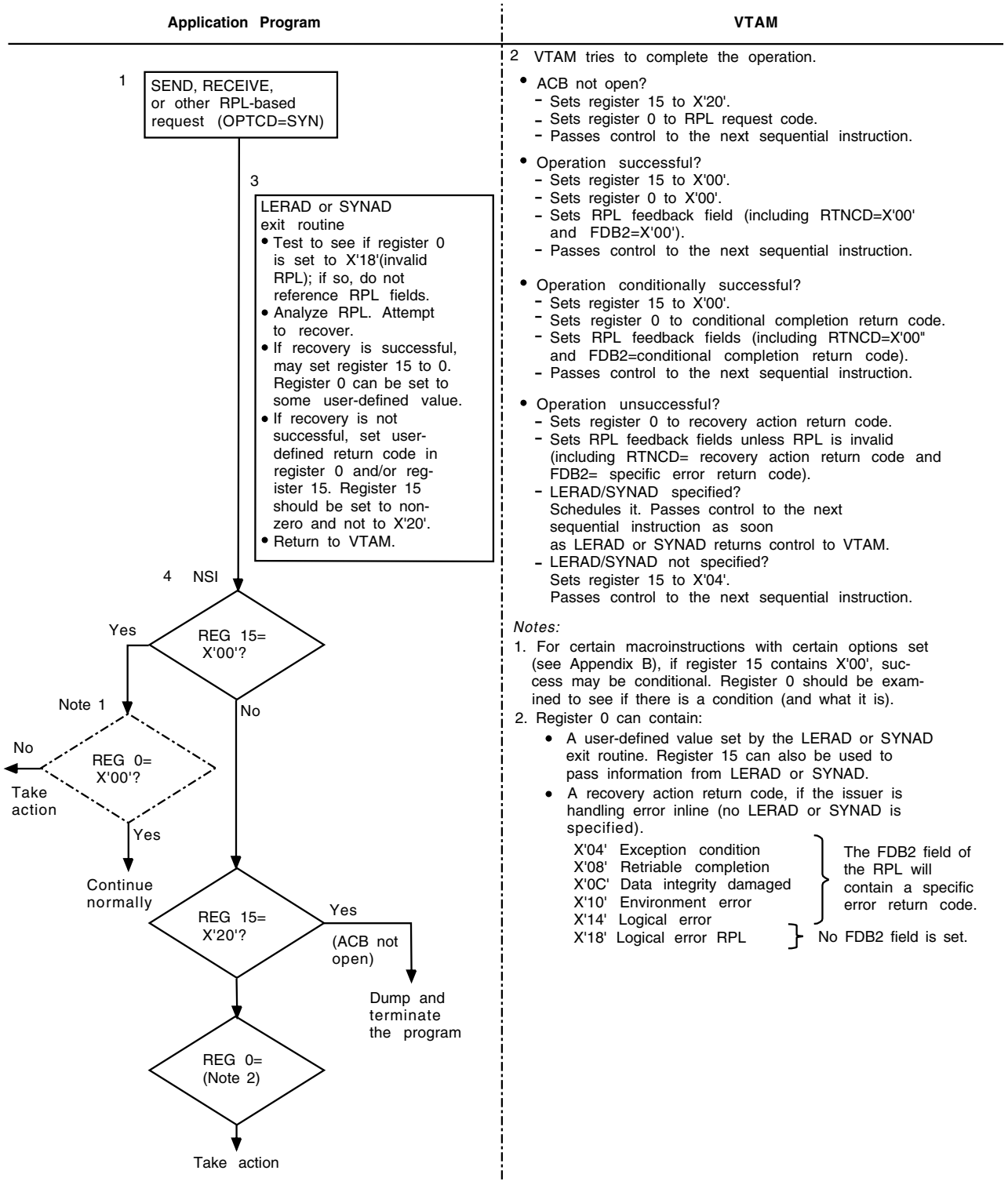
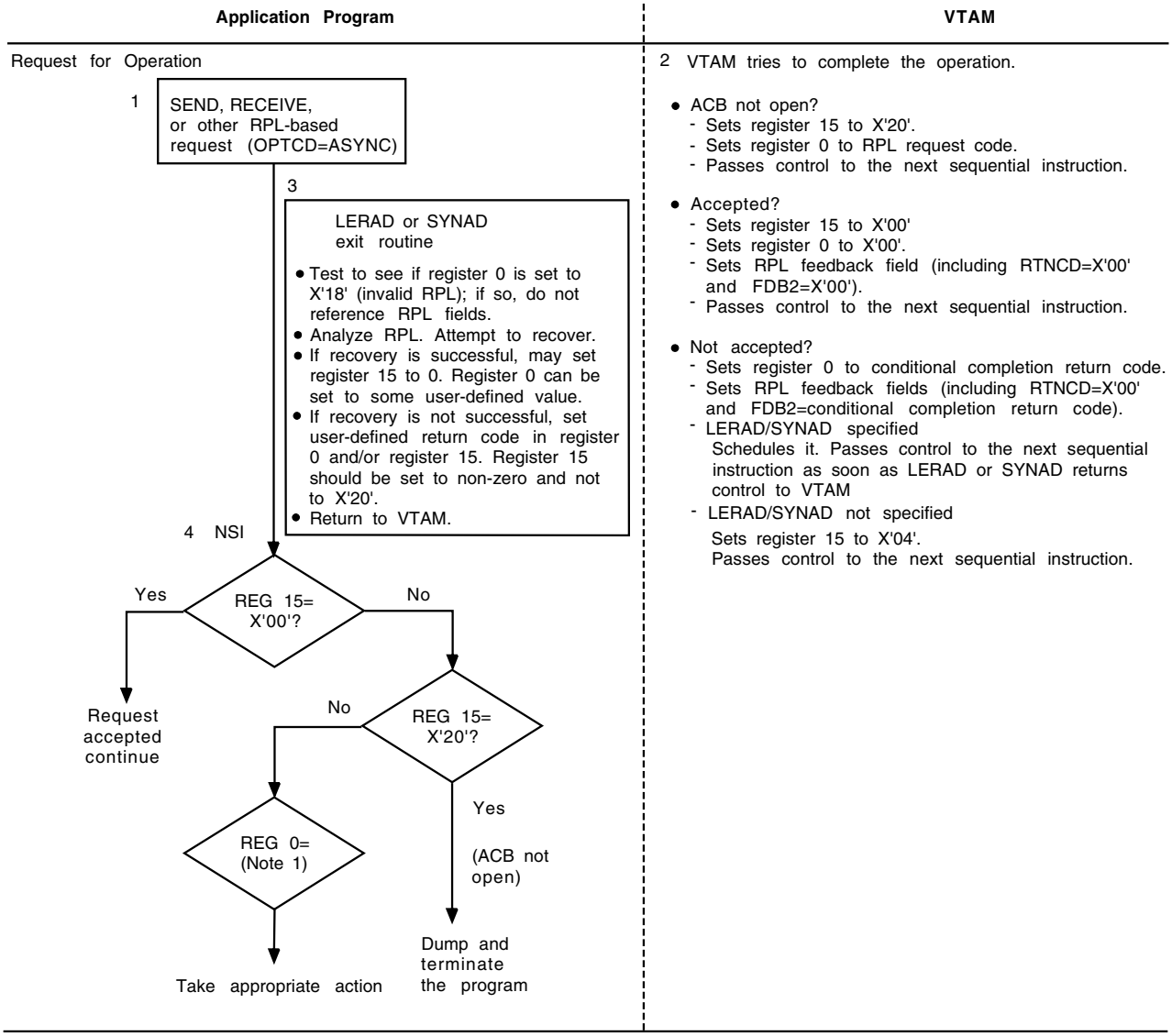


Figure 70. Summary of error and special-condition handling with synchronous operations



Notes:

1. Register 0 can contain:
   - A user-defined value set by the LERAD or SYNAD exit routine. Register 15 can also be used to pass information from LERAD or SYNAD.
   - A recovery action return code, if the issuer is handling error inline (no LERAD or SYNAD is specified).

     X'04' Exception condition
     X'08' Retriable completion
     X'0C' Data integrity damaged
     X'10' Environment error
     X'14' Logical error
     } The FDB2 field of the RPL will contain a specific error return code.

     X'18' Logical error RPL
     } No FDB2 field is set.

2. The CHECK macroinstruction can be issued (1) following discovery of a posted ECB for the RPL, (2) to wait for the ECB to be posted, or (3) after an RPL exit routine has been invoked. CHECK can be issued within an exit routine (for example, in the RPL exit routine for this RPL) or the main program. However, CHECK cannot be issued before the RPL exit routine (if specified) has been invoked. The logic of this figure applies whether ECB posting or RPL exit scheduling is used to notify the program of operation completion. Rather than issuing CHECK immediately after it is notified of operation completion, the program can first look at the RPL feedback fields. CHECK is required sooner or later, however, to set the RPL inactive and to schedule the LERAD or SYNAD exit (if specified) for error conditions.
3. For certain macroinstructions with certain options set (see Appendix B), if register 15 contains X'00', success may be conditional. Register 0 should be examined to see if there is a condition (and what it is).

Figure 71. Summary of error and special-condition handling with asynchronous operations (Part 1 of 2)

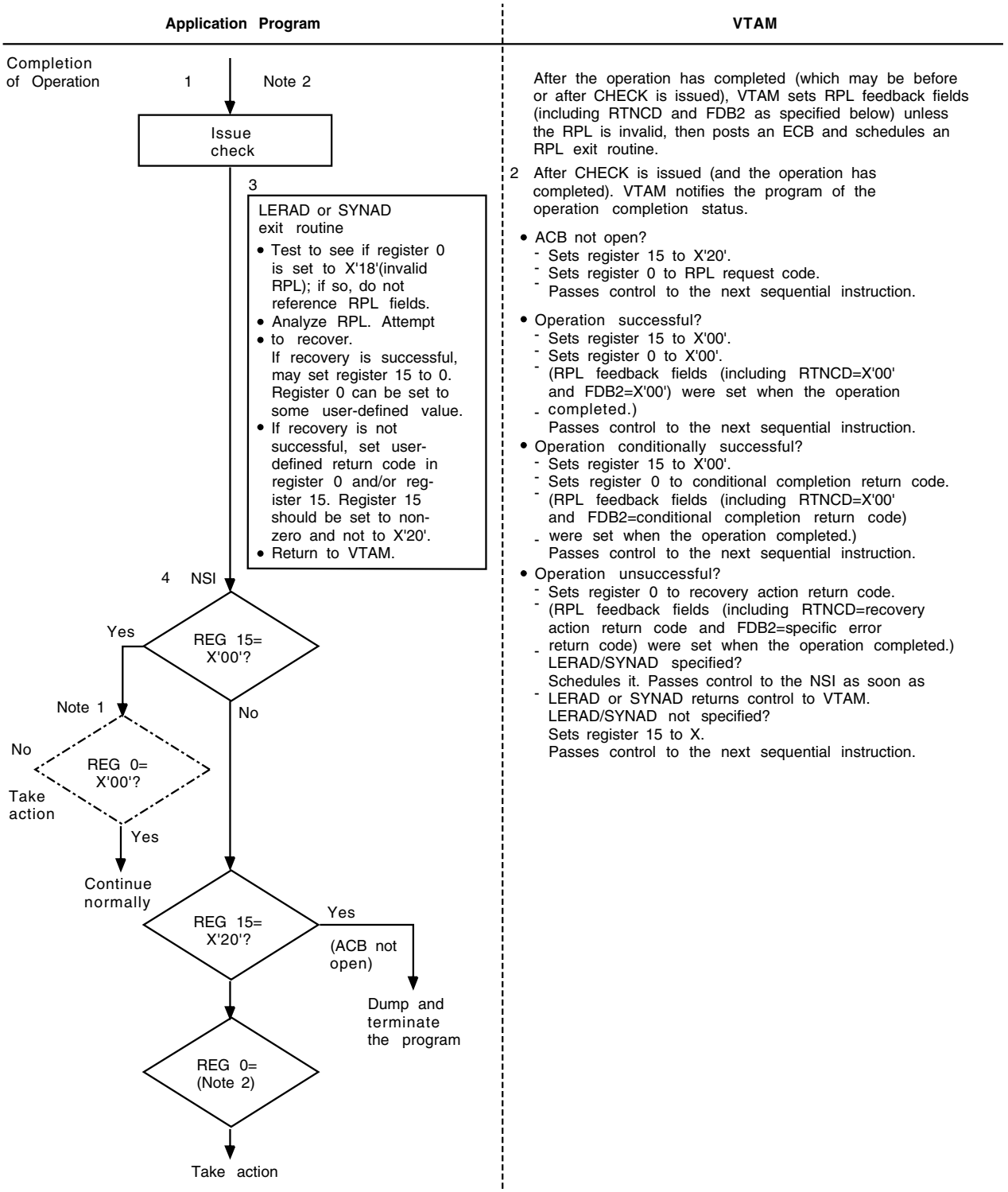


Figure 72. Summary of error and special-condition handling with asynchronous operations (Part 2 of 2)

Table 53. Completion conditions acceptance stage of asynchronous requests

Completion condition	Explanation	Example	Registers at NSI when SYNAD-LERAD not available	Registers at entry to SYNAD-LERAD when SYNAD-LERAD are available	Registers at NSI when SYNAD-LERAD are available	RPL feedback fields set	Suggested programmer action (in LERAD, SYNAD, or after next sequential instruction)
Request accepted (general return code=0, recovery action return code=0).	VTAM has accepted the asynchronous request and processes it. Completion information is available when the request is completed.		Reg 15 = 0 Reg 0 = 0	(LERAD-SYNAD not entered)	Reg 15 = 0 Reg 0 = 0 (LERAD-SYNAD not entered)	RTNCD=0 FDB2=0	
Request not accepted (general return code=4)							
Retriable completion (recovery action return code=8)	VTAM has rejected the asynchronous request because of a temporary situation.	A temporary storage shortage has occurred.	Reg 15 = 4 Reg 0 = 8	Reg 15 = Address of SYNAD exit routine Reg 0 = 8	Reg 15 Reg 0 Set by SYNAD exit routine	RTNCD=8 FDB2=0	Issue an EXECRPL macro to retry the request.
Environment error (recovery action return code=16)	VTAM has rejected the asynchronous request because of an environmental condition beyond the control of the application program.	VTAM not active.	Reg 15 = 4 Reg 0 = 16	Reg 15 = Address of SYNAD exit routines Reg 0 = 16	Reg 15 Reg 0 Set by SYNAD exit routine	RTNCD=16 FDB2= specific error return code	Request external intervention (from the VTAM operator, for example) or suspend processing.
Logical error (recovery action return code=20)	VTAM has rejected the asynchronous request because the request violates the requirements defined in this book.	NIB's MODE field not set to RECORD.	Reg 15 = 4 Reg 0 = 20	Reg 15 = Address of LERAD exit routine Reg 0 = 20	Reg 15 Reg 0 Set by LERAD exit routine	RTNCD=20 FDB2= specific error return code	Obtain a program dump and correct the program.

Table 53. Completion conditions acceptance stage of asynchronous requests (continued)

Completion condition	Explanation	Example	Registers at NSI when SYNAD-LERAD not available	Registers at entry to SYNAD-LERAD when SYNAD-LERAD are available	Registers at NSI when SYNAD-LERAD are available	RPL feedback fields set	Suggested programmer action (in LERAD, SYNAD, or after next sequential instruction)
Logical error with RPL that is not valid (recovery action return code=24)	VTAM has rejected the asynchronous request because the RPL address points to an active RPL or does not point to any RPL or request is issued from an asynchronous exit routine.	An attempt was made to reuse an RPL to which no CHECK had been issued.	Reg 15 = 4 Reg 0 = 24	Reg 15 = Address of LERAD exit routine Reg 0 = 24	Reg 15 Reg 0 Set by LERAD exit routine	RPL not set and should not be examined.	Obtain a program dump and correct the program.
Request not accepted because of a prior OPEN failure (general return code=32; no recovery action return code).	The RPL points to an ACB that has not been properly opened or that has been closed.		Reg 15 = 32 Reg 0 = Request code (see description of RPL's REQ field)	LERAD-SYNAD not entered	Reg 15 = 32 Reg 0 = Request code (see description of RPL's REQ field) (LERAD-SYNAD not entered)	RPL not set	Obtain a program dump and correct the program.

Table 54. Completion conditions completion of synchronous requests or for CHECK of asynchronous requests

Completion condition	**Explanation**	**Example**	**Registers at NSI when SYNAD-LERAD not available**	**Registers at entry to SYNAD-LERAD when SYNAD-LERAD are available**	**Registers at NSI when SYNAD-LERAD are available**	**RPL feedback fields set**	**Suggested programmer action (in LERAD, SYNAD, or after next sequential instruction)**
Normal completion (general return code=0)	The request has been completed successfully. For some macros (see right-most column), register 0 contains additional information.	INQUIRE was issued and the input area was too small.	Reg 15 = 0 Reg 0 = Conditional completion return code	LERAD-SYNAD not entered	Reg 15 = 0 Reg 0 = Conditional completion return code (LERAD-SYNAD not entered)	RTNCD=0 FDB2= conditional completion return code	INQUIRE, INTRPRET, OPNDST, RCVCMD, RECEIVE, and SIMLOGON can be completed normally with special conditions preset. If these conditions are meaningful to the application program, check register 0 or FDB2.
Abnormal completion (general return code=4)							
Exception condition (recovery action return code=4)	An exception request or a negative response has been received.	An exception request has been received.	Reg 15 = 4 Reg 0 = 4	Reg 15 = Address of SYNAD exit routine Reg 0 = 4	Reg 15 Reg 0 Set by SYNAD exit routine	RTNCD=4 FDB2= specific error return code	Take whatever action is appropriate for the FDB2 return code.
Retriable completion (recovery action return code=8)	VTAM cannot complete the request because of a temporary condition.	A temporary storage shortage has occurred.	Reg 15 = 4 Reg 0 = 8	Reg 15 = Address of SYNAD exit routine Reg 0 = 8	Reg 15 Reg 0 Set by SYNAD exit routine	RTNCD=8 FDB2= specific error return code	Issue an EXECRPL macro to retry the request.

Table 54. Completion conditions completion of synchronous requests or for CHECK of asynchronous requests (continued)

Completion condition	**Explanation**	**Example**	**Registers at NSI when SYNAD-LERAD not available**	**Registers at entry to SYNAD-LERAD when SYNAD-LERAD are available**	**Registers at NSI when SYNAD-LERAD are available**	**RPL feedback fields set**	**Suggested programmer action (in LERAD, SYNAD, or after next sequential instruction)**
Data integrity damage (recovery action return code=12)	VTAM cannot complete the request. The session has been lost or the request was reset by another operation on the session.	A hardware error occurred that disrupted the session path.	Reg 15 = 4 Reg 0 = 12	Reg 15 = Address of SYNAD exit routine Reg 0 = 12	Reg 15 Reg 0 Set by SYNAD exit routine	RTNCD=12 FDB2= specific error return code	Take whatever action is appropriate for the FDB2 return code. In general, the process that was interrupted should be restarted.
Environment error (recovery action return code=16)	VTAM cannot complete the request. The problem cannot be resolved without external intervention.	VTAM not active	Reg 15 = 4 Reg 0 = 16	Reg 15 = Address of SYNAD exit routine Reg 0 = 16	Reg 15 Reg 0 Set by SYNAD exit routine	RTNCD=16 FDB2= specific error return code	Take whatever action is appropriate for the FDB2 return code. In general, the LU is unavailable and any session with it should be terminated or VTAM operator intervention should be requested.
Logic error (recovery action return code=20)	VTAM cannot complete the request because it violates the requirements defined in this book.	The RPL does not indicate a valid RPL request code.	Reg 15 = 4 Reg 0 = 20	Reg 15 = Address of LERAD exit routine Reg 0 = 20	Reg 15 Reg 0 Set by LERAD exit routine	RTNCD=20 FDB2= specific error return code	Obtain a program dump and correct the program.

Table 54. Completion conditions completion of synchronous requests or for CHECK of asynchronous requests (continued)

Completion condition	Explanation	Example	Registers at NSI when SYNAD-LERAD not available	Registers at entry to SYNAD-LERAD when SYNAD-LERAD are available	Registers at NSI when SYNAD-LERAD are available	RPL feedback fields set	Suggested programmer action (in LERAD, SYNAD, or after next sequential instruction)
Logic error with RPL that is not valid (recovery action return code=24)	VTAM cannot complete the request because the RPL address does not point to any RPL, or because the address points to an active RPL and this macro is synchronous (not CHECK), or because the RPL exit routine specified for this RPL has not been scheduled and CHECK was issued, or a request was issued from an asynchronous exit routine.	The RPL address was destroyed before the request was executed.	Reg 15 = 4 Reg 0 = 24	Reg 15 = Address of LERAD exit routine Reg 0 = 24	Reg 15 Reg 0 Set by LERAD exit routine	RPL not set and should not be examined	Obtain a program dump and correct the program.
Abnormal completion because of a prior OPEN failure (general return code=32, no recovery action return code).	VTAM cannot complete the request because the RPL points to an ACB that has not been properly opened or that has been closed.		Reg 15 = 32 Reg 0 = Request code (see description of RPL's REQ field)	LERAD-SYNAD not entered	Reg 15 = 32 Reg 0 = Request code (see description of RPL's REQ field) LERAD-SYNAD not entered	RPL not set	Obtain a program dump and correct the program.

Coding LERAD and SYNAD exit routines

The following display shows the use of registers on entering and leaving a LERAD or SYNAD exit routine. Addressability, save area, and re-entrance requirements are described in "Procedures to follow in writing exit routines" on page 231. Examples

of LERAD and SYNAD exit routines are shown in the sample VTAM application program in Chapter 15, “Sample code of a simple application program,” on page 579.

On entering a LERAD or SYNAD exit routine:

Register	Contains
0	Recovery action return code (RTNCD field of RPL)
1	Address of the RPL
2–13	Contents as they are when RPL-based request was issued
14	Address at which control can be returned to VTAM
15	Address of the LERAD or SYNAD exit routine.

On leaving a LERAD or SYNAD exit routine:

Register	Contains
0	User return code from LERAD or SYNAD
1	Address of the RPL (although VTAM restores it)
2–12	Any value (VTAM restores the contents as they were when request was issued)
13	Address of the save area from the request-issuing part of the program
14	Address to which control is being passed
15	Optional user return code (in addition to register 0).

LERAD and SYNAD exit routines can have a common entry point; the logic at the common entry point can determine whether the error is a logic error or a physical error, and branch to the appropriate error-handling instructions.

The recovery action return code in register 0 (and in the RTNCD field of the RPL when the RPL is valid) can be used to branch to separate subroutines in the LERAD or SYNAD exit routine. This might be coded:

```
         LR    R2,Register 0       SINCE Register 0 NOT USABLE AS INDEX REG
         B     *(R2)               BRANCH TO APPROPRIATE BRANCHING
*                                  INSTRUCTION
         B     EXCEPTN             EXCEPTION CONDITION (R2='04')
         B     REISSUE             RETRIABLE COMPLETION (R2='08')
         B     DINTDAM             DATA INTEGRITY DAMAGE (R2='0C')
         B     ENVERR              ENVIRONMENT ERROR (R2='10')
         B     USLOGIC             USER LOGIC ERROR (R2='14')
         B     NOTINRPL            USER LOGIC ERROR BUT INFORMATION
*                                  IS NOT IN THE RPL (R2='18')
```

The logic that these subroutines, each representing a general category of recovery action, might contain is discussed in the following sections.

Handling exception conditions (register 0=04)

If a request arrives from the other end of the session for which VTAM detects an error (for example, the request arrived with a sequence number that was not one greater than the previous sequence number), VTAM puts X'04' in register 0, X'04' in

the recovery action return code field (RTNCD) of the RPL, and X'03' in the specific error return code field (FDB2) of the RPL. It also puts values in the SSENSEI, SSENSMI, and USENSEI fields of the RPL. Unless the other end of the session does not want a response (the RESPOND field contains NFME,NRRN), the VTAM application program, either in the SYNAD exit routine or perhaps later in some other part of the program, must send a response. An exception to this rule is this: if a chain of requests is being received, and a negative response has been returned to a request that was received as part of the chain, and remaining requests of the chain are being received as exception requests, only the first negative response should be sent and the remaining requests should be disregarded.

Before sending the negative response, the values in the SSENSEI, SSENSMI, and USENSEI fields must be transferred to the SSENSEO, SSENSMO, and USENSEO fields of the RPL used to send the response.

Sense codes are discussed in "System-sense information" on page 680.

Further action by a PLU application program

In addition to transferring these values for sending response information, the PLU application program can analyze these fields to determine what to do next. It can:

- Begin sequence number resynchronization, using the SESSIONC macroinstruction.
- Await a Request Recovery (RQR) request from the LU (which would cause scheduling of the program's SCIP exit routine) and then begin sequence number resynchronization. In this case, the LU must be capable of deciding whether to shut down or to request recovery.
- If in the process of receiving a chain of requests, set a flag to purge the buffer that contains requests in this chain that have been previously received. It can set a flag to indicate to itself that the rest of the requests in this chain are to be disregarded. When the last request arrives, the program can turn off this flag.

Further action by an SLU application program

When an SLU application program learns of an error in an incoming request, it also transfers the values in the SSENSEI, SSENSMI, and USENSEI fields to the SSENSEO, SSENSMO and USENSEO fields of the RPL used to send the negative response. In addition, the SLU application program can:

- Send a Request Shutdown (RSHUTD) request to the PLU application program, if sequence number resynchronization is required but is not possible.
- Send a Request Recovery (RQR) request to ask the PLU application program to begin sequence number resynchronization, if required. In this case, the SLU application program awaits a Clear request followed by a Set and Test Sequence Numbers (STSN) request from the PLU application program.
- If in the process of receiving a chain of requests, set a flag to purge the buffer that contains requests in this chain that have been previously received. It can set a flag to indicate to itself that the rest of the requests in this chain are to be discarded. When the last request arrives, the program can turn off this flag.

The exact action to be taken by each end of the session in the event of receiving an exception request error must be agreed upon by the programmers coding the PLU and SLU application programs before they are coded.

Further information on sequence number resynchronization action can be found in:

- "Controlling flow" on page 164
- Figure 109 on page 696

- Figure 119 on page 706
- Figure 128 on page 715.

Chapter 6, “Communicating with logical units,” on page 151, discusses the various ways that a response can be received by an application program. If the response is negative, the SSENSEI, SSENSMI, and possibly the USENSEI fields must be analyzed to determine what recovery action is possible. Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651, describes in general the possible settings of these fields. Appendix E, “Control block formats and DSECTs,” on page 739, shows the DSECT labels of these fields. In general, certain settings require that the session with the LU be terminated; other settings indicate that the situation is recoverable and that either sequence number synchronization or some other action can be taken.

Handling retriable completion (register 0=08)

This return code indicates that an operation was not successful but should be requested again. The RPL does not contain any further information. The program should issue an EXECRPL macroinstruction using the same RPL. The parameters in the RPL do not apply to the EXECRPL request itself but to the original request already specified in the REQ field of the RPL (in other words, the RPL associated with the request that is to be retried). The EXECRPL simply retries the request.

An error on retrying an operation can cause LERAD or SYNAD to be entered again. So that this situation can be recognized, a flag can be set before issuing the EXECRPL or other RPL-based macroinstruction. See “Procedures to follow in writing exit routines” on page 231.

Handling data integrity damage (register 0=0C)

Several errors fall into this category; the action to be taken depends on conditions that are unique to each program. For example, a program that has no means of recovering data that is lost on its way to an LU can decide that the LU has to revert to some earlier point in communications or inquiry (might have to reenter its inquiry), and sends a request to that effect. The LU can then reenter its request and start the process at an initial stage. On the other hand, a program can always keep a copy of its latest transmission to an LU so that it can retransmit it, if required.

Handling environment errors (register 0=10)

In general, this category requires action by a VTAM operator or terminal operator or program support representative because the LU, or some communication element between the application program and the LU, is permanently or temporarily unavailable. One or more of the following might be done:

- Send a message to the VTAM operator.
- Send a message to any log that is being kept in addition to the error logs being kept by VTAM.
- Send a request to a master LU.
- Send a request to an LU that is associated with the LU for which action is required.

If the error is temporary, the program can set a flag and retry the operation at a later time or set up an ECB that, when posted, indicates that external action has

taken place and the operation can be resumed. If the error is permanent, the session with the LU (and perhaps other LUs associated with the failing LU) must be terminated. If the program is dependent on the LU, the program must terminate.

Handling logic errors (register 0=14 and register 0=18)

Most return codes that cause entry to a LERAD exit routine are likely to occur only when the program is being debugged. These errors require that the program save as much information as necessary for debugging, perhaps request a dump of storage, and terminate the program at that point. After the program is debugged, a message to the operator can be substituted for program termination in the event that a logic error occurs.

Chapter 10. Operating system facilities

This chapter describes a number of operating-system-dependent facilities to use when writing a VTAM application program. It is assumed that the reader is knowledgeable about the particular operating system involved. Read "Normal operating system environment for a VTAM application program" on page 30 to understand how an application program appears if these special facilities are not used.

VTAM macroinstruction differences across operating systems

Operating system differences are stated in the individual macroinstructions.

If you are planning to migrate application programs from a non-ESA operating system to a z/OS operating system, several differences exist because of the increase in operating system capability in z/OS. These differences result from services available under z/OS (primarily for authorized application programs) that are not available in non-ESA operating systems. Additionally, if you are planning to use the 31-bit addressing capabilities available to application programs, further considerations exist (primarily for addressing mode and residence mode).

The main VTAM macroinstruction differences to consider are summarized here, with references to the parts of this book where they are described in detail. The list can be used as a checklist of VTAM macroinstruction language considerations when converting a VTAM application program from one operating system to another, or when changing a VTAM application program to use additional operating system capabilities. There are many other considerations in such a conversion, such as operating-system-dependent storage management and data management macroinstructions that can be used by the application program:

- OPEN and CLOSE macroinstruction operand specifications, register 15 return code handling, and ACB error code values. See Chapter 9, "Handling errors and special conditions," on page 277, and Chapter 13, "Conventions and descriptions of VTAM macroinstructions," on page 371, for details on how to handle error information and macroinstruction coding.
- Use of operating-system-dependent information for application program identification for OPEN processing if no identification is specified through the ACB. See Chapter 4, "Opening and closing an application program," on page 55, for more information related to the ACB. See "OPEN—Open one or more ACBs" on page 444, for details on OPEN processing for application program identification.
- TPEND exit routine reason codes. See "TPEND exit routine is entered" on page 74, for more information on the TPEND exit routine. See Chapter 7, "Using exit routines," on page 219, for details on how to use the TPEND exit routine.
- Manipulative macroinstruction return codes. See Chapter 9, "Handling errors and special conditions," on page 277, and Appendix I, "Return codes for manipulative macroinstructions," on page 855, for information on handling return codes.
- DSECT and control block differences. See Appendix E, "Control block formats and DSECTs," on page 739.
- VTAM operator message formats for program operators. See Appendix L, "Program operator coding requirements," on page 875.

© Copyright IBM Corp. 2000, 2013

- Cryptographic capabilities. See Chapter 5, “Establishing and terminating sessions with logical units,” on page 81, for information on single and cross-domain sessions. See Chapter 6, “Communicating with logical units,” on page 151, for information on how to identify sessions. See Chapter 13, “Conventions and descriptions of VTAM macroinstructions,” on page 371, for details on VTAM macroinstructions.
- Multitasking capabilities. See “Multitasking” for information about multitasking and dividing communication activity.
- Authorized path for macroinstructions and SRB capability for exit routines (see “Authorized path” on page 300 for information on authorized path for macroinstructions). The user should be familiar with certain limitations when running under control of an SRB. (Refer to the z/OS MVS Programming: Authorized Assembler Services Guide)
- Serialized versus concurrent processing by different parts of an application program, described in “Serialization of execution” on page 309.
- Component ID vectors, function list vectors, and macro global variables are different for different operating systems. See Chapter 4, “Opening and closing an application program,” on page 55, for more information on vectors and variables for each operating system. See Chapter 8, “Setting and testing control blocks and macro global variables,” on page 267, for more information on how to set and test control block values.
- Multiple-address-space capabilities, described in “Multiple address spaces” on page 312.
- 31- versus 24-bit addressing capability, described in “31-bit addressing” on page 319.
- Software error handling, described in “Error handling” on page 322.

Assigning operating system authorization

The use of several of the facilities described in this chapter requires the application program to be authorized by the operating system. VTAM determines whether an application program is an authorized program when OPEN ACB is issued for that program.

Authorization criteria

z/OS considers the program authorized if one or more of the following are true:

- The program is an authorized program facility (APF) designated program.
- The program is in supervisor state.
- The program is executing with a protection key 0–7 inclusive.

For further information about operating system authorization, refer to the z/OS MVS Programming: Authorized Assembler Services Guide. This manual also describes the use of the operating system MODESET macroinstruction, which is used to enter supervisor state.

Multitasking

In addition to the multithreading facilities provided by VTAM (described in “Single-thread or multithread operations” on page 36), the operating system provides multitasking facilities that can be used when writing a VTAM application program to improve performance and availability. Multitasking can be used to

separate communication activity from other activity (such as disk I/O), to divide communication activity among several tasks, or to do both.

Separating data communication activity from other activity

Multitasking can be used so that communication activity can occur while waiting for other activity, such as disk I/O processing, to be completed (see Figure 73). For example, a VTAM application program can be organized into a task that opens and closes the ACB and performs VTAM requests, and a task that performs disk I/O requests (using VSAM, for example). In such a program, a page fault in the task that performs disk I/O requests does not prevent the task that performs communication requests from getting control during the time that the system is waiting for the required page to arrive in main storage. For a single-task VTAM application program, a page fault in that task would require that the entire program wait.

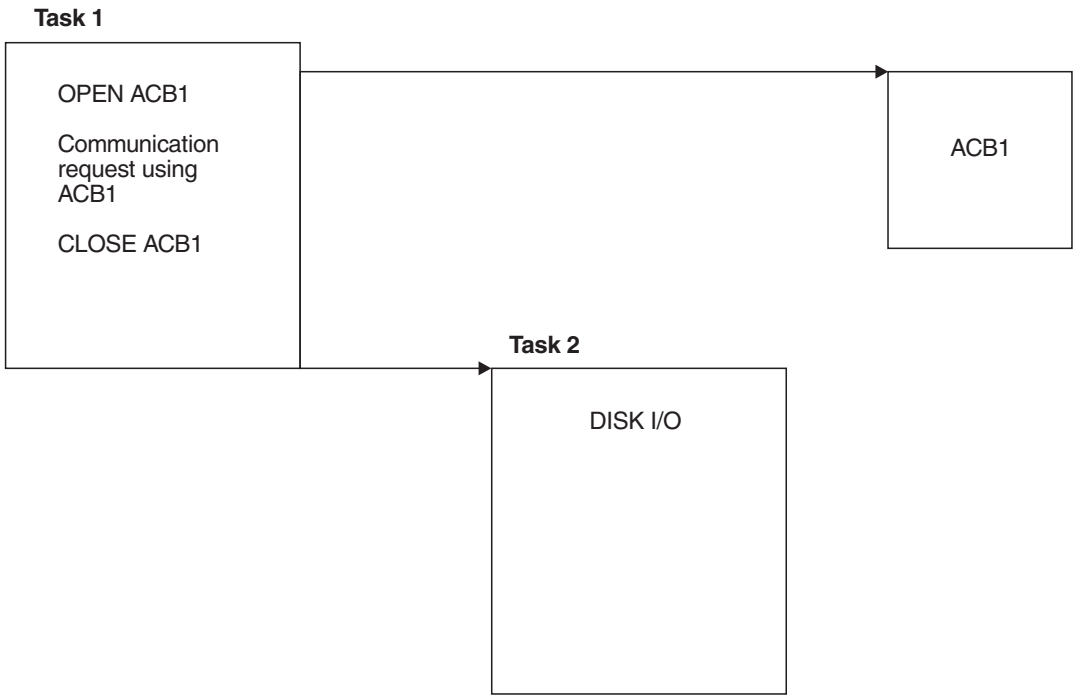


Figure 73. Multitasking a program

Dividing data communication activity among several tasks

Further efficiency might be possible by issuing VTAM requests in more than one task. For example, a program can use one task to open and close the ACB and to establish and terminate sessions. The program can use a number of other tasks, each containing a RECEIVE that specifies OPTCD=ANY and additional RPL-based communication requests, to communicate on sessions. Whenever one such VTAM communication task has to wait, the system can schedule another VTAM communication task.

There are two basic ways to use multitasking to divide communication activity among several tasks:

- You can write a program so that the first task attaches subtasks and all tasks use the same ACB.
- You can write a program so that the first task attaches subtasks and each task uses a separate ACB.

Multiple tasks, using the same ACB

When multiple tasks created by a program use the same ACB (see Figure 74), the following considerations apply:

- The macroinstructions (OPEN and CLOSE) that open and close the ACB must be in the same task.
- A subtask that did not open the ACB should not intentionally terminate if it has outstanding VTAM requests for that ACB.
- The task that closes the ACB should ensure that other tasks refrain from issuing VTAM requests during and after CLOSE processing.
- An exit routine can run under a different task (or SRB) than a communication task. See "Task association" on page 310. If any of the VTAM communication tasks are dependent on information that can be detected in such an exit routine, the exit routine must be able to communicate with that task (perhaps by posting an ECB located in a common area).
- Any abnormal termination in an asynchronous exit routine associated with the task that opened the ACB results in the abnormal termination of that task as well as all of its subtasks.

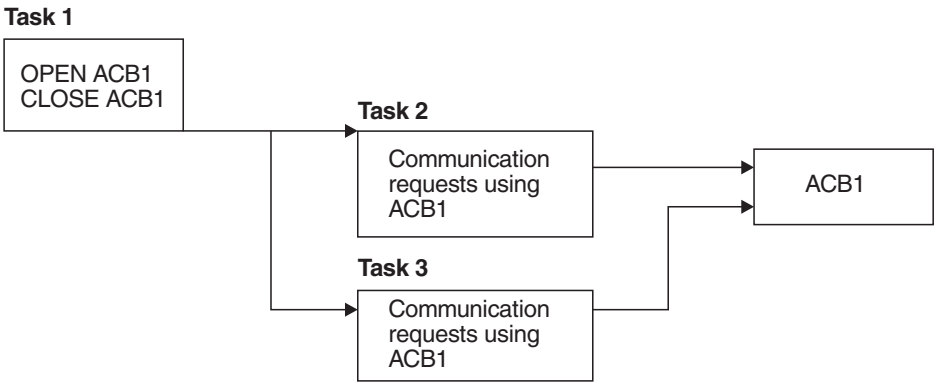


Figure 74. Multiple tasks using the same ACB

Multiple tasks, each with its own ACB

In a VTAM application program consisting of more than one task, each task can open its own ACB (see Figure 75 on page 299). In such a structure, the macroinstructions (OPEN and CLOSE) that open and close a particular ACB must be issued in the same task.

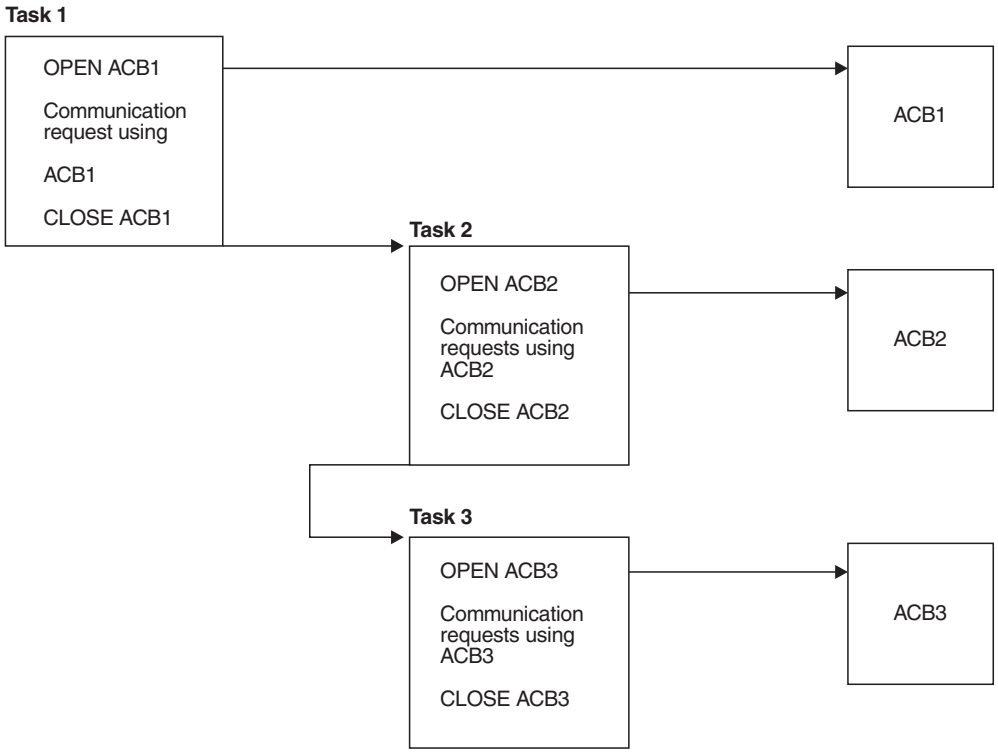


Figure 75. Multiple tasks, each with its own ACB

Using multiple ACBs within one task

About this task

A program can open more than one ACB (see Figure 76 on page 300), and thus become multiple VTAM application programs. This can be done under multiple tasks as discussed in the preceding section; it can also be done under a single task. Some ACB exit routines can be used in common by multiple ACBs, while other exit routines can be associated with only one particular ACB. A possible use of multiple ACBs is to code multiple programs which can be brought online at different times of the day, yet which can share many common routines.

Note: Subsequent OPEN ACBs under the same task should ignore the SRBEXIT parameter.

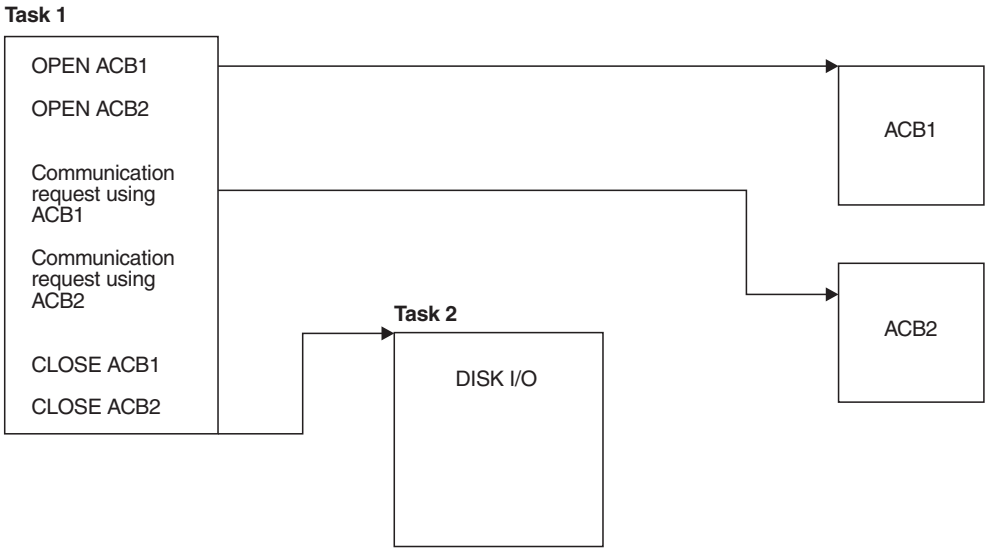


Figure 76. Single task with multiple ACBs

Authorized path

A VTAM application program can specify that RPL-based macroinstructions be executed by VTAM in a manner that generally requires fewer instructions than the normal mode of operation. The path length (number of executed instructions) is significantly decreased for the SEND, RECEIVE, RESETSR, and SESSIONC macroinstructions. This facility, called authorized path, can be used to improve the performance of a VTAM application program. Such an application program must be authorized as described in "Assigning operating system authorization" on page 296. Macroinstructions that can use authorized path are listed in Figure 77 on page 301. The OPEN, CLOSE, and manipulative macroinstructions cannot use authorized path; they cannot be issued under an SRB, but instead must run under a task control block (TCB mode).

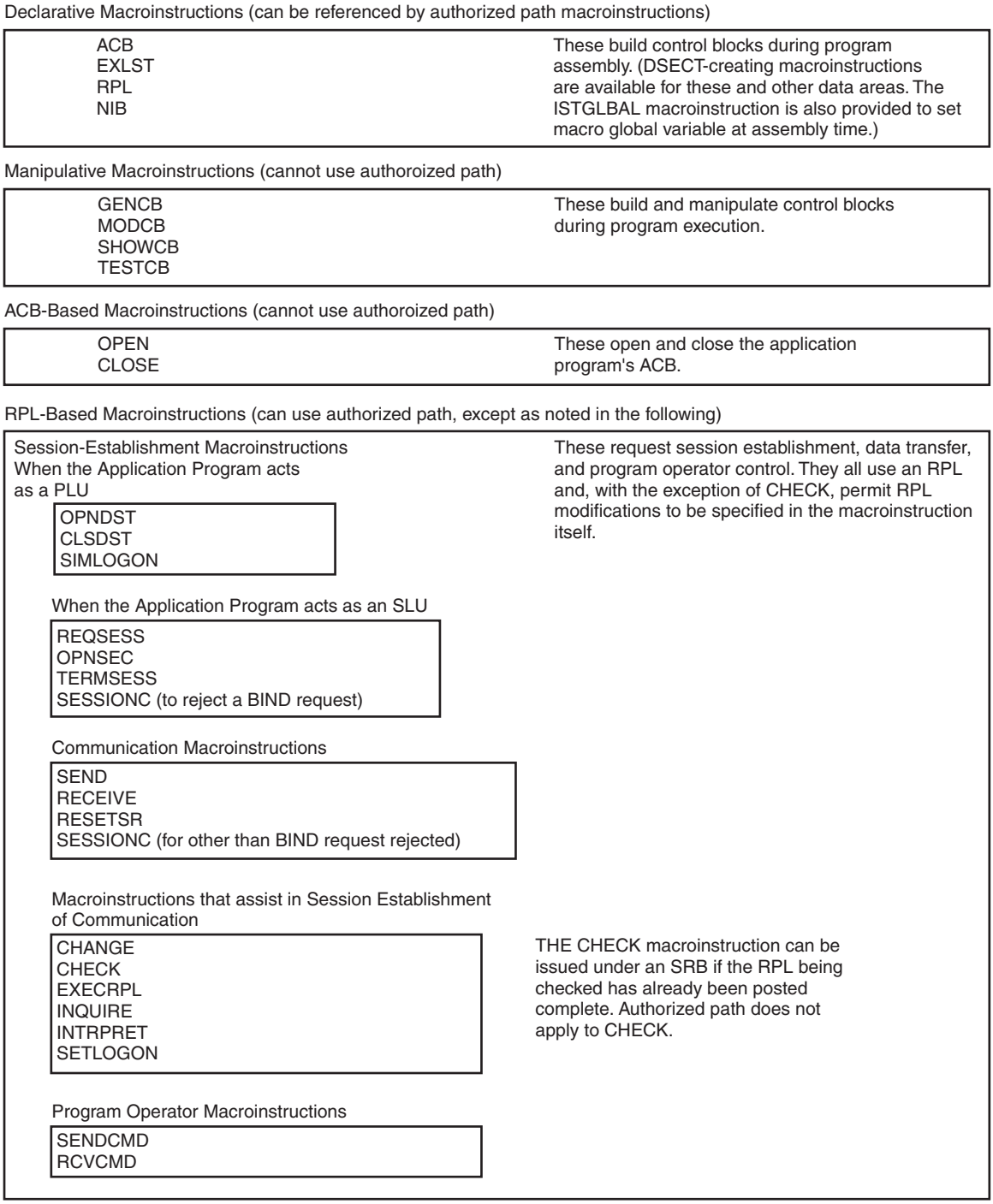


Figure 77. Categories of VTAM macroinstructions versus the authorized path function

Specifying authorized path macroinstructions

The VTAM authorized path macroinstructions can be executed asynchronously (OPTCD=ASY) or synchronously (OPTCD=SYN) under control of a service request block (SRB), or a task control block (TCB). Table 55 on page 302 shows the coding requirements for VTAM authorized path, including the VTAM authorized path

macroinstructions and the supervisor macroinstructions needed to execute the authorized path through VTAM. Consult the z/OS MVS Programming: Authorized Assembler Services Guide for information on coding the supervisor macroinstructions described in Table 55.

To use authorized path while running under a TCB, the authorized program, having put itself into supervisor state, specifies BRANCH=YES on any RPL-based macroinstruction that is to be executed using authorized path.

Subsequently, to issue any macroinstruction that is not to use authorized path and that uses the same RPL, the RPLBRANC flag in the RPL must be turned off by one of the following:

- Coding BRANCH=NO on a MODCB macroinstruction
- Referring to the field by using the IBM-supplied DSECT and turning it off with an assembler language instruction
- Coding BRANCH=NO on the subsequent macroinstruction that is not to use authorized path.

Authorized path is always used when an RPL-based macroinstruction is issued under the control of an SRB. A task identification and address space identification should be specified in the SRB as discussed in "Task association" on page 310; otherwise (RTNCD,FDB2)=(X'14',X'55') could result. One way to gain control under an SRB is for the authorized program, while running under a TCB, to specify an RPL exit routine when issuing (in supervisor state) an RPL-based macroinstruction that specifies BRANCH=YES. On entry to the RPL exit routine, the program is running under an SRB. Any RPL-based macroinstruction issued under an SRB is automatically executed using authorized path; BRANCH=YES need not be specified. An alternative way to create the SRB environment is to use the MVS SCHEDULE macroinstruction.

Table 55. Coding requirements for authorized path

	Method of program execution (OPTCD=ASY or SYN operand specified on VTAM RPL and RPL-based macroinstructions)	
Method of dispatching	**Synchronous**	**Asynchronous**
TCB	1. MODESET MODE=SUP 2. BRANCH=YES operand on authorized path VTAM macroinstruction	1. MODESET MODE=SUP 2. BRANCH=YES operand on authorized path VTAM macroinstruction 3. EXIT or ECB operand on VTAM RPL or other VTAM RPL-based macro instruction
SRB	1. MODESET EXTKEY=ZERO 2. SCHEDULE with operands 3. SETFRR with operands	1. MODESET EXTKEY=ZERO 2. SCHEDULE with operands 3. SETFRR with operands 4. EXIT or ECB operand on VTAM RPL or other VTAM RPL-based macroinstruction

Additional coding considerations for authorized path

When an application program executes two or more asynchronous authorized path macroinstructions for the same session, the requests might not be satisfied by

VTAM in the order that the macroinstructions are issued. To ensure that VTAM processes requests in order, wait until one macroinstruction is posted complete before executing the next.

Here are some examples of how requests can be processed out of order:

- If two SEND POST=RESP,OPTCD=ASY macroinstructions are executed on a given session without an intervening CHECK macroinstruction, the second request could arrive at the logical unit before the first request. This can happen if both authorized path macroinstructions are issued under two different SRBs.
- Similarly, if multiple RECEIVE macroinstructions are outstanding for a session, SRB scheduling can make it appear that input arrived on the session out of order. For example, the RECEIVE EXIT SRB for the input data request with a sequence number of 8 can execute prior to the RECEIVE EXIT SRB scheduled for input data request number 7. This can be avoided by having no more than one RECEIVE outstanding at a time for a session and issuing another RECEIVE only when the first completes.

RPL exit routines for authorized path macroinstructions are always scheduled under SRBs (in supervisor state and key 0), even if the authorized path macroinstruction was invoked under a TCB. On entry to such an RPL exit routine:

- Register 1 contains the address of the RPL.
- Register 13 does not contain a save area address because no save area is provided. (This is also true for an RPL exit routine running under a TCB.)
- Register 14 contains the return address of the dispatcher.
- Register 15 contains the entry-point address of the exit routine.

The SRB under which an RPL exit routine is run or under which control is returned to an application program is not necessarily the same SRB under which the original RPL-based macroinstruction was issued. For synchronous SRB-mode macroinstruction requests, the application optionally can request that VTAM retain the SRB of the issuer's processing thread throughout the processing of the request. This is accomplished by specifying OPTCD=(SYN,KEEPSRB) on the RPL-based macroinstruction request. This permits the processing environment established by the application, including FRRs, BAKR stacks, and other SRB-related resources, to be kept intact.

CAUTION:
To provide this function, VTAM utilizes SUSPEND and RESUME. Suspending the SRB (as opposed to exiting and returning under a different SRB) allows the environment to be preserved. However, SUSPEND and RESUME may impact performance. Take this into account when making use of this parameter for performance sensitive API invocations (such as SEND or RECEIVE).

When part of an application program is executed under the control of an SRB, that part of the application program cannot issue a WAIT macroinstruction or any other SVC except ABEND, SVC 13. VTAM does any suspension and resumption of processing for that part of the application program without using WAIT or POST macroinstructions (for example, when OPTCD=SYN is issued).

A CHECK macroinstruction can be issued if asynchronous ECB posting is used, but it can usually be issued only under the control of a TCB. A CHECK macroinstruction cannot be issued under an SRB unless it is certain that the associated RPL has been posted complete; for example, CHECK for an RPL can be issued in the RPL-exit routine scheduled for that RPL because the RPL is posted

complete before the exit routine is scheduled. If CHECK is issued under an SRB before the RPL has been posted complete, the CHECK issues a WAIT, but the WAIT SVC is not allowed under an SRB.

When an RPL exit routine is executing under the control of an SRB, the exit routine should establish a functional recovery routine (FRR) by using the supervisor macroinstruction SETFRR. For additional information about the use of functional recovery routines, see "Functional recovery routines" on page 324.

To ensure that VTAM serializes its references to VTAM control blocks, VTAM authorized users should not execute the SEND, RECEIVE, SESSIONC, or RESETSR macroinstructions while a CLSDST or TERMSESS macroinstruction for the same session is in progress.

An authorized program that holds system locks should release those locks before invoking VTAM to avoid conflicts with VTAM's use of the locks. Otherwise, if a conflict occurs, z/OS abnormally terminates the VTAM function being processed for that program.

SYNAD and LERAD exit routines are scheduled in the same mode (TCB or SRB) as the program that issued the macroinstruction giving control to the exit routine. For asynchronous requests, the SYNAD or LERAD exit routine can be entered either when the authorized path macroinstruction is issued or when a CHECK macroinstruction is issued.

Rules relating to the reentrance of certain VTAM exit routines are described in Chapter 7, "Using exit routines," on page 219. When the RPL, SYNAD, and LERAD exit routines are scheduled as a result of authorized path and corresponding CHECK macroinstructions being issued, the following two additional rules apply:

- RPL exit routines must be reentrant, because VTAM schedules RPL exit routines in parallel under different SRBs. If the RPL exit routine invokes any authorized path or CHECK macroinstructions, the SYNAD and LERAD exit routines must also be reentrant.
- If the application program itself schedules parallel processing under different SRBs, and if these SRBs invoke any authorized path or CHECK macroinstructions, the SYNAD and LERAD exit routines must be reentrant.

Simple example of authorized path usage

See Chapter 15, "Sample code of a simple application program," on page 579, for a coded example of an application program that uses authorized path macroinstructions.

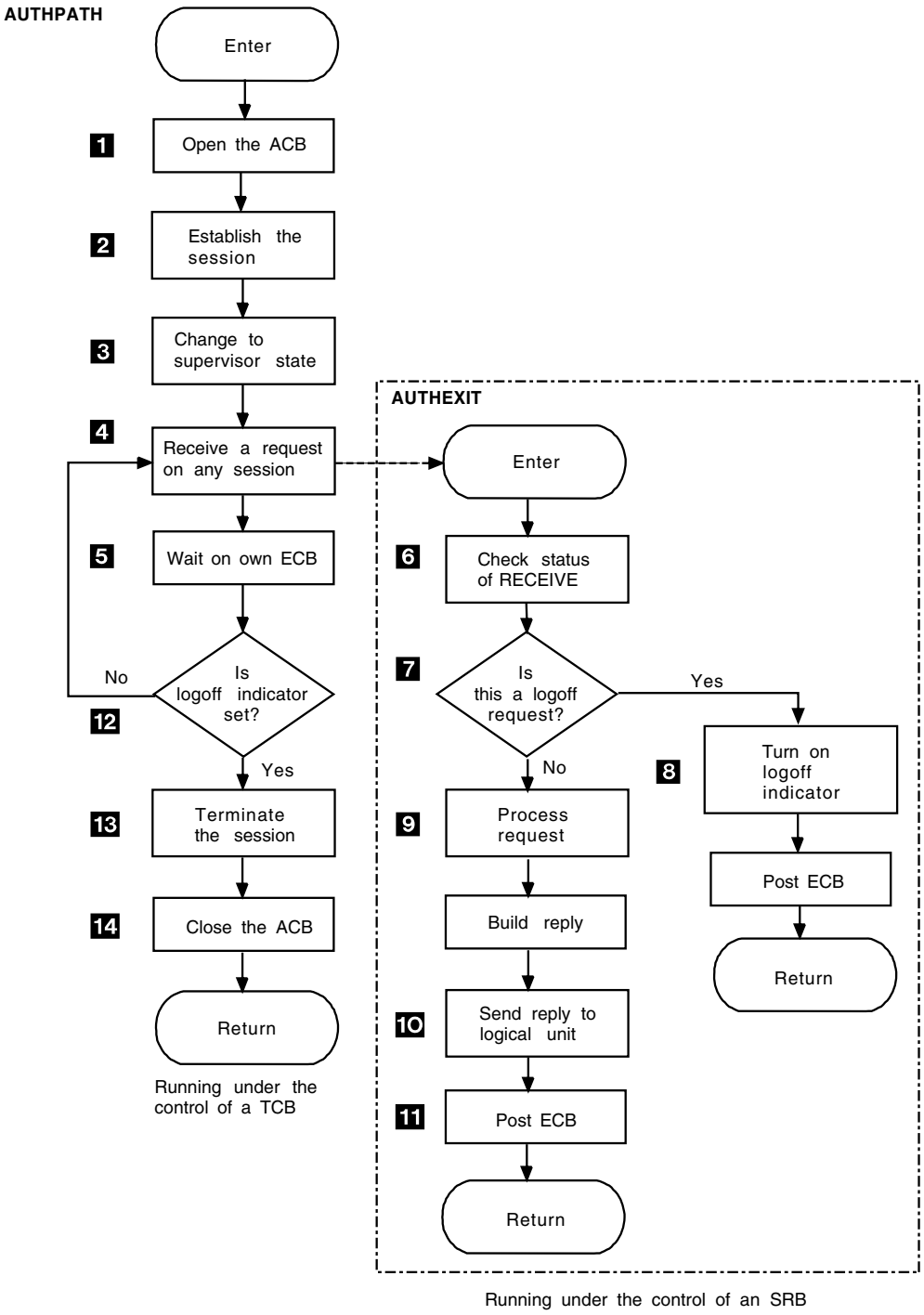


Figure 78. Example of the use of authorized path

Figure 78 illustrates the basic logic for using authorized path when running under a TCB and under an SRB. The program in Figure 78 is highly simplified. The program establishes and handles input from only one session; an actual program would establish and handle input from many sessions. In addition, the logic associated with input/output requests would be more complex in an actual program. The following notes are related to the numbers in Figure 78.

1 The application program begins processing as a task running under the

control of a TCB. As part of normal VTAM processing, it issues an OPEN macroinstruction to open an ACB. The OPEN might look like this:

```
OPEN  AUTHACB
```

In this example, AUTHACB contains:

```
AUTHACB  ACB   AM=VTAM,APPLID=APPL5ID,PASSWD=APPL5ID
```

2 The application program issues an OPNDST macroinstruction to establish a session with a logical unit. The OPNDST might be coded:

```
OPNDST RPL=AUTHRPL,OPTCD=SYN
```

The RPL, named AUTHRPL, contains the rest of the information needed for OPNDST.

3 The application program uses the MODESET macroinstruction to change into supervisor state. This is coded:

```
MODESET MODE=SUP
```

4 The RECEIVE macroinstruction conforms to the coding rules for authorized path running under the control of a TCB. The BRANCH=YES operand is specified. The RECEIVE macroinstruction might be coded:

```
RECEIVE RPL=AUTHRPL,RTYPE=DFSYN,AREA=INPUT00,              C
        AREALEN=100,OPTCD=(ASY,ANY,CS),EXIT=AUTHEXIT,      C
        BRANCH=YES
```

It is known that a request received on the session never exceeds 100 bytes.

5 Because the RECEIVE was specified as an asynchronous operation (OPTCD=ASY), the mainline program, AUTHPATH, can continue execution until an input request on the session completes the receive-any operation. In a more elaborate program, meaningful processing could be done here. But in AUTHPATH, the program immediately enters a wait state, waiting on its own ECB.

6 When a request is received on the session, control goes to the RPL exit routine named AUTHEXIT. This exit routine runs under the control of an SRB. VTAM processing is completed with the scheduling of the exit routine; VTAM itself is not suspended. For example, VTAM could immediately schedule another RPL exit under an SRB.

The CHECK macroinstruction frees the RPL for reuse and causes entry to a LERAD or SYNAD exit routine if necessary. The CHECK macroinstruction is coded:

```
CHECK RPL=AUTHRPL
```

7 The exit routine then tests the input request to see if it is a logoff request (a request in a special format that indicates the logical unit wants to end the session with the program AUTHPATH).

8 If the request is a logoff request, the exit routine turns on a logoff indicator, posts the ECB (as in step **11**), and exits, thus returning control to the mainline program, AUTHPATH.

9 If the request is not a logoff request, the exit routine analyzes the request and builds a reply.

10 The exit routine is running under the control of an SRB because it is an RPL exit routine entered from a macroinstruction using authorized path. The SEND macroinstruction, therefore, automatically uses authorized path. The SEND looks like this:

```
SEND  RPL=AUTHRPL,OPTCD=(SYN,CA),CONTROL=DATA,             C
      STYPE=REQ,RTYPE=DFSYN,RECLEN=95,AREA=OUTPUT00,       C
      POST=SCHED,RESPOND=(NEX,NFME,NRRN)
```

The macroinstruction specifies that the SEND operation is to be performed synchronously (SYN in OPTCD), meaning that the exit routine surrenders control until the SEND operation is scheduled. The macroinstruction also specifies that no response is returned, which assumes that failure of the request to arrive is detected by analyzing the next request entered by the terminal operator.

11 After the SEND operation has been scheduled, the exit routine posts the ECB on which the mainline program (AUTHPATH) has been waiting. This must be done by a branch entry to the supervisor POST routine, because SVCs cannot be issued in SRB mode. The exit routine then exits and thus returns control to AUTHPATH.

12 Because the ECB has been posted, the wait at **5** is satisfied and AUTHPATH continues execution. It tests to determine whether the logoff indicator has been set. If the indicator has not been set, it returns to **4** to reissue the RECEIVE macroinstruction. Thus, execution continues using steps **4** through **12** for as long as input requests other than logoff are received on the session.

When the logoff indicator has been set (indicating that the request received from the logical unit was a logoff request), execution continues at **13**.

13 The program terminates the session by using the CLSDST macroinstruction. The CLSDST might be coded:

```
CLSDST RPL=AUTHRPL,BRANCH=NO,OPTCD=SYN
```

14 The CLOSE macroinstruction closes the ACB.

If desired, both the OPNDST and CLSDST macroinstructions could have been coded to use authorized path. This could have been done by interchanging steps **2** and **3**, and by coding BRANCH=YES on the RPL macroinstruction.

Authorized asynchronous exit routines

For authorized application programs, VTAM optionally gives control to asynchronous exit routines in supervisor state. The exit routines are branch entered from VTAM instead of being entered through the SYNCH macroinstruction. This allows for more efficient operation than if the exit routines were entered normally. The routines are entered in the storage protection key of the authorized program. This type of entry is provided only if SRBEXIT=YES is specified on the APPL definition statement. See “Assigning operating system authorization” on page 296 for the definition of an authorized VTAM application program.

Execution of exit routines

EXLST and RPL exit routines might execute in either SRB mode or TCB mode. See “Task association” on page 310 for information related to error handling.

EXLST exit routines other than LERAD and SYNAD

The following EXLST exit routines can be specified to run in either SRB mode or TCB mode:

- DFASY
- LOGON
- LOSTERM
- NSEXIT
- RELREQ
- RESP
- SCIP
- TPEND.

Selection of the mode of execution is made by coding the ACB macroinstruction PARMS= keyword operand value, or the APPL definition statement operand SRBEXIT=YES or **NO**. In either case, the exit routine must return to VTAM upon completion.

When SRBEXIT=YES is specified, SRB processing is used for the EXLST exit routines listed in the preceding section. The exit routines are branch entered in SRB mode, supervisor state, and key 0. The application program must be authorized in order to open an ACB that has a corresponding APPL definition statement that specifies SRBEXIT=YES; otherwise, the OPEN request is rejected with an OPEN error field value of 244 (hex F4), "application program not authorized." See "Assigning operating system authorization" on page 296 for further information about authorized paths.

If SRBEXIT=NO is specified, or if SRBEXIT is not specified on either the ACB macroinstruction or the APPL definition statement, the exit routines in the preceding list are entered in TCB mode, in problem state, using the key associated with the TCB.

If an exit routine runs in SRB mode, the program environment that existed when the exit routine was entered must be maintained when the exit routine returns to VTAM. This includes returning under the same SRB used for entry, running in supervisor state, with key 0, and with the same control registers and FRR stack. If changes occurred in the exit routine, the application program must restore these variables to their original state before returning to VTAM. Also, any system locks set by the exit routine must be released before returning to VTAM.

Note: If multiple applications open an ACB under one task, the SRBEXIT operand is recognized for the first application opening the ACB. Therefore, the first open ACB determines the exit routines execute in SRB or TCB mode. The SRBEXIT operand is ignored for all subsequent applications opening the ACBs under the same task. If SRBEXIT=YES is specified on the first application opening the ACB, the subsequent applications opening ACBs under the same task must be authorized.

LERAD and SYNAD exit routines

LERAD and SYNAD operate in the same mode (TCB or SRB), state (problem or supervisor), and key as the part of the program that issued the RPL-based or CHECK macroinstruction whose issuance caused the LERAD or SYNAD exit routine to be invoked. Thus, LERAD and SYNAD operate essentially as extensions to that part of the program, and are subject to exactly the same interrupt conditions and other operating system rules as that part of the program.

LERAD and SYNAD need not directly return to VTAM. Instead, they can branch to another part of the application program, still operating as an extension to the

original part of the program from which they were invoked. If they do return to VTAM, VTAM then gives control to the next sequential instruction after the original RPL-based or CHECK macroinstruction.

RPL exit routines

The TCB or SRB mode of execution of an RPL exit routine is determined primarily by whether the associated RPL-based request uses authorized path. See "Authorized path" on page 300, for further information on authorized paths. The SRBEXIT parameter on the APPL definition statement determines the execution mode only if authorized path is not used.

If authorized path is used for an RPL-based request, the RPL exit routine always runs under an SRB, independently of the specification of the SRBEXIT parameter. The exit routine need not return to VTAM.

If authorized path is not used for an RPL-based request, SRB mode is used for the RPL exit routine only if SRBEXIT=YES: TCB mode is used if SRBEXIT=NO. In either case, the exit routine must return to VTAM upon completion. If SRBEXIT=YES is specified, the application program must be authorized as described in "EXLST exit routines other than LERAD and SYNAD" on page 307. The program environment must be maintained for return to VTAM.

In all cases, if SRB mode is used for an RPL exit routine, the exit routine is entered in supervisor state with key 0. In TCB mode, problem state and the key associated with the TCB are used.

Serialization of execution

In "Normal operating system environment for a VTAM application program" on page 30, a simple environment is described as one in which asynchronous exit routines and the mainline program run serially; that is, only one asynchronous exit routine runs at one time and the mainline program cannot run when an asynchronous exit routine has been invoked. The operating system facilities of multitasking (and SRB mode) allow concurrent execution of various parts of the same application program. This section describes the conditions under which several parts of an application program can run concurrently, and the conditions under which serial execution must take place. If several parts of an application program can run concurrently, the application program design must ensure that resources (for example, control blocks and save areas) are properly handled.

When authorized path is used for an RPL-based macroinstruction and an RPL exit routine is specified, that exit routine runs under an SRB. Also, the non-exit parts of a z/OS application program can be written to run under SRBs. Any number of these exit and non-exit SRBs can run concurrently. In all the operating systems, an application program can be written to use multitasking, and multiple tasks can run concurrently. Also, SRB parts of a z/OS application program can run concurrently with task parts of the same program.

All asynchronous exit routines, except authorized path RPL exit routines, are dispatched through a user exit queue.

Multiple user exit queues can exist, one for the task that opens the ACB (the ACB task) and one for each task that issues any RPL-based macroinstruction other than a communication macroinstruction. If multiple user exit queues exist, then, potentially, multiple asynchronous exit routines can run concurrently, one from

each queue. Only a single asynchronous exit routine can be run at one time from each user exit queue; that exit routine must return to VTAM before the next asynchronous exit routine from that user exit queue can be given control. (TPEND with reason code 8 is an exception because it is always immediately invoked.)

The particular user exit queue used for a given asynchronous (non-authorized path) exit routine is determined by the task association of that exit routine. Task association is also used for error isolation and is described in "Task association." When an asynchronous exit routine is dispatched from a task's user exit queue, the task is normally used to process the exit routine, thus suspending any part of the mainline program associated with that task.

However, if SRB mode is used for the exit routine, the exit routine and task can run concurrently. For further information on the SRB and TCB modes, see "Execution of exit routines" on page 307.

If multiple VTAM application programs use the same task, they all share the same user exit queue for that task; consequently, it is important that each exit routine using the user exit queue return to VTAM, because it is possible for an application program's exit routine to hold up the queue and prevent scheduling of exit routines for another application program.

The LERAD and SYNAD exit routines are not serialized directly and are not associated with any user exit queue. They run as extensions of the part of the program that issued the RPL-based or CHECK macroinstruction that caused them to be invoked. Thus, they are serialized only to the extent that the part of the program from which they were invoked is serialized.

Task association

When an application program issues a VTAM macroinstruction or when an application program exit routine is invoked by VTAM, that particular part of the program is explicitly or implicitly associated with an operating system task. A session can also be associated with a task; the task associated with a session is either the task associated with the OPNDST or OPNSEC macroinstruction used to establish the session or, for LU 6.2 applications, the task that opened the ACB. Task association is used to isolate errors to a particular task, as described in "Isolation of errors" on page 322. It is also used to determine which user exit queue is used to dispatch certain asynchronous exit routines; user exit queues are discussed in "Serialization of execution" on page 309.

The associated task can be:

- The ACB task—the task that opened the application program's ACB
- The issuing task—the task associated with the macroinstruction being issued
- The session task—the task associated with the OPNDST or OPNSEC macroinstruction that originally established the session for which another macroinstruction is now being issued. A VTAM LU 6.2 application's session task is associated with the task that opens the ACB.

Exit routine task association

The associated task is the ACB task when the following exit routines are dispatched:

- ACB exit routines–NSEXIT, LOGON, LOSTERM, RELREQ, TPEND

- SCIP exit routine (either ACB or NIB specified) for BIND and UNBIND processing
- ACB-specified session exit routines–DFASY, RESP, SCIP
- NIB-specified session exit routines if authorized path OPNDST or OPNSEC was not used to establish the session–DFASY, RESP, SCIP
- RPL exit routine for an RPL-based macroinstruction that does not use authorized path (additionally for communication macroinstructions, an authorized path OPNDST or OPNSEC macroinstruction was not used to establish the session).

The associated task is the session task when the following exit routines are dispatched:

- NIB-specified session exit routines if authorized path OPNDST or OPNSEC was used to establish the session–DFASY, RESP, SCIP (for other than BIND and UNBIND processing)
- RPL exit routine for an RPL-based communication macroinstruction if authorized path OPNDST or OPNSEC was used to establish the session
- RPL exit routine for an RPL-based communication macroinstruction that does use authorized path.

The associated task is the issuing task when the following exit routine is dispatched:

- RPL exit routine for an RPL-based non-communication macroinstruction that does use authorized path.

The associated task for a LERAD or SYNAD exit routine is the task associated with the corresponding RPL-based or CHECK macroinstruction.

The associated task for a RECEIVE OPTCD=ANY RPL exit routine can be the ACB task or the session task, depending on when the exit routine is scheduled. For LERAD or SYNAD exit routines invoked because of RECEIVE OPTCD=ANY, the associated task can be the issuing task, the ACB task, or the session task.

Macroinstruction task association

OPEN and CLOSE macroinstructions are always associated with the issuing task. RPL-based macroinstructions issued in TCB mode are usually associated with the issuing task; however, for communication macroinstructions, VTAM does most of its processing under the session task and so switches the task association to that task. (For RECEIVE OPTCD=ANY, intermediate processing is also done under the ACB task, which thereby becomes the associated task for a certain period of time.)

SRB requirements

RPL-based macroinstructions issued in SRB mode must specify their task association in the SRB by using the task identification and address space identification fields, with the address space being that of the issuing macroinstruction. Failure to do this for non-communication RPL-based macroinstructions results in an error with (RTNCD,FDB2)=(X'14',X'55'). (For compatibility with previous releases of VTAM, if no task association is specified in the SRB used by SESSIONC in the ACB-address space when sending a request-rejected response to BIND, the ACB task is assumed by default.) For communication macroinstructions (other than RECEIVE OPTCD=ANY), the session task association must be specified in the SRB; for RECEIVE OPTCD=ANY, the

issuing task association must be specified. Failure to specify the correct task association for communication macroinstructions produces unpredictable results.

Multiple address spaces

The multiple-address-space facility under z/OS allows VTAM RPL-based macroinstructions that are issued in different address spaces to reference one ACB. The parts of the application program executing in the address spaces selected are logically grouped into a single logical unit associated with the ACB. Thus, the application program can be thought of as spanning multiple address spaces. The multiple-address-space facility can provide increased flexibility and error isolation for application programs. Improved operation can be achieved by running one session for each address space to maximize isolation or protection, or by grouping a number of sessions in one address space, based on the installation's requirements.

See “Isolation of errors” on page 322 for further information about error isolation.

When using the multiple-address-space facility, sessions associated with an ACB can reside in any available address space in addition to the address space used for opening the ACB. When not using the multiple-address-space facility, all sessions associated with a particular ACB must reside in the address space used for opening the ACB.

Types of address spaces

The three types of address spaces are the ACB address space, the associated address space, and the session address space. These are defined in the following sections. Examples of multiple-address-space assignments using one ACB address space, and others using more than one ACB address space are shown in Figure 79 on page 313 and Figure 80 on page 314. Figure 81 on page 315 summarizes the use of VTAM macroinstructions within multiple address spaces.

ACB address space

An ACB address space is an address space in which an ACB is opened. The ACB must reside in common storage if multiple address spaces are to be used for the application program. An ACB address space has the following characteristics:

- It can use the full set of VTAM macroinstructions.
- It can have more than one ACB.
- It can issue VTAM macroinstructions in TCB mode (authorized path or non-authorized path), or in SRB mode.
- It can be an associated address space (defined in the following) with respect to another ACB.
- It can be a session address space (defined in the following).
- It is the address space where certain EXLST exits are executed.
- It is the only address space in which the ACB can be closed.

If the ACB address space terminates or the ACB is closed, all sessions associated with the ACB are terminated, no matter which address space each session uses.

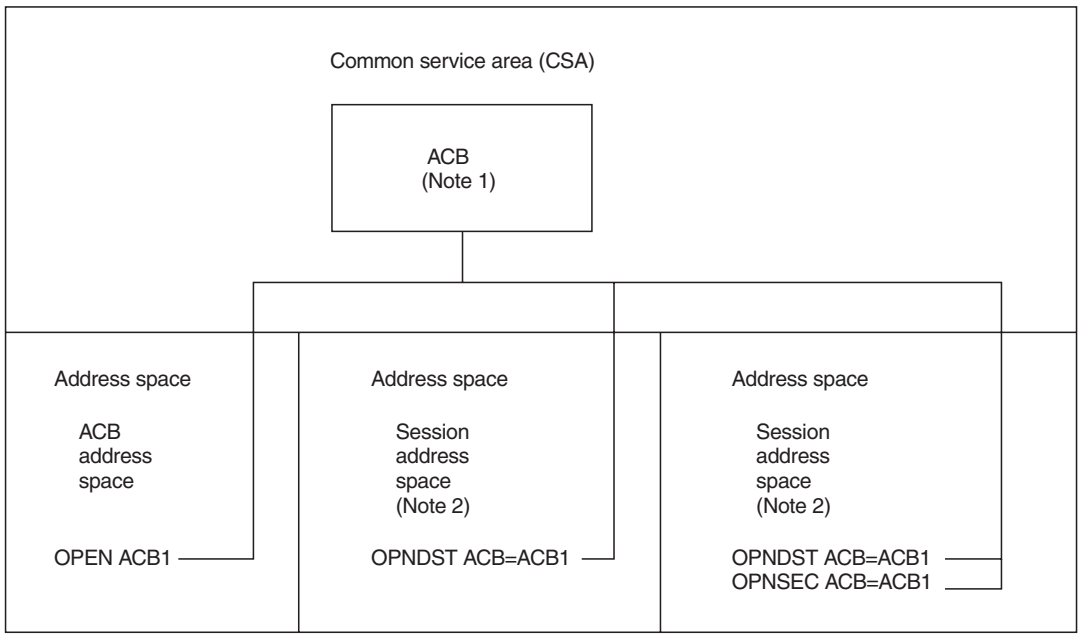


Figure 79. Example of a multiple-address-space configuration with one multiple-address-space ACB

Note:

1. The ACB must reside in commonly addressable storage if VTAM requests specify that the ACBs are issued in address spaces other than that which issued the OPEN ACB (that is, if multiple address spaces are used for the application program).
2. One or more sessions can be associated with each session address space.

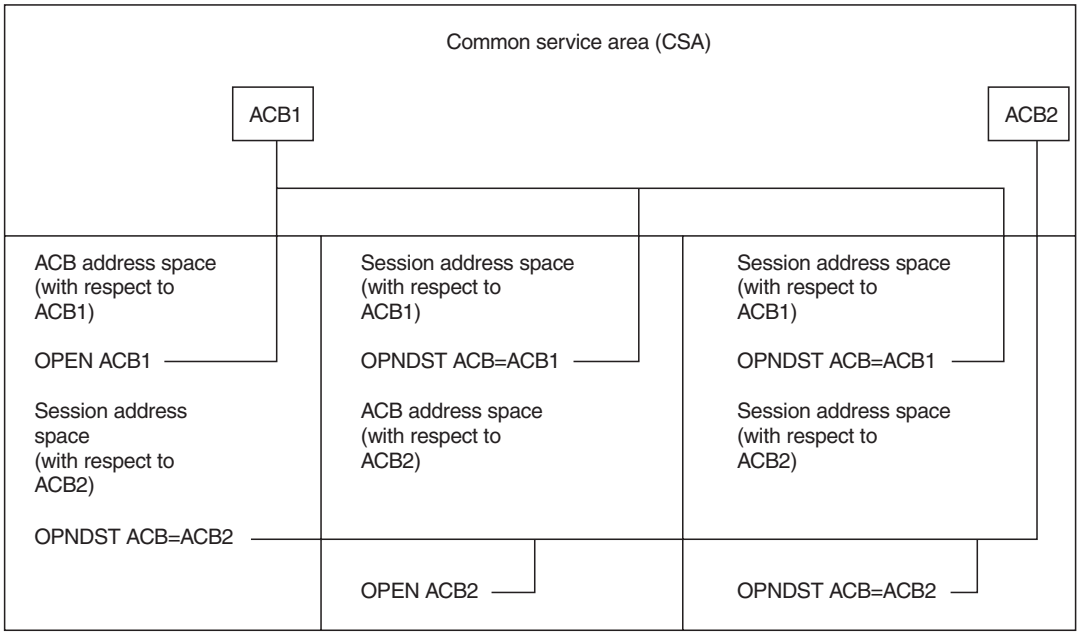


Figure 80. Example of a multiple-address-space configuration with more than one multiple-address-space ACB

Declarative Macroinstructions (must be in the address space of any macroinstructions that reference the control blocks)

ACB EXLST RPL NTB	These build control blocks during program assembly. (DSECT-creating macroinstructions are available for these and other data areas. The ISTGLBAL macroinstruction is also provided to set macro global variable at assembly time.)

Manipulative Macroinstructions (must be issued in the address space of the control blocks that they reference)

GENCB MODCB SHOWCB TESTCB	These build and manipulate control blocks during program execution.

ACB-Based Macroinstructions (can be issued from any address space, except as noted)

OPEN CLOSE	These open and close the application program's ACB. CLOSE must be issued in the address space from which OPEN was issued.

RPL-Based Macroinstructions (can be issued from any address space, except as noted))

Session-Establishment Macroinstructions — These request session establishment, data transfer, and program operator control. They all use an RPL and, with the exception of CHECK, permit RPL modifications to be specified in the macroinstruction itself.

When the Application Program acts as a PLU

OPNDST CLSDST SIMLOGON

When the Application Program acts as a SLU

REQSESS OPNSEC TERMSESS SESSIONC (for negative response to BIND)

Communication Macroinstructions (must be in the session address space, except as noted)

SEND RECEIVE (1) RESETSR SESSIONC (for other than negative response to BIND)

Macroinstructions that assist in Session Establishment of Communication

CHANGE CHECK (1,2) EXECRPL (3) INQUIRE INTRPRET INTRPRET SETLOGON

Program Operator Macroinstructions

SENDCMD RCVCMD

Note:

1. See "Address space used for exit routine execution" on page 316 for RECEIVE OPTCD=ANY considerations.
2. Except for RECEIVE OPTCD=ANY, CHECK must be issued in the address space used to issue the RPL-based macroinstruction.
3. Follows the same rules as the RPL-based macroinstruction it replaces.

Figure 81. Categories of VTAM macroinstructions versus the multiple-address-space functions

Associated address space

An address space that issues VTAM macroinstructions specifying an ACB that was opened in another address space is an ***associated address space*** with respect to the ACB. An associated address space can reference only an ACB that resides in common storage. An associated address space has the following characteristics:

- It can be associated with more than one ACB.
- For an ACB in another address space, it can issue only authorized path VTAM macroinstructions using either SRB mode or authorized path TCB mode.
- It cannot issue a CLOSE macroinstruction for an ACB that was opened in another address space.
- It can OPEN an ACB and thus also be an ACB address space.

Termination of an ACB address space also results in removal of the VTAM structures through which other address spaces are associated with the ACB address space. This includes termination of any sessions or RPL-based operations in the associated address spaces that relate to the ACB.

Session address space

A session address space is an address space used to issue VTAM macroinstructions that establish sessions (OPNDST or OPNSEC). A session address space is either an ACB address space or an associated address space. The RECEIVE OPTCD=SPEC, RESETSR, SESSIONC (except when used to send a request rejected response to BIND), and SEND macroinstructions can be issued only in a session address space. If one of these macroinstructions is issued in an address space other than the address space in which the corresponding session was established, the macroinstruction is posted complete with (RTNCD,FDB2)=(X'14',X'24'), indicating a logic error and an address space ID (ASID) mismatch.

Rules for coding macroinstructions and exit routines

The following rules apply with a multiple-address-space application program:

- The ACB must reside in the common service area (CSA). The APPLID and PASSWD data areas referenced by the ACB must reside in the ACB address space. Similarly, for a communication network management application program, the NIB and all data areas referenced by it must be in the ACB address space.
- The EXLST referenced by the ACB or NIB must reside in the CSA.
- The RPL and data areas referenced by the RPL must be addressable from the address space issuing an RPL-based macroinstruction. For RECEIVE OPTCD=ANY, the RPL and data areas must also be addressable from the ACB and session address spaces.
- It is recommended that any exit routine that might run in more than one address space be reentrant and placed in the link pack area (LPA).

Address space used for exit routine execution

With a few exceptions, the address space in which an asynchronous exit routine runs is the ACB address space. The exceptions are:

- If a session is established by using authorized path for an OPNDST or OPNSEC macroinstruction, then any RESP, DFASY, or SCIP exit routine specified in the NIB EXLST for that session runs in the session address space. (BIND and UNBIND processing by a SCIP exit routine is always done in the ACB address space.)

- An RPL exit routine runs in the issuing macroinstruction's address space. CHECK must be issued in that address space. (RECEIVE OPTCD=ANY is handled as described next.)

BIND is always processed by the SCIP exit routine specified in the ACB EXLST; the exit routine runs in the ACB address space. UNBIND processing by an SCIP exit routine (either ACB or NIB EXLST specified) is also done in the ACB address space; if an NIB SCIP exit routine was specified for the session, that exit routine is used, but it runs in the ACB address space and thus might require that the NIB SCIP exit routine be in the operating system's LPA.

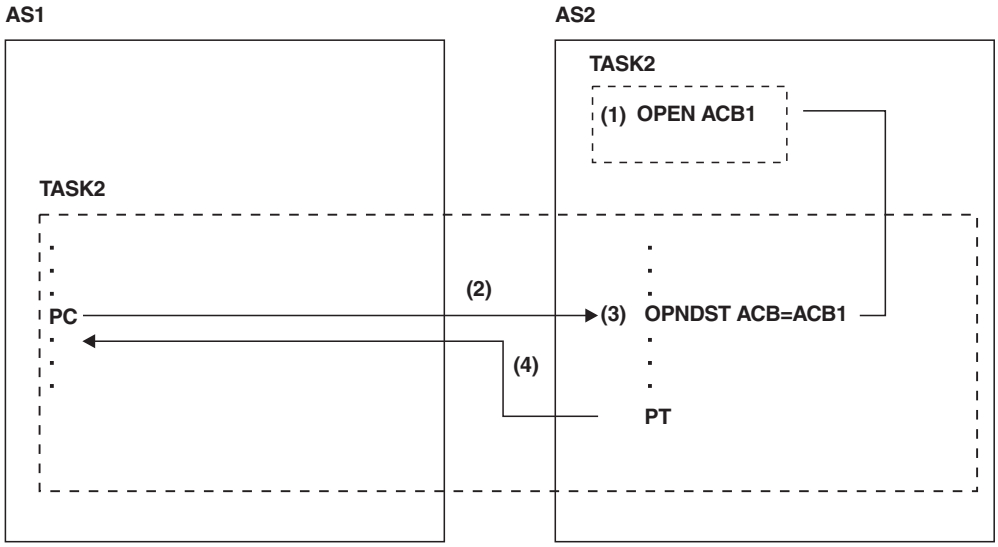
RECEIVE OPTCD=ANY processing occurs partially in the issuing macroinstruction's address space, partially in the ACB address space, and partially in the address space of the session for which the RECEIVE OPTCD=ANY operation is posted complete (that is, the address space of the session for which input is received). An RPL exit routine can be invoked in either the ACB or session address space. Thus, the application program might need to be written to have such an RPL exit routine in LPA, or to issue RECEIVE OPTCD=ANY only from the ACB address space, and to restrict the sessions that can run in continue-any mode to be those for which the session address space is the ACB address space.

LERAD and SYNAD exit routines are normally executed in the address space used by the CHECK or RPL-based macroinstruction for which the LERAD or SYNAD exit routine was invoked. LERAD and SYNAD exit routines for RECEIVE OPTCD=ANY are exceptions in that they can also execute in the ACB address space or the session address space, as well as in the issuing macroinstruction's address space. The program design considerations described for RPL EXIT routines in the preceding paragraph apply also to LERAD and SYNAD exit routines.

Cross-memory application program interface (API) support

The cross-memory API support enables application programs to issue VTAM API macroinstructions from an address space other than the application's home address space. A program that is executing in one address space uses the PROGRAM CALL (PC) instruction to branch to an entry point in another address space.

When the program is first dispatched, the home address space and the primary address space are the same. After the PC instruction occurs, the two address spaces are different, though linked. In cross-memory mode, z/OS treats the primary address space as a session address space. See “Session address space” on page 316 for more information about the session address space. Figure 82 on page 318 shows an example of an application program operating in cross-memory mode.



(1) A TCB-mode task (TASK1) opens the VTAM application program's ACB (ACB1)

(2) A program executing in AS1 issues a PC instruction to request a service from a function located in AS2. TASK2 now enters cross-memory mode with AS2 as the primary address space and AS1 as the home and secondary address space.

(3) The function in AS2 issues the VTAM macroinstruction which references the ACB opened by TASK1. All data passed in the VTAM macroinstruction must be addressable in the primary address space (AS2 in this example).

(4) The function in AS2 issues the PT (program transfer) to return control to the program in AS1.

Figure 82. Example of an application program operating in cross-memory mode

The PROGRAM CALL dispatchable unit of work must exist as a TCB or SRB. Refer to the z/OS MVS Programming: Extended Addressability Guide for more information about z/OS capabilities and restrictions.

VTAM places the following constraints on the application that is in cross-memory mode:

- OPEN and CLOSE macroinstructions must be issued in non-cross-memory mode by mainline processing under TCB control.
- OPEN and CLOSE macroinstructions must be issued in the address space that becomes the primary address space during a cross-memory VTAM API request.
- The application must be in SRB mode or authorized TCB mode.
- The application program must be in supervisor state, enabled, unlocked, and in primary addressing mode.
- Specific data passed to VTAM must be addressable in the primary address space. This data includes:
  - RPL, including the RPL extension ISTRPLEX
  - RPL6
  - ACB
  - NIB
  - User data to be passed to VTAM
  - RPL exit or ECB
  - ACB EXLST and NIB EXLST.

31-bit addressing

Instruction and data addresses can be treated as 24- or 31-bit values. Addressing mode describes the size of addresses being used. Application programs executing while the system is in 24-bit addressing mode can address up to 16 megabytes of virtual storage; programs executing in 31-bit mode can address up to 2 gigabytes (approximately 2 billion bytes) of virtual storage.

Every application program is assigned two new attributes, an AMODE (addressing mode) and an RMODE (residence mode). (Existing application programs are assigned default AMODE and RMODE attributes of 24.) AMODE specifies the addressing mode in which the program is designed to receive control. RMODE indicates where in virtual storage the program can reside.

Generally, an application program is designed to execute in the same addressing mode in which it receives control. However, a program can switch modes and can have different AMODE attributes for different entry points within a load module.

The valid AMODE and RMODE specifications are shown in Table 56.

Table 56. Valid AMODE and RMODE specifications

AMODE and RMODE	AMODE and RMODE specifications
AMODE=24	Specifies 24-bit addressing mode
AMODE=31	Specifies 31-bit addressing mode
AMODE=ANY	Specifies either 24- or 31-bit addressing mode
RMODE=24	Indicates that the module resides in 24-bit mode. The RMODE=24 specification can be used for 31-bit programs that have 24-bit dependencies.
RMODE=ANY	Indicates that the module can reside anywhere in virtual storage

If the application program does not specify AMODE and RMODE, the operating system assigns the following defaults: AMODE=24 and RMODE=24. To override the defaults, specify AMODE or RMODE or specify AMODE and RMODE on one or more of the following:

- AMODE and RMODE statements within the assembler source code for an application program. For example:

```
XYZ      CSECT
         AMODE xxx
         RMODE xxx
```

- The EXEC statement of a link-edit step:

```
//LKED    EXEC PGM=HEWLH096,PARM='AMODE=xxx,RMODE=xxx,...'
```

- The linkage editor MODE control statement (one per load module):

```
MODE     AMODE(xxx),RMODE (xxx)
```

- The LINK or LOADGO TSO commands:

```
LINK    AMODE(xxx),RMODE (xxx)
LOADGO  AMODE(xxx),RMODE (xxx)
```

In the preceding examples, *xxx* is one of the valid specifications for AMODE and RMODE listed in Table 56.

Opening by the Application Program

VTAM supports an application program in either 24- or 31-bit addressing mode, depending upon the addressing mode of the application program at the time the ACB is opened. The OPEN macroinstruction is used to identify the application program to VTAM.

Control block fields referenced by the OPEN and CLOSE macroinstruction can reside in either 24-bit or 31-bit storage. Use of 31-bit storage should be consistent with the addressing mode of the application program. MODE=31 must be coded on the OPEN and CLOSE macroinstructions if the ACB or other control blocks reside in 31-bit storage.

For an application program defined with an RMODE attribute of 24, the OPEN macroinstruction creates a parameter list in 24-bit storage; for an application program defined with an RMODE attribute of ANY or 31, the application program must build a parameter list in 24-bit storage and then issue the execute form of the OPEN macroinstruction. See “OPEN—Open one or more ACBs” on page 444, for a detailed description of the OPEN macroinstruction.

Specifying macroinstructions

Special considerations apply when issuing the different types of macroinstructions. The following sections explain the considerations for the declarative, manipulative, ACB-based, and RPL-based macroinstructions.

Declarative macroinstructions

The declarative macroinstructions (ACB, EXLST, NIB, and RPL) build control blocks during program assembly.

ACB macroinstruction:

Generates an access method control block (ACB) that conforms to the requirements for full 31-bit addressing.

EXLST, NIB, and RPL macroinstructions:

Generate control blocks (EXLST, NIB, and RPL control blocks, respectively) that conform to the requirements for full 31-bit addressing.

ACB-based macroinstructions

The ACB-based macroinstructions (OPEN and CLOSE) open and close the application program's ACB. See “Opening by the Application Program” and “Closing by the Application Program” on page 321 for considerations when issuing these macroinstructions. Descriptions of the OPEN and CLOSE macroinstructions are located in Chapter 13, “Conventions and descriptions of VTAM macroinstructions,” on page 371.

RPL-based macroinstructions

All RPL-based macroinstructions except the CHECK macroinstruction execute in the addressing mode of the application program that issued the macroinstruction, and return control to the application program in the same mode.

CHECK macroinstruction:

Must be issued in an addressing mode consistent with the addressing mode of the application program at the time the original request was made. For example, if an asynchronous RECEIVE with EXIT=RECVEXIT was issued in 24-bit addressing mode, the code in procedure RECVEXIT must be in 24-bit addressable storage. Issuing the CHECK macroinstruction in 24-bit addressing mode for a request that was issued in 31-bit addressing mode can have unpredictable results. In addition, all control blocks used by the CHECK macroinstruction must reside in storage consistent with the application program's addressing mode.

Executing exit routines

VTAM allows the use of exit routines so a VTAM application program can gain control to handle a specific event. When the event occurs, VTAM gives the exit routine control as soon as possible.

Exit routines can receive control in either 24-bit or 31-bit addressing mode, depending on the type of exit routine. For some exit routines, the addressing mode is determined when the application issues a request. For other exit routines, the addressing mode is determined when the application first opens its ACB. Table 57 shows how the addressing mode is determined for each exit routine.

Table 57. Addressing mode for each kind of exit routine

Exit routine	Addressing mode used
RPL	Mode that the application was using when it issued the RPL request
LERAD and SYNAD for synchronous operations	Mode that the application was using when it issued the synchronous request
LERAD and SYNAD for asynchronous operations	Mode that the application was using when it issued the CHECK macroinstruction
ACB exit routines and session-level exit routines identified by the EXLST macroinstruction	Mode that the application was using when it opened its ACB

Closing by the Application Program

About this task

An application program issues a CLOSE macroinstruction to disassociate itself from VTAM. The CLOSE macroinstruction specifies the same ACB that was originally used by the OPEN macroinstruction to identify the application program to VTAM.

Control block fields referenced by the OPEN and CLOSE macroinstruction can reside in either 24-bit or 31-bit storage. Use of 31-bit storage should be consistent with the addressing mode of the application program. MODE=31 must be coded on the OPEN and CLOSE macroinstructions if the ACB or other control blocks reside in 31-bit storage.

For an application program defined with an RMODE attribute of 24, the CLOSE macroinstruction automatically creates a parameter list in 24-bit storage; for an application program defined with an RMODE attribute of ANY or 31, the application program must build a parameter list in 24-bit storage and then issue

the execute form of the CLOSE macroinstruction. See “CLOSE—Close one or more ACBs” on page 390, for a detailed description of the CLOSE macroinstruction.

Error handling

If an abnormal termination occurs while VTAM is processing under an application program task, the degree of recovery is operating system dependent.

The abnormal termination might be restricted to affect only a part of the application program, as described in “Isolation of errors.”

When VTAM abnormally terminates, VTAM rejects application program requests, including CLOSE ACB.

If possible, the TPEND exit routine is scheduled with reason code 8.

Isolation of errors

To minimize the disruption caused by an error detected while VTAM is processing, an attempt is made to isolate the error to the failing request, session, task, or application program, in that order. Additionally, if a task or address space is terminated while processing for other than VTAM (for example, if an abend occurs in the data processing part of an application program task), VTAM attempts to isolate the disruption to the task structure or address space involved. Only failure of the ACB task or ACB address space abends the whole application program.

Request level isolation

If VTAM can isolate an error to a specific RPL-based request, that request is posted complete with (RTNCD,FDB2)=(X'10',X'0E'). However, if damage was done to a session while processing the request, the error is considered to be at the session level or task level and is handled as described in the following sections.

For non-communication macroinstructions processed under the VTAM task, certain errors might result only in a VTAM operator message; it might not be possible to post the request complete.

Session level isolation

If VTAM can isolate the error to a single session, an UNBIND request (usually with type code X'0E'— Recoverable LU Failure) is sent to both logical units in that session. The manner in which this session outage notification is presented to the application program depends on whether the application program is the primary or secondary logical unit; whether SCIP, NSEXIT, or LOSTERM exit routines were specified; and whether the SONSCIP parameter was used on the APPL definition statement. See “Session outage notification” on page 110 for a description of these options.

“SCIP exit routine” on page 256 provides a brief description of the potential UNBIND type codes.

VTAM also attempts to post any outstanding requests for the session with (RTNCD,FDB2)=(X'10',X'0E'). Depending on the error that caused the original outage, this posting might not always be possible.

The application program can attempt to reestablish the session after performing any specific user application cleanup procedures that might be necessary.

Task level isolation

If VTAM is unsuccessful in isolating an error to a session, it attempts to isolate the error to a task. The task in error is abended, all sessions associated with it are terminated, and outstanding task-associated requests are posted complete, if possible. The next section describes VTAM procedures if the failing task is the ACB task. See "Task association" on page 310 for a discussion of how sessions and requests are associated with a task.

The task's abend exit routine can attempt to recover the task; however, VTAM independently cleans up its association with the failing task.

VTAM terminates all sessions associated with the failing task. For each session, each logical unit in the session receives session outage notification, either in the form of an UNBIND request or a CLEANUP request. The manner in which this session outage notification is presented to the application program depends on whether the application program is the primary or secondary logical unit; whether SCIP, NSEXIT, or LOSTERM exit routines were specified; and whether the SONSCIP parameter was used on the APPL definition statement. See "Session outage notification" on page 110 where these options are described. If the task in error is the one in which the SCIP, NSEXIT, or LOSTERM exit would run, the exit cannot be scheduled.

Application program (ACB) level isolation

If a task level failure (as described in the previous section) occurs for the task that opened the ACB for the application program (the ACB task), then the application program is disassociated from VTAM.

If multiple address spaces are being used for the application program, VTAM's disassociation from the application program also occurs from all associated address spaces for the application program.

All sessions with the application program are terminated. The other end of each session is notified with either a CLEANUP (from its SSCP) or an UNBIND (from the abending application program). "Session outage notification (SON) codes on UNBIND" on page 93 discusses how such notification is received by a VTAM application program if it is the other end of the session.

The ACB name of the failing application program cannot be reused in another OPEN ACB until VTAM has completed its cleanup for the failing application program.

Note: If an application program has enabled persistence, an application that is capable of persistence can reuse the ACB name. An attempt can be made to open an ACB using the same name; however, the OPEN fails with an error code of 112 (X'70') if VTAM has not finished the cleanup. If VTAM successfully finishes, all resources associated with the previous application program (for example, logical units in session with it) are freed. If a failure occurs during the VTAM cleanup, some resources can remain unavailable. VTAM operator commands (VARY TERM and VARY NET INACT,APPL=) can be used to free these resources. If this is unsuccessful, restart VTAM to free the resources.

Task termination and address space termination

If a VTAM application program task abends while processing for other than VTAM (for example, while doing disk I/O or because the application program itself issued ABEND) and the task does not recover from the abend by a retry, then

VTAM is notified. VTAM then does the task level or application program level (for an ACB task) cleanup processing described in the preceding sections.

If an address space in which a VTAM application program resides is terminated, VTAM is notified of whether the reason for termination was related to VTAM processing. VTAM then cleans up that address space for VTAM processing related to that application program. If the failing address space is an associated address space, the ACB address space and other associated address spaces are not disrupted (see "Multiple address spaces" on page 312).

Functional recovery routines

Functional recovery routines can be provided by the application program, or the application program may opt to use VTAM's recovery routine.

Application program functional recovery routines:

If an SRB-mode application provides FRR coverage before issuing an RPL-based macroinstruction, several factors control whether the recovery environment of the issuer is maintained:

- PARMS=(KEEPFRR=YES|NO) on ACB macroinstruction
- OPTCD=ASY|SYN processing for RPL-based request
- Cross-memory vs. non-cross-memory request mode
- Issuer's SRB kept or exited during VTAM processing
- Whether control will be returned under the same SRB provided by the issuer.

If the application program provides its own FRR coverage, and if these FRRs should be retained through VTAM's processing for requests and subsequent return to the issuer, specify PARMS=(KEEPFRR=YES) on the program's ACB macroinstruction. This causes VTAM to keep any user-provided FRRs in place during VTAM's macroinstruction request processing for:

- All asynchronous (OPTCD=ASY) requests
- All cross-memory synchronous (OPTCD=SYN) requests
- Non-cross-memory synchronous requests when the issuer's SRB is suspended during VTAM processing and resumed at completion to return to the issuer (OPTCD=(SYN,KEEPSRB)).

If the application provides FRR coverage but it is not retained during VTAM processing, the application will have to reestablish its FRR coverage when control returns to the program after the macroinstruction. This is the case if PARMS=(KEEPFRR=NO) is coded on the ACB macroinstruction, or if KEEPFRR= is omitted, or for non-cross-memory synchronous requests that do not use OPTCD=KEEPSRB.

VTAM recovery routine:

When an application program issues an RPL-based macroinstruction while running under an SRB, VTAM subsequently returns control to the application program under an SRB, either at the next sequential instruction or at the entry to the LERAD or SYNAD exit routine.

In this case, a functional recovery routine (FRR) can be optionally supplied by VTAM to gain control if an abnormal termination occurs. Also, the RPL exit routines for macroinstructions using authorized path have optional FRR coverage. In both cases, if VTAMFRR=YES is coded on the APPL definition statement, FRR

coverage is supplied; if VTAMFRR=NO is coded, the FRR stack is purged before the application program is given control in SRB mode.

In all other cases (that is, all asynchronous exit routines other than authorized path RPL exit routines), when VTAM gives control to the application program in SRB mode, FRR coverage is automatically supplied by VTAM.

If the VTAM-supplied FRR gains control during an abnormal termination, it causes the associated task's ESTAE routine to be invoked if an ESTAE routine has been specified. If VTAMFRR=NO is specified, no such notification by an ESTAE takes place.

Chapter 11. Programming for the IBM 3270 Information Display System

This chapter describes VTAM application programming for sessions that use LU type 0 protocols. Other 3270 protocols (for example, for LU types 1, 2, and 3) are not described here; refer to the 3270 manuals for these protocol descriptions.

Before using the information in this chapter, you should be familiar with the 3270 system in the *IBM 3270 Information Display System Description and Programmer's Guide* applicable to your control unit. Some of the SNA protocols mentioned (for example, brackets) are further described in Chapter 6, "Communicating with logical units," on page 151.

Types of 3270 terminals

The following types of 3270 terminals are directly supported by VTAM in its domain:

SNA 3270 terminals:

Local (PU type 2)
3270 terminal attaches to a PU type 2 cluster controller, which in turn attaches to the VTAM host processor by a System/370 channel. Example: 3278 Display Station on 3274-1A Control Unit.

Remote SDLC (PU type 2)
3270 terminal attaches to a PU type 2 cluster controller, which in turn attaches by an SDLC link to a communication controller containing an NCP, or directly to a processor through a communication adapter. Example: 3278 Display Station on 3276 Control Unit.

Remote SDLC (PU type 1)
3270 terminal attaches to a PU type 1 cluster controller, which in turn attaches by an SDLC link to a communication controller containing NCP, or directly to a processor through a communication adapter. Examples: 3288 Line Printer on 3271-12 Control Unit or a 3767 terminal.

Non-SNA 3270 terminals:

Local 3270 terminal attaches to a non-SNA cluster controller, which in turn attaches to the VTAM host processor by a System/370 channel. Example: 3277 Display Station on 3272 Control Unit. A 3270 terminal attached to a 4331 Display/Printer adapter also presents this appearance.

Remote BSC
3270 terminal attaches to a non-SNA cluster controller, which in turn attaches by a BSC link to a communication controller containing NCP, or directly to a processor through a communication adapter. Example: 3286 Printer on 3271-2 Control Unit.

In addition, VTAM supports sessions between application programs in the VTAM host processor and 3270 terminals in another domain in which each cross-domain terminal provides the appearance of one of the types of 3270s listed in the preceding discussion.

© Copyright IBM Corp. 2000, 2013

Characteristics of LU type 0 for 3270 terminals

The SNA protocols and other considerations for PU type 2 3270 terminals are described in the 3270 manuals and are not further described in this chapter.

In general, VTAM makes each non-SNA 3270 terminal appear to a VTAM application program as though it were a PU type 1 3270 terminal. In doing this, VTAM also masks the attachment differences between BSC and channel-attached non-SNA terminals. To the application program, each terminal acts as if it were a device-type LU. The application program does not consider line control protocols because they are completely handled by the network.

Except as noted, the same VTAM facilities are available for these LUs as are available for LUs associated with SNA devices. Thus, for example, a session can initiate with SIMLOGON and establish with OPNDST OPTCD=ACCEPT. Communication then occurs using SEND, RECEIVE, RESETSR, and SESSIONC. The session finally terminates with CLSDST.

In general, the SNA protocols supported for non-SNA 3270 terminals and PU type 1 3270 terminals are more limited than the protocols supported by other LUs. The term LU type 0, used in the 3270 context, denotes this limited set of protocols. Note that LUs associated with PU type 2 3270s support LU types 1, 2, and 3 (described in the component description manual) and do not support LU type 0. The LU type 0 protocols are described here.

Data stream

The data stream (request unit contents) sent to or received from an LU type 0 3270 logical unit is further described in the component description manual. The data stream sent from an application program, using the SEND macroinstruction, consists of a 3270 command code and associated control information, followed optionally by 3270 orders and data. Note that the application program does not specify any line control information such as STX (start of text) and ETX (end of text) and does not specify an ESC (escape character). VTAM supplies any information required to communicate with the particular terminal involved in each session. The data stream received from the logical unit, using the RECEIVE macroinstruction, consists of an AID (attention identifier) byte, usually followed by cursor information, orders, and data. As mentioned for output, no line control information is present in the RUs passed to the application program. Therefore, the basic data stream is the same for all LU type 0 3270s, independent of BSC, SDLC, or channel modes of attachment. Of course, specific device capabilities (such as a selector pen or a magnetic card reader) that each terminal has can make data streams differ from terminal to terminal.

Other information received from a 3270 terminal does not pass directly to the application program. It is, instead, intercepted by the network. It can map into an RU before passing to the application program, or can be used to cause an action, such as the scheduling of the LOSTERM exit routine. Refer to “Exception conditions and sense information” on page 332, for information about how exception requests and negative responses are received by the application program. Refer to “Test request” on page 335, for information about the purpose of the test request within the network.

An EBCDIC character set is normally used for LU type 0. For these terminals, the application program must set the code-selection indicator to

CODESEL=STANDARD for SEND; VTAM sets CODESEL=STANDARD when RECEIVE completes. For each byte of the data stream, the code point (hexadecimal representation) used by VTAM is the one defined in the component description manual for communicating with an EBCDIC BSC 3270 terminal. Note that an ASCII BSC terminal can attach to a communication controller; however, the communication controller must translate the terminal's ASCII input into EBCDIC before sending the data to the host processor and must translate EBCDIC output data from the host processor into ASCII before sending it to the BSC terminal.

Certain PU type 1 3270 terminals support an ASCII character set. For these terminals, CODESEL=ALT is used for SEND and RECEIVE. The communication controller does not translate the data stream for these terminals; ASCII data is passed between the application program and the terminal.

Function management headers are not used by LU type 0, so the application program must set OPTCD=NFMHDR for SEND; VTAM sets OPTCD=NFMHDR when RECEIVE processing is completed.

An RU sent to a 3270 terminal can cause data to be returned to the application program. For example, the RU can contain a 3270 command such as Read Modified. The data returned by the 3270 in reply to such an RU is obtained by the application program by using a RECEIVE macroinstruction. The 3270 can also send data asynchronously without having been sent a request for input. For example, pressing the ENTER key causes the 3270 to send an RU to the application program.

A 0-length RU can be sent to an LU type 0 3270 LU. This is usually done to begin or end a bracket when no data is available to be sent at the same time. SEND RECLEN=0, with BRACKET set to the desired values, accomplishes this function.

An application program cannot send just the Erase/Write command to function as a no-operation at a BSC 3270. When a data stream is sent to a BSC or non-SNA channel-attached 3270, VTAM adds a null Write Control Character (WCC) to the data stream following the command code. Therefore, when the command arrives at the 3270, it is no longer just the command code and it can no longer function as a no-operation.

Data flow control

LU type 0 uses function management profile 2.

Data-flow-control requests

No data-flow-control requests can be used. Thus, only CONTROL=DATA applies for SEND and RECEIVE.

Chaining

Every request must be a single-request chain. Thus, CHAIN=ONLY applies for each SEND and RECEIVE.

Normal-flow Send/Receive mode

Full duplex is used. No change-direction protocol is allowed. Thus, for each RU, the RPL is set to CHNGDIR=(NCMD).

Responses

Requests received from the 3270 LU always specify no response, RESPOND=(NEX,NFME,NRRN). Requests sent to the 3270 LU can specify exception response, RESPOND=(EX,FME,NRRN), or definite response, RESPOND=(NEX,FME,NRRN). However, when sending data to a printer, or when sending a begin-bracket or end-bracket indicator, the application program must ask for a definite response. For all requests sent to the LU, only response type 1 is allowed; thus, NRRN must always be specified. The queued-response indicator is not supported by the PU 3270 PU type 1. You should specify it as RESPOND=(NQRESP) for the application program to work with all LU type 0 3270s.

Brackets

An LU type 0 session can operate either with or without bracket protocols. The choice is specified in the BIND session parameter.

In the following discussion, *BB* denotes that the begin-bracket indicator is set on in a request, and *EB* denotes that the end-bracket indicator is set on in a request.

Note: The conditional end bracket indicator is not supported by LU type 0 3270 terminals.

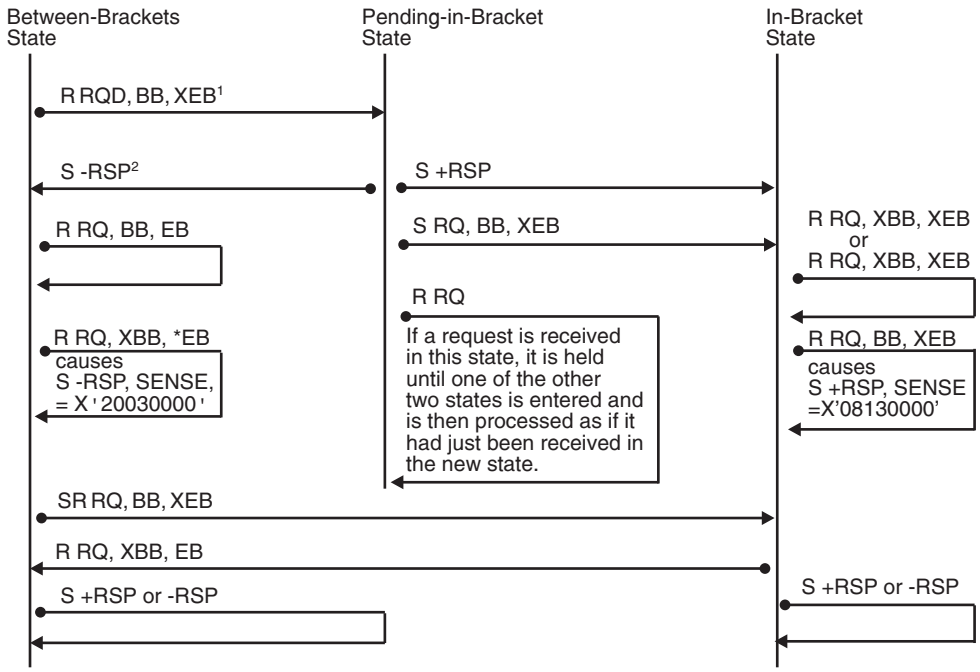
If brackets are used, the bracket state after the BIND request (sent by OPNDST) and after the SNA Clear request (sent by SESSIONC) is the between-brackets state. The between-brackets state also occurs when a bracket is ended by sending a request specifying EB. The first data RU sent when the session is in the between-brackets state must specify BB. The application program cannot send BID to begin a bracket, because no data-flow-control requests are allowed for LU type 0. The 3270 LU is always the first speaker; that is, the LU is always the winner if both the LU and the application program try to begin a bracket at the same time. If the session is in the between-brackets state and the 3270 operator sends data to the application program, the program receives a request specifying BB. If the application program was trying to begin a bracket at the same time, the application program's request is rejected with a negative response (SNA sense code = X'08130000').

After a bracket has been started and until it is ended by the application program, all requests from either the LU or the application program are sent with both the begin-bracket and end-bracket indicators set off. Only the application program can send a request specifying EB. The unconditional bracket termination rule is used; this means that the bracket is ended immediately when a request specifying EB is sent (it is not necessary to receive a positive response to the request).

The bracket can also be ended by sending an SNA Clear request using the SESSIONC macroinstruction. The Clear request is useful to establish synchronization of bracket states between the 3270 LU and the application program. After Clear completes, both ends of the session are in the between-brackets state no matter what state they were in previously.

The application program should not begin or end a bracket with data that contains a 3270 command that causes input to occur (for example, Read Modified).

The bracket state transitions that can occur at the 3270 LU, which is always the SLU, and the bracket-protocol first speaker, are shown in Figure 83 on page 331.



[1] The PLU is required to specify RQD if BB and XEB are indicated. If this rule is disobeyed and the SLU received RQE, BB, XEB when in between-bracket state, it immediately enters in-bracket state. If the SLU then returns an -RSP for the BB request, the SLU will usually return to between-bracket state. See also [2].

[2] An -RSP sent for a BSC 3270 terminal attached to a communication controller can cause entry into in-bracket state instead of between-bracket state; this then causes a subsequent BB request to be rejected with sense=X'08130000'. If this happens, the begin-bracket can be turned off and the request can be resent.

Figure 83. Bracket-state transitions at the 3270 SLU

Sequence numbering

Sequence numbering of normal-flow RUs is done for each direction of flow for an LU type 0 session. For certain types of 3270 LUs, the sequence number wraps to 0 after it reaches 255; for all other LUs, the sequence number wraps to 0 only after it reaches 65 535. The application program must always use the full 2-byte sequence number field for any comparisons (for example, when trying to match a response with a previously sent request) and should not be coded to be sensitive to the point at which the sequence number wraps.

Transmission control

Transmission services profile 2 is used with LU type 0. Pacing is supported from the application program to the 3270 LU, but not from the LU to the application program. The only session-control request that can be used is Clear, which resets bracket and pacing states, purges any outstanding requests and responses, and also sets the session sequence numbers to 0. After SESSIONC CONTROL=CLEAR has been posted complete, either the application program or the 3270 LU can attempt to send data; if the optional bracket protocol is being used, the begin-bracket

indicator must be set on in the first data request sent after Clear. The Start Data Traffic request is not required or allowed. Only a non-negotiable BIND can be specified to start an LU type 0 session. Cryptography is not allowed.

Exception conditions and sense information

The application program can receive exception requests and negative responses from the 3270 LU or from other network components on behalf of the LU. These RUs contain 4 bytes of SNA sense information as described in "SNA sense fields" on page 679. VTAM makes this information available in the SSENSEI, SSENSMI, and USENSEI fields of the RPL. The SSENSMI field is usually set to 0, in contrast to SNA LUs. For some (but not all) of the 3270 devices, the SSENSMI field is not 0 as indicated in Table 58.

The USENSEI field can contain information derived from a 3270 BSC status and sense message, or from the CSW status byte and sense bytes associated with a non-SNA channel-attached 3270 terminal. Such information is sent directly from a PU type 1 3270 terminal. The USENSEI format is the same for all LU type 0 3270 terminals. Associated error recovery procedures are described in detail in the component description manual. The bits in the 2-byte field can be set in combination and should be tested individually. They are defined in Table 59 on page 333.

An exception request is received typically when a stand-alone device end is generated by the 3270 terminal (for example, when a terminal powers on or when intervention is supplied to a terminal that had previously indicated "intervention required"; in this case, the SNA sense is X'00000200'). Exception requests can also indicate other conditions such as an SNA sequence number error (X'20010000'), or unit specify in combination with data check (X'00000404'). An exception request with all zero sense (X'00000000') can also occur; this exception should be ignored. However, this RU and any other exception request has the begin-bracket indicator on if bracket protocols are being used in the session, and the session had just previously been in the between-brackets state. These RUs can cause a bracket to be started.

Table 58. SNA sense information received at the application program

SNA sense	SNA definition	Cause for exception
80*xxyyyy*	Path error	Request could not be delivered [1]
400A0000	No-response not allowed	RESPOND=(NEX,NFME,NRRN)
400B0000	Chaining not supported	CHAIN=(FIRST or MIDDLE or LAST)
20030000	Bracket state error	BRACKET=NBB and no bracket currently exists [2]
20010000	Sequence number error	Session sequence number error
10030000	Function not supported	CONTROL=(DATA or CLEAR)
10000020	Request error	Command rejected
08210000	Session parameter not valid	Parameters not valid in BIND
08130000	Bracket bid reject—No RTR Forthcoming	BRACKET=BB and a bracket already exists [2]
0000*zzzz*	Other exception	Device exception—USENSEI values are defined in Table 59 on page 333.

Table 58. SNA sense information received at the application program (continued)

SNA sense	SNA definition	Cause for exception

Notes:

1. *xx* is defined in *SNA Formats* For a PU type 1 3270 terminal, *yyyy* can be set to 0010 (intervention required). For a BSC 3270 terminal attached to a communication controller, *yyyy* represents the NCP system response byte and extended response byte returned for some path error conditions. For information on how these bytes are defined, refer to *NCP and EP Reference Summary and Data Areas*.
2. This sense code applies only if bracket protocols are being used in the session.

Table 59. Explanation of USENSEI information

USENSEI byte 0	USENSEI byte 1	Meaning
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	
. . . . x . . .	. . . .	Device busy
. . . . . x . .	. . . .	Unit specify
. . . . . . x .	. . . .	Device end
. . . . x	. . . .	Transmission check
. . . .	. . x	Command rejected
. . . .	. . . x	Intervention required
. . . .	. . . . x . . .	Equipment check
. . . .	. . . . . x . .	Data check or bus-out check
. . . .	. . . . . . x .	Control check
. . . .	 x	Operation check
x x x x	x x	Reserved

Note: Transmission check is indicated for a channel-related error (such as PCI, channel program check, protection check, channel data check, channel control check, interface control check, or chaining check) for a channel-attached non-SNA 3270.

The application program can disobey the LU type 0 protocols by attempting to send:

- A data-flow-control request
- A response
- A request indicating other than a single-request chain
- A request that does not ask for a definite or exception response type 1 (FME).

If you attempt any of the preceding items, the following results occur, depending upon the actual type of terminal used in the session:

- The SEND macroinstruction is rejected with (RTNCD,FDB2)=(X'14',X'47').
- VTAM returns a negative response.
- VTAM ignores the protocol violation, and unpredictable results can occur.

Session parameter

VTAM defines a standard session parameter set for LU type 0 3270 sessions. This set of parameters specifies with the following MODEENT macroinstruction in the IBM-supplied default logon mode table:

```
IBMS3270 MODEENT LOGMODE=S3270,                                        C
         FMPROF=02,                                                  C
         TSPROF=02,                                                  C
         PRIPROT=71,                                                 C
         SECPROT=40,                                                 C
         COMPROT=2000
```

If desired, the system programmer can replace this entry in the default logon mode table or define a different entry for each 3270 terminal as described in the z/OS Communications Server: SNA Resource Definition Reference. Do this to specify that you should not use brackets. The following table indicates the required and optional bit settings of certain session parameter fields. The definitions of these bits are in Appendix F, "Specifying a session parameter," on page 793, in the ISTDBIND DSECT.

```
FMPROF:   0000 0010
TSPROF:   0000 0010
PRIPROT:  0DRR 000B
SECPROT:  0100 0000
COMPROT:  00B0 0000 0000 0000
```

where:

`D` can be 0 or 1.

`RR` can be 01, 10, or 11.

`B` is 1 if brackets are used in the session.

Also, the system programmer can specify the PSERVIC parameter on MODEENT to convey information, such as display screen size or Query support, to the application program. SNA does not define the format of the presentation-services field for LU type 0; however, a good convention is to use the same format as used for LU types 2 and 3. See Appendix F, "Specifying a session parameter," on page 793 for information on LU types 2 and 3.

Device characteristics field

The device characteristics field (see IDSDVCHR in Appendix E, "Control block formats and DSECTs," on page 739) contains only limited information about a 3270 LU. It contains the model number specified in the FEATUR2 network definition operand for the 3270 terminal (refer to z/OS Communications Server: SNA Resource Definition Reference for information on operands). Use this model number to derive a default display screen size or the buffer size for the terminal. The preferable way to determine this default, however, is to use information that can be supplied in the presentation-services field of the session parameter as described in the preceding section.

Also usually available in the device characteristics field is the physical device address needed as the "from" address for the 3270 copy function. Also, the copy function is not supported by channel-attached 3270s.

The field labeled DEVTCODE in the device characteristics field should not be used to attempt to determine the device type of the terminal. All 3270 terminals should be considered to be LUs and the DEVTCODE field must be ignored.

Logon message

The logon message obtained with INQUIRE OPTCD=LOGONMSG is the user data field from the logon or Initiate that requested the session. This is exactly the same as for LUs other than LU type 0.

Logoff

Terminal operators of the 3270 should be advised that, unless they enter a logoff prior to actually shutting off the device, under certain circumstances, unauthorized users could gain access to the application program with which the terminal has been in session.

Test request

The message that results from pressing the TEST REQ key is intercepted by the network and is not received as an RU by the application program. The test request message is usually sent to the SSCP. The action taken by the network depends on the attachment of the terminal issuing the test request message. In Table 60, "<*xxxxx*>" means that the terminal operator typed the information represented by "*xxxxx*" and then pressed the TEST REQ key; "*poll*" is the specific poll address of a terminal on a BSC cluster controller. The poll address can be entered in either uppercase or lowercase. The action taken as a result of an unformatted system services (USS) request depends on the data in the request. For example, a LOGOFF causes the scheduling of the LOSTERM exit routine.

Table 60. Actions taken by the network when a test request message is received

Terminal attached to	Terminal input	Action
Channel on this host	CLEAR key, then <*data*>	"*data*" is treated as a USS request.
BSC line attached to a communication controller	CLEAR key, then <*poll, poll*> with no data allowed	LOSTERM is scheduled with reason code 12.
BSC line attached to a communication adapter (in another domain)	CLEAR key, then <*poll, poll, data*>	"*data*" is treated as a USS request.
SDLC line attached to a communication controller	CLEAR key, then <*data*>	"*data*" is treated as a USS request.

Note: If a 3270 attention other than Test request occurs and if no LU-LU session currently exists with that 3270 LU, then the associated input is sent to the SSCP. Such input, if created using the ENTER key or a magnetic card reader, is treated as a USS request (for example, a logon can be entered). Input associated with any other type of 3270 attention is rejected by the SSCP with an error message.

Summary of differences among LU type 0 3270 terminals

Although VTAM masks most of the differences that exist among the various LU type 0 3270 terminals, situations exist in which an application program can detect terminal differences. Normally, you should write application programs to be independent of the terminal differences; otherwise, the program might have to be modified later when additional LU type 0 3270 terminals must be supported by the program. This is particularly important in a multiple-domain environment, in which the application program can be in one domain and the terminals can be in other domains. The types of terminal attachment can change in the other domains

with little or no notice to the person maintaining the application program. Application programs should be coded in the first place to avoid being sensitive to such changes.

The following items are discussed in more detail earlier in this chapter. For each item, a suggested coding technique is given.

- The queued-response indicator is not supported by the PU type 1 3270. See "Responses" on page 330. Do not use QRI for LU type 0 3270 terminals.
- The sequence number wraps at 255 for some 3270 LUs and at 65 535 for others. See "Sequence numbering" on page 331 and "Sequence number dependencies for LU type 0 3270 terminals" on page 894. Always use the full 2-byte sequence number for comparisons. Do not become dependent on when the sequence number wraps, for example, by trying to predict what the sequence number is when SEND completes.
- Sometimes a negative response for a BB request which was sent to a BSC 3270 terminal causes entry to the in-bracket state instead of the between-brackets state. See Figure 83 on page 331 which also describes a recovery procedure to use if a subsequent BB request is rejected. This procedure should be used for all terminals.
- Not all terminals return the same SNA sense codes for particular error situations. See "Exception conditions and sense information" on page 332. Application programs should be coded to handle all of the listed codes from any LU type 0 3270 terminal. The recovery procedure used for each sense code should be independent of which type of terminal sent the code.
- If an application program disobeys LU type 0 3270 protocols, the results depend on the terminal involved. See "Exception conditions and sense information" on page 332. Always obey the LU type 0 3270 protocols.
- The copy function is not available for non-SNA channel-attached 3270 terminals. Do not become dependent on the use of the copy function.
- Do not use the model number to determine screen size; instead, use information from the presentation-services field of the session parameter described in the next item.
- Different 3270 terminals can support different screen sizes and might support other extended 3270 data stream facilities such as Query. For further information, see "Session parameter" on page 333. Application programs can use the presentation-services field of the session parameter for the terminal to determine such capabilities for the terminal. Note that this requires the system programmer to code a logon mode table for the terminal to specify this information. If a terminal supports Query, that facility can, in turn, be used to determine details about the capabilities of the terminal. (Refer to *3174 functional description*.)

Chapter 12. Coding for the communication network management interface

This chapter describes the coding required for an application program to function on the communication network management (CNM) interface. You can also refer to z/OS Communications Server: SNA Customization for information related to the CNM routing table and the z/OS Communications Server: SNA Network Implementation Guide for information about authorizing application programs to use the CNM interface.

CNM interface

Using the CNM interface, an application program can communicate with the VTAM SSCP to acquire information from physical units (or to send information to physical units) in session with the SSCP. This communication is made possible by two processes:

- The SSCP and the PU exchange information through an SSCP-PU session.
- The application program and the SSCP exchange information over the CNM interface.

In most cases, the application program and the SSCP communicate over the CNM interface by using Forward and Deliver request units (RUs). A Forward RU places in a "data envelope" all data or requests for data that the application program wants sent to the PU in session with the SSCP. The SSCP transmits the data or the request for data to the PU through the SSCP-PU session. Similarly, the PU replies with the requested information (or provides unsolicited information) through the PU-SSCP session. A Deliver RU places the data provided by the PU in a "data envelope." The SSCP then sends the Deliver RU to the application program over the CNM interface.

Some communication between the application program and the SSCP over the CNM interface can use a group of request units that are not embedded in Forward or Deliver RUs. For example, the TR-INQ (Translate Inquiry) RU and the TR-REPLY (Translate Reply) RU can be passed to or sent by the CNM application program without embedding Deliver RUs. See "Types of network services request units not embedded" on page 350 for more information about these RUs.

Functions of the application program

With the SSCP transmitting information between an application program and a PU, an application program can request data from a PU or send data to a PU. Specifically, an application program can:

- Request maintenance-related information from a PU and receive maintenance-related information from a PU
- Request session data from VTAM and NCP
- Request performance data from VTAM
- Receive a load request from a PU and send load data to a PU
- Receive hardware alerts from a PU

© Copyright IBM Corp. 2000, 2013

Gathering maintenance-related information from a PU

About this task

An application program, using VTAM facilities, can request and receive status and maintenance-related information from physical units within the same domain. This type of application program is called a CNM application program. For example, the NetView program is a CNM application program that receives error detection data through the CNM interface.

Gathering session data from VTAM and NCP

About this task

A VTAM application program can request and receive session-related data from VTAM and NCP. A VTAM application program can also collect session information about cross-network sessions and information about the explicit route being used for a session. A VTAM application program uses the CNM interface to do the following:

- Gather information about subarea nodes and about the transmission groups that comprise an explicit route
- Initiate and terminate the collection of session data from the SSCP.

Some examples of IBM-supplied VTAM application programs that use the CNM interface to gather session data are:

- The NetView program
- NLDM Release 2 or 3.

Gathering performance data from VTAM

The performance monitor interface provides a means by which certain CNM application programs, called monitors, can request and receive information from VTAM pertaining to internal performance activity and resource utilization. Based on the raw data received over this interface, monitors can then report information on a real-time, interactive basis to their end users. Monitors use the CNM interface to control their data collections.

Loading a PU

An application program, using VTAM facilities, can load a PU within its domain and return the completion status of the load operation to VTAM. For example, the Downstream Load Utility (DSLU) licensed program receives load requests and sends load data over the CNM interface.

Receiving hardware alerts

Specific alert RUs can be routed to a specific CNM application. These alerts are generated by devices in the network in response to hardware, software, or line failures.

Request unit flow

An application program establishes a session with the VTAM SSCP when it opens its ACB. Communication then proceeds through the exchange of formatted Forward and Deliver request units. Both RUs contain network services request units (NS RUs) and, if appropriate, the name of the resource to receive the NS RU.

Figure 84 shows an example of Forward and Deliver RU flow. The application program (using the SEND macro) sends VTAM a formatted Forward RU. The Forward RU contains an NS RU (indicated in parentheses in Figure 84) and the name of the SSCP or the name of a PU to receive the NS RU. If the Forward RU contains the name of a PU, the VTAM SSCP sends the embedded NS RU to the specified PU over an SSCP-PU session. If the Forward RU contains the name of the SSCP, then the embedded NS RU is routed to the application program specified in the CNM routing table.

If the NS RU is not within a Forward RU, the NS RU is also processed by the SSCP.

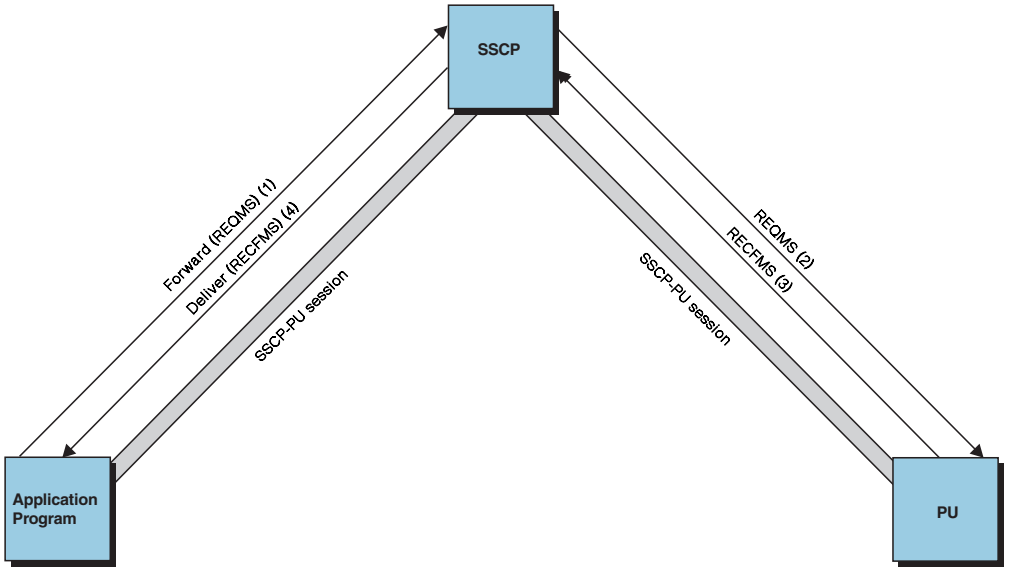


Figure 84. Example of Forward and Deliver request unit flow

The application program uses the RECEIVE macroinstruction to receive Forward responses and formatted Deliver request units from the VTAM SSCP. The Deliver RU contains an embedded NS RU (indicated in parentheses in Figure 84). This NS RU might have originated within the SSCP or it might have been sent to the SSCP by the PU over a PU-SSCP session. The PU can send an NS RU to reply to an NS RU (contained in a Forward RU), to transmit unsolicited data, or to request a load.

The application programmer must know the format of the Forward and Deliver RUs, how to specify network names within the network name fields of these RUs, and the types and formats of embedded NS RUs that can be placed within the Forward and Deliver RUs. VTAM internal CNM RUs should only be sent from the CNM application that is currently in session with ISTPDCLU. See “Requirements for receiving session-awareness and trace data” on page 355 for more information.

Application program coding requirements for the CNM interface

For an application program to use the CNM interface, AUTH=CNM must be specified on its APPL definition statement.

To receive unsolicited RUs, an application program must be authorized to use the CNM interface and be designated in the CNM routing table (with the name specified in the name field of the APPL definition statement). Refer to z/OS Communications Server: SNA Customization for more information about authorizing application programs to use the CNM interface and designating the application program name in the CNM routing table.

If a VTAM application program has been authorized to use the CNM interface, the program can gain access to the SSCP-LU session by specifying PARMS=(NIB=*nib address*) in its ACB. This NIB contains parameters describing the session to be established (the SSCP-LU session) just as an NIB does when it is used to establish an LU-LU session with OPNDST or OPNSEC. After OPEN completes processing for the ACB, the NIB contains the CID of the SSCP-LU session that was established.

The application program uses a subset of the RPL-based macroinstructions (SEND, RECEIVE, RESETSR, EXECRPL, and CHECK) to communicate on the SSCP-LU session just as it uses these macroinstructions to communicate on LU-LU sessions. Because the SSCP-LU session uses FM profile 6, it uses only a very limited set of data-flow-control functions; for example, the SSCP sends no data-flow-control requests and sends and receives only single-request chains. Refer to *SNA Formats* for details about FM profile 6. Similarly, because you use TS profile 1 on the SSCP-LU session, you do not send or receive session-control requests. Thus, do not use SESSIONC.

The NIB macroinstruction specified by PARMS=(NIB=*nib address*) has the permissible values listed here. See "NIB—Create a node initialization block" on page 433 for details.

```
NAME=blanks
USERFLD=user data
LISTEND=YES
SDT=SYSTEM
EXLST=exit list address
ENCR=NONE
RESPLIM=response limit
LOGMODE=0
BNDAREA=0
PROC=(CA or CS or RPLC,
NDFASYX or DFASYX,
NRESPX or RESPX,
NCONFTXT or CONFTXT,
KEEP or TRUNC,
SYSRESP or APPLRESP,
ORDRESP or NORDRESP,
NNEGBIND or NEGBIND)
```

RPL, SEND, RECEIVE, RESETSR, and EXECRPL macroinstructions used to access the SSCP-LU session can have the following subset of RPL operands specified. See the macroinstruction descriptions in Chapter 13, "Conventions and descriptions of VTAM macroinstructions," on page 371 for details.

2. FM profile 6 does not allow any expedited data-flow-control requests to be sent to or from the SSCP.

ACB=*acb address*
ARG=(*register containing CID of SSCP-LU session*)
AREA=*data area address*
AREALEN=*data area length*
RECLEN=*data length*
ECB=INTERNAL
ECB=*event control block address*
EXIT=*rpl exit routine address*
BRANCH=YES or NO
SEQNO=*sequence number*
POST=SCHED or RESP
RESPOND=(EX or NEX,FME,NRRN,QRESP or NQRESP)
CONTROL=DATA
CHAIN=ONLY
CHNGDIR=(NCMD)
BRACKET=(NBB,NEB,NCEB)
RTYPE=(DFSYN or NDFSYN,DFASY or NDFASY,RESP or NRESP)
STYPE=REQ or RESP
SSENSEO=0 or CPM or STATE or FI or RR
SSENSMO=*system-sense modifier value*
USENSEO=*user-sense value*
CRYPT=NO
CODESEL=STANDARD
OPTCD=(TRUNK or KEEP or NIBTK, FMHDR, SPEC or ANY, SYN or ASY, CS or CA, Q or NQ, BUFFLST or NBUFFLST, NLMPEO, USERRH or NUSERRH)

CNM interface requests and responses

The structure and operation of the CNM interface is based on Systems Network Architecture protocols and procedures. Communication of a request embedded within another request is a function of management services. An application program embeds both SSCP-PU and LU-SSCP network services requests; the VTAM SSCP embeds both PU-SSCP and SSCP-LU network services requests, which are routed by the SSCP to the application program.

The Forward RU, which the SSCP receives from the application program, contains an embedded NS RU and a destination name. If the destination name is the name of one of the SSCP's PU resources, the SSCP sends the embedded NS RU to the PU using the destination's SSCP-PU session and session rules. If the destination name is the SSCP's name, the embedded request is processed by the SSCP. Responses to an application program LU-SSCP Forward request can return a 4-byte sense field, which is placed in the user's RPL field.

When an SSCP either originates a request or receives one from a PU and determines that the request is for an application program, it embeds the RU (without change) into a Deliver request, translates appropriate network addresses into network names, places them in the Deliver request, and then sends the completed request to the application program.

Protocols and procedures

An application program SSCP-LU session uses TS profile 1 and FM profile 6.

For each destination PU, the application program must adhere to the request mode supported by that PU on the SSCP-PU session; otherwise, the request is rejected with a negative response (sense code = X'08510000'). For example, a PU that operates in immediate request mode should not be sent a new request until it has responded to the last request.

A procedure-related identifier (PRID) is available for use by the application program for request/reply correlation in some types of RUs. The PRID, if present, is contained in bytes 5–6, bits 4–15, in the CNM standard header (described in "Standard CNM headers" on page 343).

The application program generates a PRID if it is sending a Forward request that solicits a reply. The destination PU, which receives the request, saves the PRID (which was possibly translated by the SSCP) and returns it in all replies that correspond to the request. The PRID value (retranslated by the SSCP, if necessary) is returned to the application program within the embedded reply RU of the Deliver request.

The application program must determine (usually through use of its own timer) that the Deliver RU is not received for a PRID value. No notification is given to the application program if the PU to which the embedded NS RU was sent becomes inoperative before the reply RU can be sent by the PU.

A CNM application program is responsible for proper operation when a Deliver RU is not received for its corresponding Forward RU. A CNM application can cancel the correlation between the Deliver and Forward RUs. The Deliver RU that would have been correlated to the Forward RU is discarded.

A CNM RU (X'810814') with a subtype indicating cancel can be sent to VTAM by a CNM application to cancel a Deliver RU. The Cancel RU contains the PRID assigned to the Forward RU, and the name of the destination used in the Forward RU. The CNM application must be able to receive the Deliver RU until the response to the CNM RU is received.

If the reply from the PU consists of multiple requests, the PU inserts the same PRID into each request. To handle this, each such reply has a "not last" bit defined as part of its format. This bit is used by the PU that is sending the requested data to indicate that at least one additional request RU is still to be sent. A flag is set in the Deliver RU by the SSCP when the embedded request contains a standard CNM header with a defined PRID field. A solicited flag is set in the CNM header by the replying PU whenever the PU generates a reply. Unsolicited Deliver requests have the solicited flag set off in the embedded RU. A PRID value can be reused after the application program receives the last reply RU using that PRID.

Request unit (RU) formats

The following sections describe the formats of the RUs that can be used on the CNM interface.

Standard CNM headers

The CNM headers, shown in Table 61 and Table 62, allow a standard method of routing replies to requesting application programs; this routing is based on PRID correlation. The header format also allows a standard correlation scheme for the application program. When an NS RU, embedded in a Deliver request, contains a standard CNM header, a CNM header bit is set in the flag field of the Deliver RU that is sent to the application program. When an NS RU, embedded in a Forward request, follows the standard format and has reply data associated with it, the application program can set the reply-requested field of the Forward request to receive the requested reply data. Failure to set this flag causes reply data to be treated as unsolicited data by the SSCP and to be routed to the application program assigned to receive such an unsolicited NS RU. For details on the routing table, refer to z/OS Communications Server: SNA Customization.

Table 61. Standard CNM header, SSCP-PU request format

Byte	Bit	Value	Description
0–2			Network service header
3–7			CNM Header is bytes 3–7
3–4			CNM target ID; refer to CNM target ID descriptor field in byte 5
5–6	0–1		Reserved
	2–3		CNM target ID descriptor
		B'01'	Network address of target resource is in bytes 3 and 4
		B'00'	Local address of target resource is in byte 4; byte 3 is reserved
	4–15		PRID
7	0		Limited usage (See the following note)
	1		Reserved
	2–7		Type, specifies the format of request-specific data when present in bytes 8–n

Note: Bit 0 in byte 7 is used as a set/reset indicator in certain requests, for example, REQMS. Specific requests can use this bit where required. Whenever a central data base is used, requests to reset counters recorded by the data base must come only from the application program that updates the data base with the reset information; otherwise, the data base contains misleading information.

Table 62. Standard CNM header, PU-SSCP request format

Byte	Bit	Value	Description
0–2			Network service header
3–7			CNM header is bytes 3–7
3–4			CNM target ID; refer to CNM target ID descriptor field in byte 5
5–6	0–1		Reserved
	2–3		CNM target ID descriptor
		B'01'	Network address of target resource is in bytes 3–4. Network address information should not be used by application programs, because this network-specific data is subject to change.

Table 62. Standard CNM header, PU-SSCP request format (continued)

Byte	Bit	Value	Description
		B'00'	Local address of target resource is in byte 4; byte 3 is reserved
	4–15		PRID
7	0–1		Reply Flags:
	0		Solicitation Indicator Reply Flag
		B'0'	Unsolicited request
		B'1'	Solicited (reply) request
	1		Not-Last-Request Indicator
		B'0'	No more data (for example, last reply)
		B'1'	More data (for example, not-last reply)
	2–7		Type (Specifies the format of the data in bytes 8–n)

Forward request unit

The format of the Forward RU is shown in Table 63. The Forward request flows from the application program to the VTAM SSCP. The embedded NS RU is contained in bytes 8–n. The NS RU is removed from the Forward RU and sent to the destination named in the Forward request. Network names are required in the Forward RU. A Forward request includes the name of the destination, which is changed to an address by the SSCP for transmission purposes. The name of the target resource that the embedded NS RU applies to is also placed in the Forward request. Usually, this target-resource name is changed to an address and placed in the CNM target ID field of the embedded NS RU by the SSCP. For those NS RUs without CNM headers, such as NS-IPLFINAL, NS-IPLINIT, NS-IPLTEXT, and NS-LOADSTAT, the target name is resolved into a network address, but it is not placed in the embedded NS RU.

The category field in an embedded NS RU specifies a session within the same domain.

Table 63. Forward request unit format 0 request format

Byte	Bit	Value	Description
0		X'81'	Network services, logical services
1		X'08'	Management services
2		X'10'	Request code
3		X'00'	Format 0
4	0–4		Reserved, flags
	5		NMVT X'04' subvector processing required
		0	For NMVT, ignore the target names; for the other RUs, place the translated target names in bytes 3–4 of the embedded request
		1	Place the translated addresses in the X'04' vector
	6	0	Embedded NS RU contains a PRID
		1	Embedded NS RU contains no PRID
	7	0	Embedded NS RU contains a CNM header
		1	Embedded NS RU contains no CNM header

Table 63. Forward request unit format 0 request format (continued)

Byte	Bit	Value	Description
5			Reserved
6–7			NS RU length
8–n			NS RU
n+1–p			Network name of destination PU:
n+1			Type:
		X'F1'	PU
n+2			Destination PU name's length in binary
n+3–p			Destination PU name
p+1–q			Network name of target resource:
p+1			Type:
		X'F1'	PU
		X'F3'	LU
		X'F7'	Adjacent link station
		X'F9'	Link
p+2			Length, in binary, of target resource
p+3–q			Target resource name
q		X'00'	End of record for single targets. For additional targets (if any): repeat the preceding structure from p+1 as required. The list of targets is ended by a type code of X'00'.

Deliver request unit

The format of the Deliver RU is shown in Table 64. The Deliver request contains an embedded NS RU and flows from the VTAM SSCP to an application program within the same host. The SSCP does not modify any information within the embedded NS RU.

Table 64. Deliver request unit format 0

Request format:	Byte	Bit	Value	Description
0			X'81'	Network services, logical services
1			X'08'	Management services
2			X'12'	Request code
3		0–7	X'00'	Format 0
4		0–4		Reserved, flags
		5	0	Only one target resource name is included.
			1	Multiple target resource names are included.
		6	0	Embedded NS RU does not relate to resources in a secondary network.
			1	Embedded NS RU relates to resources in a secondary network.

Table 64. Deliver request unit format 0 (continued)

Request format:	Byte	Bit	Value	Description
		7	0	Embedded NS RU contains a CNM header
			1	Embedded NS RU does not contain a CNM header
5				Reserved
6–7				NS RU length
8–n				NS RU
n+1–p				Network Name of Origin PU
n+1				Type:
				X'F1' PU
n+2				Length, in binary, of origin PU name
n+3–p				Origin PU name
p+1–q				Network name of target resource
p+1				Type:
				X'F1' PU
				X'F3' LU
				X'F7' Adjacent link station
				X'F9' Link
p+2				Length, in binary, of target resource
p+3–q				Target resource name
q+1–r				Configuration hierarchy name 1 (Note: This name is in the same format as the target resource name.)
.				.
r+1–s				Configuration hierarchy name *M*
s+1			X'00'	End of configuration hierarchy name fields
s+2–t				Additional target resource name
.				.
.				.
.				.
.				.
.				.
				(See Table 65 on page 348 for format of target names)
.				.
t+1–u				
.				.
t+1–u				
.				.
t+1–u				Additional target resource name *k*
u+1			X'00'	End of additional target resource name fields.

The Deliver RU includes the name of the originator of the embedded NS RU. The name of the originator is derived from the origin network address contained in the transmission header (TH) in the path information unit (PIU) that provided the NS RU.

Target resource names: For NMVT RUs, multiple target resource names are allowed.

The target-resource name field contains the name of the first network component described by the embedded NS RU.

- If the flag byte (byte 4, bit 5) in the Deliver RU indicates that only one target-resource name is included, the target-resource name field contains the name of the first and only target resource. The target-resource name is derived from the resource address in the NS RU. The target resource name is the same as the name of the originator of the embedded NS RU when the RU does not contain a resource address.

  When only one target-resource name is included in the Deliver RU, the Deliver RU ends with the X'00' type byte that indicates the end of the configuration hierarchy list.
- If the flag byte in the Deliver RU indicates that multiple target-resource names are included, the target-resource name field contains the name of the first target resource, and subsequent names (1–M additional names) follow the X'00' type byte that indicates the end of the configuration hierarchy list.

  When multiple target-resource names are included in the Deliver RU, the configuration hierarchy list is provided only for the first target resource. The Deliver RU ends with the X'00' type byte that indicates the end of the additional target-resource name fields.

On all the operating systems, the configuration-hierarchy name fields provide a list of PU and link configuration information. The number of names varies. See Table 61 on page 343 for standard CNM headers. The configuration-hierarchy list contained in the Deliver request is generated by using one of the following algorithms. The hierarchy contains the list of addressable nodes between the target-resource name and the PU that contains the boundary function for the resource.

- When the resource is a PU or an LU at a PU type 1 or PU type 2 node, the configuration hierarchy is an ordered list of the network names of each component in the connection, starting from the resource and including the name of the PU containing the boundary function. The resource name is not included in the list.
- When the resource is an LU, PU, or link that does not require boundary function support at a PU type 4 or PU type 5, there is no list.
- When the resource is a link station, the configuration hierarchy is an ordered list of names consisting of the link and PU type 4 or 5 node to which the line is attached (and which is the originator of the embedded RU).
- When the resource is a PU for which a load is being requested, there is no list.

Single target names: The target-resource name field contains the name of the network component described by the imbedded NS RU. The target-resource name is derived from the resource address in the NS RU. Because a resource address does not appear in the NS RU sent by a PU type 1 or a PU type 2, the resource name is the same as the name of the originator of the embedded NS RU.

NMVT (Network Management Vector Transport) considerations:

When multiple network or local addresses are included in an NMVT RU, the additional target-resource name fields provide a list of information about each target resource other than the first target resource. The additional target-resource name fields in the Deliver RU are derived from multiple network or local addresses that are contained in the SNA address list (hex 04) subvector of the embedded NMVT RU. If the SNA address list subvector indicates that groups of two addresses are session partners, the procedure correlation identifier (PCID) that is associated with the session is placed in the Deliver RU as part of the target-resource name field. Refer to *SNA Formats* for the meaning of and the types of information that are contained in the NMVT RU.

Request unit names

The application program uses only network names in all requests. Network name fields in the Forward and Deliver requests use the format shown in Table 65. The SSCP changes these names into addresses. The destination name in a Forward RU is changed to a destination network address by the SSCP prior to sending the embedded NS RU to the destination. When an embedded NS RU is to contain a target-resource address, the SSCP resolves the target-resource name in the Forward request into an address, and places the address into the NS RU. This address is either a local address or a network address, depending upon the PU type of the destination resource. If the destination resource is a PU type 1 or 2, the target resource must be the PU itself. The SSCP does not validate the network name type field in the Forward RU.

NS RUs are delivered to the application program with their address fields unchanged by the SSCP. The application program should not use the address fields because they are subject to change. The SSCP, however, resolves each address in the embedded NS RU to a network name and inserts the network name into the Deliver RU. If the origin PU is a PU type 1 or 2 or if no SNA address list subvector is present in an NMVT RU, only local address 0 (that is the address of the PU itself) is supported.

If multiple network addresses are included in an NMVT RU, and the groups of two addresses are indicated as session partners, the PCID associated with the session is part of the network name field in the Deliver RU.

Table 65. Format of additional targets in Deliver requests

Byte	Value	Description
0–i		Network name/PCID of target N
0	X'B3'	Procedure correlation identifier (PCID) for the session defined by preceding F3-E3 pair
	X'2D'	Group name
	X'E3'	Session partner name
	X'F1'	Physical unit or link station name
	X'F3'	Logical unit name
	X'F4'	SSCP name
	X'F5'	Procedure name
	X'F7'	Link station name
	X'F9'	Link name
	X'FC'	Channel-link locally defined name

Table 65. Format of additional targets in Deliver requests (continued)

Byte	Value	Description
	X'FF'	Error code indicating that address to name translation is not possible
1		Length, in binary, of target name
2–i		Symbolic name of target in EBCDIC characters

Examples of embedded network services request units

Examples of permissible types of embedded network services request units are described in Table 66.

SNA Formats does not include the formats for the NS-INITLOAD and NS-LOADSTAT RUs; these two RU formats are shown in Table 67 on page 350 and Table 68 on page 350.

Table 66. Examples of embedded network services request units

Request	NS header	PRID correlation	
FORWARD	X'810810'		
NMVT	X'41038D'	**Yes**	This RU is unsolicited, or is a solicited request requiring a reply.
NS-IPL-FINAL	X'410245'	**N/A**	The RU does not contain a standard CNM header.
NS-IPL-INIT	X'410243'	**N/A**	The RU does not contain a standard CNM header.
NS-IPL-TEXT	X'410244'	**N/A**	The RU does not contain a standard CNM header.
NS-LOADSTAT	X'3F0234'	**N/A**	The RU does not contain a standard CMM header.
REQMS	X'410304'	**Yes**	The embedded reply is an RECFMS RU.
DELIVER	X'810812'		
NMVT	X'41038D'	**Yes**	This RU is unsolicited, or a solicited request requiring a reply.
NS-INITLOAD	X'3F0233'	**N/A**	The RU is sent unsolicited and does not contain a standard CNM header.
RECMS	X'010381'	**N/A**	This RU is sent unsolicited by the PU, and does not contain a standard CNM header.
RECFMS	X'410384'	**Yes**	This RU is unsolicited or is solicited by an embedded REQMS RU.
ROUTE-INOP	X'410289'	**No**	The RU is sent unsolicited to the CNM application program to provide information about failing routes within this network or another network. It does not contain a standard CNM header.

Table 67. Initiate load request (NS-INITLOAD) request unit format

Byte	Bit	Value	Description
0		X'3F'	Network services, access method RU
1		X'02'	Configuration services
2		X'33'	Request code
3–6			Reserved
7–8			Reserved
9			Load request state:
		X'00'	Load requested during PU activation
		X'01'	Load requested during PU activation
10–17			IPL load module name

Table 68. Load status request (NS-LOADSTAT) request unit format

Byte	Bit	Value	Description
0		X'3F'	Network services, access method RU
1		X'02'	Configuration services
2		X'34'	Request code
3–6			Reserved
7–8			Reserved
9			Load request state:
		X'00'	Load requested during PU activation
		X'01'	Load failed—unable to activate
10			Load status:
		X'00'	Load successful
		X'01'	Load failed—unable to process load request. Bytes 11–17 set to zero.
		X'02'	Load failed during the load procedure. Bytes 11–17 contain additional information.
11–13			Request code of the failing NS RU
14–17			Sense data returned in the negative response for the failing NS RU.

Types of network services request units not embedded

Each of the request units described in Table 69 on page 351 can be either sent or received by the application program without embedding it in a Forward or Deliver RU. The formats of these Network Services RUs are shown in Table 70 on page 351 through Table 79 on page 354.

Alias application

The ALIAS application translates LU, class-of-service, owning SSCP, and logon mode names when the names are duplicated or have other meanings in other networks. VTAM requests translation with a Translate Inquire RU. The ALIAS returns the translated name in a Translate Reply RU. For further details on the translated name, see *Planning for NetView, NCP, and VTAM*.

The requirements for an ALIAS application are:

- Authorized as a CNM application
- Authorized by the CNM routing table (refer to z/OS Communications Server: SNA Customization) to receive Translate Inquire RU
- Able to receive unsolicited Translate Inquire RUs
- Able to send Translate Reply RUs as solicited data
- Able to correlate the procedure relation ID received in the Translate Inquire RU with the Translate Reply RU.

The sense code in Table 79 on page 354 shows whether the ALIAS translation was unsuccessful.

The sense code in Table 83 on page 355 represents completion of the translation for the specific element or name. Translation can occur for fewer than all items on the Translate Inquire RU.

Table 69. Types of network services request units not embedded

Request	NS header	PRID correlation	Description
CNM	(X'810814')	Yes	The RU is sent by the CNM application program to control session-awareness and trace processing within VTAM. Depending on the CNM RU type field, a reply can be returned by the SSCP.
TR-INQ	(X'3F0814')	Yes	The request is sent to the CNM application program to request name translations. The reply is a TR-REPLY RU.
TR-REPLY	(X'3F0816')	Yes	This RU is solicited with a TR-INQ RU. It is sent by the CNM application program and contains either translated names or a sense code indicating that the translation could not be done.

Table 70. CNM request unit format

Byte	Bit	Value	Description
0		X'81'	Network services, access method RU
1		X'08'	Management services
2		X'14'	Request code
3–4			Reserved
5–6			CNM identifier. (A local address or network address, depending on value of bits 2 and 3 of byte 5.)
	0–3		Reserved
	4–15		Procedure Relation Identifier (PRID)
7	0	0	Unsolicited request
		1	Reply request
	1	0	Last request
		1	Not last request
	2–7		Reserved
8			Subtype code

Table 70. CNM request unit format (continued)

Byte	Bit	Value	Description
9–n			CNM data

Table 71. Timeout CNM request unit format

Byte	Bit	Value	Description
0		X'81'	Network services, access method RU
1		X'08'	Management services
2		X'14'	Request code
3–4			Reserved
5–6			Correlator field
	0–3		Reserved
	4–15		The CNM PRID that is being canceled (from the Forward RU)
7	0	0	Reserved
8			Subtype code (X'1C' for a timeout request)
9–16			Destination name from the Forward RU for the request being canceled

The following tables show the Inquiry, Reply, and Constant formats for the ALIAS application translation.

Inquiry data

Table 72. Translate-inquiry request (TR-INQ) request unit format

Byte	Bit	Value	Description
0		X'3F'	Network services, access method RU
1		X'08'	Management services
2		X'14'	Request code
3–4			Reserved
5–6	0–3		Reserved
	4–15		Procedure relation identifier (PRID)
7	0	0	Unsolicited request
		1	Solicited request
	1	0	Last request
		1	Not last request
	2–7		Reserved
8–9			Count of translate-inquiry requests
10–13			Reserved
14–n			Translate-inquiry data[1]

Note:

1. Bytes 14–n represent each byte in this Translate-Inquiry Data table.

Table 73 represents one translate inquiry data record. This record, and possibly additional records, are contained in bytes 14–n of the Translate-Inquiry Request (see Table 72 on page 352). The total number of records is equal to the count value shown in bytes 8–9 of Table 72 on page 352.

Table 73. Translate inquiry data

Byte	Description
0	Length of request data
1	Input name characteristics
1	Reserved
2	Class of input name
3	Input name type code
4	Length of input name
5	Input name to be translated

Table 74. Network ID 1—ID in which the name to be translated was used

Byte	Description
0	Network ID type code (X'FE')
1	Length of network ID (maximum 8)
2	Network ID

Table 75. Network ID 2—ID of the network in which the translated name is used

Byte	Description
0	Network ID type code (X'FE')
1	Length of network ID (maximum 8)
2	Network ID

Constant values

Table 76 shows the value of the class-of-input names for bytes 2 of Table 73 and Table 80 on page 354.

Table 76. Class-of-input names

Value	Meaning
X'00'	Input name is an alias name
X'01'	Input name is real name

Table 77 shows the value of the input name type code (translate inquiry only) for byte 3 of Table 73.

Table 77. Input name type code (TR-INQ)

Value	Meaning
X'F3'	LU
X'FB'	Class-of-service name
X'FD'	Logon mode name

Table 78 shows the value of the input name type code (translate-reply) for Table 80 byte 3.

Table 78. Input name type code (TR-REPLY)

Value	Meaning
X'F3'	LU
X'F4'	SSCP
X'FB'	Class-of-service name
X'FD'	Logon mode name

Reply data

Table 79. Translate reply request (TR-REPLY) request unit format

Byte	Bit	Value	Description
0		X'3F'	Network services, access method RU
1		X'08'	Management services
2		X'16'	Request code
3–4			Reserved
5–6	0–3		Reserved
	4–15		Procedure relation identifier (PRID)
7	0	0	Unsolicited request
		1	Solicited request
	1	0	Last request
		1	Not last request
	2–7		Reserved
8–9			Count of translate-reply requests
10–13			Sense code
14–n			Translate reply data [1]

Note:

1. Bytes 14–n represent each byte in this Translate-Reply Data table.

Table 73 on page 353 represents one translate inquiry data record. This record, and possibly additional records, are contained in bytes 14–n of the Translate-Inquiry Request (see Table 72 on page 352). The total number of records is equal to the count value shown in bytes 8–9 of Table 72 on page 352.

Table 80. Translate reply data

Byte	Description
0	Length of reply data
1	Reply name characteristics
1	Reserved
2	Class of input name
3	Input name type code
4	Length of input name (maximum 8)
5	Input name to be translated

Table 81. Network identifier 1

Byte	Description
0	Network ID type code (X'FE')
1	Length of network ID
2	Network ID

Table 82. Network identifier 2

Byte	Description
0	Network ID type code (X'FE')
1	Length of network ID (maximum 8)
2	Network ID

Table 83. Translated name data

Byte	Description
0	Sense code for this reply
4	Translated name type code (maximum 8)
5	Length of translated name
6	Translated name

Requirements for receiving session-awareness and trace data

Session-awareness and access method trace data are requested from VTAM using the CNM requests sent over the CNM interface to the SSCP. For an application program to receive session-awareness and trace data, the application program must initiate two sessions with a VTAM application LU named ISTPDCLU: one for session-awareness data and one for trace data.

VTAM definition requirements

Because the ISTPDCLU application program is a subtask within the VTAM licensed program, no definition is required for its use. However, the user's application program must be defined to VTAM using an APPL definition statement. The operands of the APPL definition statement that must be specified (not defaulted) are:

- AUTH=ACQ
- VPACING=value determined by the user
- AUTH=CNM.

Interfaces and interactions

The two sessions that the application program must initiate with ISTPDCLU use the following LU-LU session type 0 protocols:

- The user's application program initiates the sessions and acts as the primary logical unit.
- No error recovery exists for the sessions.
- The user's application program must restart any failed session.

- The user's application program only receives data over the sessions; it might never send data.
- The ISTPDCLU application program sends data to the user's application program.
- No responses are requested from the primary logical unit.
- The session uses function management profile 2 and transmission profile 3.

The session parameter contained in the BIND must specify the following:

- No brackets and no chaining of RUs are permitted.
- Inbound data to the user's application program is paced.
- No responses are requested from the primary session end.
- The secondary session end runs in delayed request mode.
- Normal flow is specified.
- No compression is allowed.
- Data flow control RUs are not allowed.
- There are no function management headers.
- The SEND/RECEIVE mode is full-duplex.
- The LU type is 0.
- Cryptography is not supported for the session.
- Negotiable BIND is not supported.

When the ***trace data session*** is bound, the BIND user data area must contain the EBCDIC characters TRACE. When the session-awareness data session is bound, the BIND must contain the EBCDIC characters SESSA. The ISTPDCLU application program's SCIP exit routine ensures a session limit of two and rejects any BIND requests after the limit is exceeded.

Session-awareness data buffer

The format in Table 84 defines the layout of the session-awareness data buffer. This buffer is built by VTAM and is passed to the user's application program through the ISTPDCLU session that is established with SESSA in the user field of the BIND.

Table 84. Session-awareness data buffer header

Byte	Description
0–15	Data routing information
0–1	Binary number representing the total buffer size (in bytes)
2–3	Process identifier
4–7	Data-type identifier = X'00040001' (FUNC=CONFIG,SUBFUNC=SESSA)
8–15	Reserved
16–23	Buffer eyecatcher = 'NLDM SAW'
24–25	Buffer sequence number (0≤n≤X'FFFF')
26–29	Length of session notification entries
30–35	Reserved
36–47	Access method work area (not used by user's application program)
48–p	One or more session notification entries

Session notification data format

Session notification data is placed in buffers that might contain multiple entries of variable length. Each variable-length entry contains information about one session. The session information is presented in a series of vectors.

Trace data buffer

Table 85 and Table 86 define the format of the PIU trace data buffer. This buffer is built by VTAM and is passed to the user's application program through the ISTPDCLU session established with TRACE in the BIND user data field.

Table 85. PIU trace data buffer header

Byte	Description
0–15	Data routing information
0–1	Binary number representing the total buffer size (in bytes)
2–3	Process identifier
4–7	Data-type identifier
8–15	Reserved
16–25	Buffer eyecatcher = 'NLDM TRACE'
26–27	Buffer sequence number
28–29	Total number of PIU trace data buffer entries
30–35	Reserved
36–47	Access method work area

Table 86. PIU trace data buffer entry

Byte	Bit	Value	Description
0–6			Time stamp (high-order 7 bytes of STCK)
7	0	0	PIU truncated to 40 bytes maximum
		1	PIU is not truncated and requires multiple PIU trace data buffer entries. An integral number of 40-byte trace buffer entries is always used. The size of the PIU can be calculated using the data count field in the TH of the PIU.
	1-7		Buffer entry type: **X'00'** Normal PIU **X'01'** Normal PIU — inbound **X'02'** Normal PIU — outbound **X'03'** Access method specified symptom string **X'04'** Discard PIU — session active **X'05'** Discard PIU — session inactive **X'06'** Internally generated PIU
8–47			Trace data

Requirements for receiving performance data

Performance data can be requested and received using the performance monitor interface. This interface involves two-way communication between VTAM and the monitor as follows:

- Monitor to VTAM

  The monitor sends data collection requests to VTAM over the CNM interface.

  The monitor sends requests to VTAM over the program operator interface to modify an installation-wide performance monitor exit (ISTEXCPM) associated with the monitor.
- VTAM to monitor

  VTAM reports data to the performance monitor exit, which forwards this data to the monitor.

Performance monitor definition requirements for initialization

About this task

The following steps are necessary for VTAM to honor the data collection requests issued over the CNM interface:

- You must define your monitor as a CNM application, and as a secondary or primary programmed operator. This authorization can be specified on the APPL definition statement for the monitor as AUTH=CNM and AUTH=SPO|PPO, respectively. Refer to z/OS Communications Server: SNA Resource Definition Reference for more information on coding the APPL definition statement.
- You must open the monitor with PARMS=(PERFMON=YES) coded on the ACB macroinstruction. See "Access method control block (ACB)" on page 56 for details about coding the ACB. Failure to properly define your monitor will result in an open ACB error code of X'8C'. See "OPEN macroinstruction" on page 57 for details.

  Upon open ACB completion, you should test bit X'51' of the access-method-support vector list (mapped by the ISTAMSVL DSECT) to verify that VTAM supports the performance monitor interface. The performance-monitor vector may also be returned in the resource-information vector list (mapped by the ISTRIVL DSECT). See "The access-method-support vector list" on page 62 for details.

An installation-wide performance monitor exit for your monitor must also be active in order to retrieve performance data from VTAM. The exit must be coded and installed according to the requirements in z/OS Communications Server: SNA Customization.

Data collection mechanism

The performance monitor interface provides a method to control the collection of performance data. The CNM RUs provided to support this process are:

- Start Performance Data Collection (X'22')
- Stop Performance Data Collection (X'23')
- Stop All Performance Data Collection (X'24')
- Collect Performance Data (X'25')

A monitor initiates a data collection by issuing a Start Performance Data Collection CNM RU. This signals VTAM to begin gathering the specified performance data. While the data is being gathered, the monitor can issue Collect Performance Data CNM RUs to retrieve the data. When a request to collect data is issued, VTAM copies the performance data into a parameter list and invokes the monitor's performance monitor exit in VTAM's address space. The performance monitor exit then passes a copy of the parameter list to the monitor. The response to the Collect Performance Data CNM RU is not synchronized with the exit invocation. A Stop Performance Data Collection CNM RU should be issued when the monitor no longer requires the data to be gathered.

Categorization of data

The data available through the performance monitor interface is organized into major categories and subcategories as shown in Table 87. Typically, collection requests are made at the major category level, with the option of specifying multiple subcategories. Collect Performance Data CNM RUs can also combine major category requests, with some restrictions. Stop All Performance Data Collection CNM RU allows the monitor to stop all collections with one request. For virtual route data requests, specification of the resources to be controlled, or *target resource set*, is specified as follows:

- A single VR (all Transmission Priorities included)
- All virtual routes to a destination subarea
- All routes.

For RTP connection data requests, the target resource set consists of all RTPs between the host VTAM and a given CP in the network.

Table 87. Data categorization for performance monitor interface

Subcategory code	**Description**
VTAM global data	
Environment data	Data related to this VTAM's characteristics
Installation-wide data	Data related to installation-wide exits
Storage data	Data related to this VTAM's usage of storage
Session data	Data related to sessions and resource definitions
APPN directory services data	Data related to directory services search requests
APPN topology data	Data related to APPN topology and route calculations
CSM data	Data related to the use of CSM storage and CSM buffer pools
Virtual route data	
Basic route data	Data related to individual virtual routes
RTP connection data	
Basic RTP connection data	Data related to individual RTP connections
Application data	
Application data	Data related to applications recovering from node failures, doing an MNPS planned takeover, or enabled for persistence in a sysplex that supports multinode persistent sessions

Table 87. Data categorization for performance monitor interface (continued)

Subcategory code	Description
MNPS coupling facility structure data	
MNPS coupling facility structure data	Data related to the multinode persistent sessions coupling facility structure

To minimize the overhead incurred in data gathering, monitors should limit their collections to only those subcategories of interest and stop collections once they are no longer desired. However, data can *only* be retrieved while a collection is in progress.

Automatic data delivery

A monitor does not have to issue a Collect Performance Data CNM RU to receive performance data from VTAM. VTAM may deliver unrequested data to the monitor for resource inactivation events or when automatic data delivery is enabled for other events. The performance monitor can enable automatic data delivery for SMF intervals or for non-inactivation events on the Start Performance Data Collection CNM RU. Automatic data delivery can also be turned on or off for SMF intervals or non-inactivation events by issuing a switching request. See "Switching requests" on page 361 for more information.

Resource inactivation

VTAM automatically reports the following events to the performance monitor when collection is in progress for the event.

- Directory services border node data for the affected node is reported when an adjacent non-native network node is no longer in use (the CP-CP session is lost). For the exit to receive this data, the VTAM in which the exit resides must be a border node.
- Basic route data is reported when a virtual route transitions into the inactive state. To distinguish this record from block/unblock records, the virtual route state passed within the basic route data is inactive (X'01').
- RTP connection data is reported when an RTP connection is deleted. To distinguish this record, byte 8 of the RTP data vector is set to B'0100 0000'. For more information, refer to z/OS Communications Server: SNA Customization.
- Multinode persistent session data is reported when a persistence-enabled application closes its ACB in a sysplex that supports multinode persistent sessions.

Non-inactivation events

For the events described in this section, the performance monitor can enable automatic data delivery on byte 5, bit 6 of the Start Performance Data Collection CNM RU for the event.

- Virtual route blockages

  Basic route data is reported for an individual virtual route when it becomes blocked and again when it is subsequently unblocked. The flow control state passed within the basic route data distinguishes the two cases.
- HPR path switch

RTP connection data is reported when an HPR path switch occurs. To distinguish this record, byte 8 of the RTP data vector is set to B'1000 0000' or B'0010 0000'. For more information, refer to z/OS Communications Server: SNA Customization.

When an HPR path switch occurs with a CP name change, a 'resource inactivation' is delivered if the old CP name is being monitored. Collection occurs for the new CP name only if a 'start collection' has been issued for the endpoint.

- Application recovery data is reported when an application recovers on a VTAM in a sysplex that supports multinode persistent sessions.

System Management Facility (SMF) intervals

The performance monitor can automatically receive collection data at the expiration of SMF intervals by setting byte 5, bit 4 of the Start Performance Data Collection CNM RU for the event.

Potential exception conditions relating to SMF-based collections may arise that prevent VTAM from delivering the data as expected. In each case, VTAM notifies the monitor as follows:

- If one SMF interval expires before the previous interval can be processed, VTAM indicates this condition to the performance monitor exit when SMF data is finally delivered. The SMF time stamp, which could be for either interval, is also passed to the exit. Data for the missed interval is not produced.
- If VTAM abends while processing an SMF interval, notification is not given until a subsequent collection request is processed by VTAM. The monitor is notified through the performance monitor exit that VTAM can no longer provide data automatically at SMF intervals. No collection data accompanies this particular invocation.

  Upon notification, all SMF collections are disabled but VTAM continues to gather the data. Monitors can then attempt to retrieve the data in the following ways:

  – Issuing Collect Performance Data CNM RU requests for the data.
  – Re-enabling their SMF collections with Start Performance Data Collection CNM RU requests. See "Switching requests" for details.

    Note: Start Performance Data Collection CNM RUs enabling SMF collections that are outstanding at the point of notification should be reissued.

Refer to z/OS Communications Server: SNA Customization for information on how to distinguish the various reasons for exit invocation.

Switching requests

When a collection is started using a Start Performance Data Collection CNM RU, enablement indicators determine whether data should be delivered automatically for SMF intervals or non-inactivation events. A monitor can also dynamically enable or disable automatic data delivery during a collection without modifying the monitor's current set of collections. This is known as a *switching request*.

A switching request is performed by issuing a Start Performance Data Collection CNM RU with a switching indicator specified. The enablement indicators along with their corresponding switching indicators determine whether non-inactivation event notification or SMF data delivery will be enabled or disabled for the collections specified on the request. If the enablement indicator is specified on the

request, non-inactivation event notification or SMF data delivery will be enabled. Otherwise, non-inactivation event notification or SMF data delivery will be disabled. If both switching indicators are on, both switching functions are performed for the same request.

Data collection dynamics

When starting and stopping collections, a start/stop collection request can override a previous request so that the overall set of collections can be refined over time. For example, collections may be started for all resources in a major category, then be overridden by stopping collection on an individual resource. In this case, collections are in effect for all resources except the resource that is explicitly stopped.

VTAM tracks the following information for each RTP connection, virtual route, destination subarea, or VTAM Global subcategory:

- Collection enablement (whether VTAM should gather data)
- SMF enablement (whether data should be delivered at SMF intervals)
- Blocked route enablement (whether data should be delivered when the virtual route becomes blocked or unblocked).

For each VTAM Global Data subcategory:

- Collection enablement is determined by the most recent request to start or stop data collection of the subcategory.
- SMF enablement is determined by the most recent Start Performance Data Collection CNM RU specifying the subcategory.
- Blocked route enablement does not apply.

For each virtual route:

- Collection enablement is determined by the most recent request to start or stop data collection which includes the route within its target resource set.
- SMF enablement is determined by the most recent Start Performance Data Collection CNM RU (except those specifying blocked route switching only) which includes the route within its target resource set.
- Blocked route enablement is determined by the most recent Start Performance Data Collection CNM RU (except those specifying SMF switching only) which includes the route within its target resource set.

The information tracked for virtual routes is also tracked for destination subareas, except when specified on a single virtual route request. Tracking information is kept so it can be applied to any virtual route that is defined dynamically under a destination subarea. For each RTP connection:

- Collection enablement is determined by the most recent request to start or stop data collection which includes the connection within its target resource set.
- SMF enablement is determined by the most recent Start Performance Data Collection CNM RU which includes the connection within its target resource set.
- Path switch enablement is determined by the most recent Start Performance Data Collection CNM RU (except those specifying SMF switching only) which includes the connection within its target resource set.

Performance monitor interface termination

In order to terminate the performance monitor interface normally, you should perform the following termination sequence:

- Stop all collections your monitor may have in progress using the Stop All Performance Data Collection CNM RU.
- Inactivate the Performance Monitor Exit.
- Close the ACB if communication with VTAM is no longer desired.

If the monitor terminates abnormally, VTAM performs the termination sequence on behalf of the monitor.

When VTAM terminates, VTAM will drive your monitor's TPEND exit, if one is supplied, to notify the monitor to close its ACB. See “TPEND exit routine” on page 263 for more details.

Performance data types

The data available through the performance monitor interface can be one of the following general types:

- Static data

  Static data refers to data that does not change frequently. It is not of a statistical nature. Static data can be, for example, a storage address, a resource name or address, or a configuration option.
- Current usage data

  Current usage data represents the current usage of a resource. For example, it might be the number of active sessions, or number of buffers in use.
- Highwater/Low water mark data

  Highwater mark data represents the maximum value that a particular statistic reaches during a collection interval, while low water mark data represents the minimum value.

  Highwater and low water mark data maintained on an interval basis by VTAM is reset to the current value when a Collect Performance Data CNM RU specifying the reset option is processed or when an SMF interval expires.

  In some cases, the highwater mark or low water mark data is time stamped so that the monitor can determine when the mark was reached within the collection interval.
- Counter data

  Counter data is data that is increased by some amount as activity occurs. It can be used to measure the rate of activity over a time interval. Counters kept for use by the VTAM performance monitor interface are maintained as *rolling counters*. That is, the data is never reset to zero, but rather, rolls over to zero when the counter reaches its capacity.

  Note: The monitor should send a Collect Performance Data CNM RU to VTAM at the beginning of a collection to determine the initial value of the rolling counter. The initial value returned should be regarded as a baseline number to be used with data received after subsequent retrievals.

Implications of the multiple monitor environment

When multiple monitors are open concurrently, VTAM maintains only one internal value for each resettable statistic. However, each monitor must be able to maintain

the highwater and low water marks for their interval. To provide this capability, VTAM allows multiple active instances of the performance monitor exit routine. When the exit routine is invoked, all the instances are driven so that each open monitor can receive the data as it is reset.

Monitors might receive data for collections that they are not tracking. The data received for a given invocation, depends on the reason for data delivery:

- For a Collect Performance Data CNM RU, the data delivered is the specified data for which the requesting monitor has collection enabled.
- For SMF interval data, the data delivered is the union of all monitors' SMF-enabled collections.
- For VTAM event notification, data for the affected resource is delivered if at least one monitor has appropriately enabled this notification.

For more information about multiple exit support, refer to z/OS Communications Server: SNA Customization.

Request unit formats for the performance monitor interface

This section describes the general format of the performance monitor CNM request units that are supported by the performance monitor interface. Only monitor applications can issue these CNM request units.

A typical request unit begins with the CNM request unit format described in Table 70 on page 351 which contains a subtype code identifying the desired function, as follows:

- Start Performance Data Collection (X'22')

  Sent by the performance monitor to start the collection of performance data as specified in this CNM RU or to dynamically enable or disable SMF and blocked route data collection.
- Stop Performance Data Collection (X'23')

  Sent by the performance monitor to stop the collection of specified performance data that has been previously started via the Start Performance Data Collection CNM RU.
- Stop All Performance Data Collection (X'24')

  Sent by the performance monitor to stop all performance data collections that have been previously started via the Start Performance Data Collection CNM RU.
- Collect Performance Data (X'25')

  Sent by the performance monitor to retrieve performance data as specified in this CNM RU.

Following the subtype code byte are eleven reserved bytes.

For all the above RUs except Stop All Performance Data Collection (X'24'), the CNM request unit format and the reserved 11 bytes must be followed by a request packaged in the following manner:

- The request code vector must be first, as shown in Table 88 on page 365. It identifies both the categories of data and the set of resources targeted by the CNM RU. Specify one major category, one or more subcategories within that major category, and the desired resource, if applicable, that should be processed for the request.
- The request code vector may be followed by a resource data description (presented in Table 89 on page 367) that supplies the name for a resource that

the request is targeting. It is comprised of a prefix vector and an identifier vector. The identifier vector is where the resource is actually named. The prefix vector provides necessary information about the identifier format or the resource itself.

Note: The prefix vector is a reserved field.

The resource data description must only be included when applicable, as indicated by the target resource set field in the request code vector.

There is a limit of one request per CNM RU, except for Collect Performance Data (X'25') RUs, where combining is allowed so long as none of the requests require a resource data description. When more than one request follows a CNM RU, the last request must be flagged.

Table 88. Request code vector format

Byte	Bit	Value	Description
00-01			Length of request code vector (including length field)
02-03			Request flags (common)
	01-14		Reserved
	15		Last request (Collect Performance Data subtype only)
		0	Yes
		1	No
04-05			Request flags (based on CNM RU subtype code)
	0-11		Reserved
			Start Performance Data Collection
	12		SMF data enablement
		1	Enable
		0	Disable
	13		SMF switching function indicator
		1	SMF switch requested
		0	SMF switch not requested
	14		Non-inactivation event enablement
		1	Enable
		0	Disable
	15		Non-inactivation event switching function indicator
		1	Non-inactivation event switch requested
		0	Non-inactivation event switch not requested
			Collect Performance Data
	12-14		Reserved
	15		Reset requested
		1	Yes
		0	No
			Stop Performance Data Collection

Table 88. Request code vector format (continued)

Byte	Bit	Value	Description
	12-15		Reserved
06-07			Reserved
08-09			Major category code
		X'0001'	VTAM global data
		X'0002'	Virtual route data
		X'0003'	RTP connection data
		X'0004'	Application data
		X'0005'	Coupling facility structure data
10-11			Reserved
12-19			Subcategory code (based on specified major category code): **Note:** One or more valid subcategories can be specified.
			VTAM global data
	0		Environment data
	1		Installation-wide exit data
	2		Storage data
	3		Session data
	4		APPN directory services data
	5		APPN topology data
	6		CSM data
	7-63		Reserved
			Virtual route data
	0		Basic route data
	1-63		Reserved
			RTP connection data
	0		Basic RTP connection data
	1-63		Reserved
			MNPS data
	0		Basic application data
	1-63		Reserved
			MNPS coupling facility structure data
	0		Basic MNPS coupling facility data
	1-63		Reserved
20-23			Reserved
24			Target resource set (based on specified major category code):
			VTAM global data
			Reserved
			Virtual route data

Table 88. Request code vector format (continued)

Byte	Bit	Value	Description
		X'00'	Single virtual route—resource data description for single virtual route must be attached to request code vector
		X'80'	By destination subarea—resource data description for destination subarea must be attached to request code vector
		X'FF'	All routes
			RTP connection data
		X'80'	Destination CP name - resource data description for destination CP must be attached to request code vector
		X'FF'	All RTP connections (Note that this value is not valid for start performance data collection)
25-27			Reserved

Table 89. Format of all possible resource data descriptions

Byte	Bit	Value	Description
			Destination subarea: Prefix
00-01			Length of prefix including length field
02-03			Reserved
			Destination subarea: Identifier
00-01			Length of identifier including length field
02-m			Destination subarea number (EBCDIC, numeric only, leading zeroes allowed)
			Single virtual route: Prefix
00-01			Length of prefix including length field
02-03			Reserved
			Single virtual route: Identifier
00-01			Length of identifier including length field
02-03			Virtual route number (EBCDIC, numeric only, leading zero required)
04-m			Destination subarea number (EBCDIC, numeric only, leading zeroes allowed)
			Destination CP: Prefix
00-01			Length of prefix including length field
02-03			Reserved
			Destination CP: Identifier
00-01			Length of resource data including length field
02-09			Network ID (EBCDIC, left justified)
10-17			CP name (EBCDIC, left justified)

Sense codes for the performance monitor CNM RUs

This section identifies the conditions that generate a negative response to the performance monitor CNM RUs. For these conditions, the RPL RTCD/FDBK is always X'0404'.

Protocol violations

Conditions in this category cause the entire request to be rejected before the desired function is executed, without impact on subsequent processing. The following table shows the protocol violations for all the RUs collectively. Sense Codes marked with '+' do not apply to Stop All Performance Data Collection CNM RUs.

Sense code	Description	Reason(s)
X'10010000'	RU Data Error	NS Header or CNM control key not recognized because the VTAM is down level.
X'080E0000'	LU Not Authorized	Issuer of request is not known to VTAM as a monitor application.
X'10020000'	RU Length Error	One of the required support components is omitted or is not long enough.
X'10050000'+	Parameter Error	Specification error within a required request code vector: • Major category value specified is not valid • Target resource set value specified for the major category or subtype code is not valid • No valid subcategories specified for the major category
X'08060000'+	Resource Unknown	Specification error within required Resource Data Description: • Specified VR number or DSA number is not strictly numeric or is out of range. • Specified VR number is not currently defined for the destination subarea. • Network ID and CP name are not valid. • Switching request is for an unknown resource (A "start collect" has not been issued for this resource).
X'08400000'+	Procedure not allowed	Combination of request code vectors is not valid

Process exceptions

Conditions in this category only occur once the desired function begins execution. The impact of processing up to the point of detection is described for each case. The following apply to all requests:

Sense code	Description	Reason(s)	Impact
X'081C0000'	Request Not Executable	An abend occurred within VTAM while processing the request.	Unknown

The following process exceptions apply to Start Performance Data Collect CNM RUs only. Those marked with "+" are not applicable to VTAM global data requests or when any switching function is specified.

Sense code	Description	Reason(s)	Impact
X'08090027'+	Mode inconsistency —function not enabled	No virtual routes defined within the specified target set.	Collections will automatically start for subsequent dynamic VR definitions falling within the target resource set; the blocked route and SMF enablement specifications will also be propagated.
X'08120011'+	Insufficient storage	Storage shortage occurred prior to completion	Impact is as if the request were issued with both blocked route and SMF enablement specified.
X'084F0000'	Service not available	Attempt to register to ENF failed for a request that is enabling SMF collections.	None

The following process exceptions apply to Collect Performance Data CNM RUs only.

Sense code	Description	Reason(s)	Impact
X'08090027'	Mode inconsistency —function not enabled	No data was collected due to one of the following: • Collection is not in progress for any of the requested data. • No virtual routes defined within specified target resource set. • All defined virtual routes within specified target resource set are either reset, inactive, or pending active (collections might be in progress). • No RTP connection defined. For combined requests on a single RU, this condition is reported only when encountered for all Request Code Vectors.	The performance monitor exit is not driven in this case.

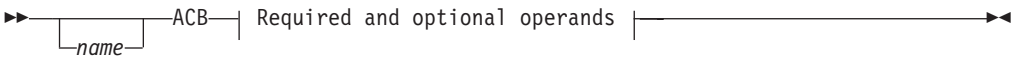
Chapter 13. Conventions and descriptions of VTAM macroinstructions

This chapter describes the format of the macroinstructions and then presents each macroinstruction in alphabetical order.

You should be familiar with the concepts of the VTAM application program described in previous chapters.

How the macroinstructions are described

The following example shows the syntax for the macroinstruction:



How the macroinstructions are coded

The macroinstructions are coded in the same format as assembler instructions, using name, operation, and operand fields.

- The optional 1–8 character *name* field provides a label for the macroinstruction. If you use a name, it must begin in column 1 and be followed by at least one blank.
- The operation identifier (such as ACB or SEND) begins in column 10 and must be followed by at least one blank.
- Operands can occupy columns 16 through 71 and must be separated by commas. The last operand on a line must be followed by at least one blank. If the operands require more than one line, you must place a non-blank *continuation character* in column 72.

© Copyright IBM Corp. 2000, 2013

```
NAME1        ACB          OPERAND1,OPERAND2,OPERAND3,      X
                          OPERAND4

|            |            |                                |
|            |            |                                |
|column  1   |column  10  |column  16                 column  72 |
```

Figure 85. How to code comments and continuation lines

Each macroinstruction has its own section that contains a brief description of its function and use. The remainder of each section contains:

- A syntax diagram
- An operand-by-operand description
- Examples (if applicable)
- A summary of status information.

Name

The macroinstruction name provides a label for the macroinstruction. The name, if used, can be specified as any symbolic name valid in the assembler language.

Operation

This field contains the mnemonic operation code of the macroinstruction. It is always coded exactly as shown.

Operands

The operands provide information for the macroinstruction expansion program in the assembler. Generally, the information provided by the operands is made part of a parameter list provided to VTAM during program execution. All of the macroinstruction's operands are indicated in the operands column of the assembler format table.

Types of operands

All operands are either keyword or positional operands. Most of the VTAM macroinstruction operands are keyword operands.

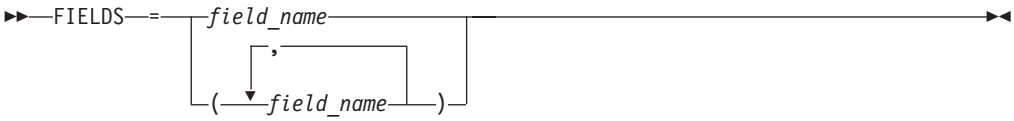
Keyword operands consist of a fixed character string (called the operand keyword), an equal sign, and a single or multiple operand value. The equal sign distinguishes the keyword from positional operands. Keyword operands do not have to be coded in the order shown in the operands column. For example, a macroinstruction having the operands `AREALEN=`*`data_length`* and `AREA=`*`data_area_address`* could be coded as either:

```
AREALEN=132,AREA=WORK
        or
AREA=WORK,AREALEN=132
```

Keyword operands must be separated by commas. If a keyword operand is omitted, the commas that would have been included with it are also omitted.

In some cases, keyword operands and their values are themselves sub-operands, such as PARMS=(USERFLD=*user_data*,NIB=*nib_address*).

There are a few instances in the VTAM macroinstructions when more than one value can be coded after the keyword, but parentheses are required to do this. For example, an operand specified as:



can be coded as either:

```
FIELDS=RECLEN
```

or

```
FIELDS=(RECLEN)
```

when only one field name is used. When more than one field name is used, however, the names must be enclosed in parentheses:

```
FIELDS=(RECLEN,RTNCD,FDB2)
```

The field names must be separated by commas. If a field name is omitted, the comma that would have been included with it is also omitted. For example, omitting the first field name from the previous example would result in:

```
FIELDS=(RTNCD,FDB2)
```

Positional operands (used in OPEN and CLOSE macroinstructions only) must be coded in the exact order shown in the operands column. Positional operands are separated by commas, as are all operands, but if a positional operand is omitted, the surrounding commas must still be entered. For example, consider a macroinstruction that has three positional operands DCB1, INOUT, and ACB1. If all three are used, they are coded as:

```
DCB1,INOUT,ACB1
```

but if only DCB1 and ACB1 are wanted, they are coded as:

```
DCB1,,ACB1
```

If the last positional operand or operands are omitted, the trailing comma or commas should not be coded.

Comments and continuation lines

Comments can contain any characters valid in the assembler language. Comments can be continued on more than one line by placing an asterisk in column 1 as shown in the following example. In this book, the comments field is not shown in the macroinstruction's assembler format table.

Operands can also be continued on additional lines as shown in the following. If the operands are not extended to column 71, they must be separated after a comma. The continuation character in column 72 can be any nonblank character, but it cannot be a character of an operand. Comments must be separated from operands by at least one blank. The following table shows where continuation characters appear; these characters are omitted from other examples in this book.

```
LABEL1      OP1          OPERAND1,OPERAND2,OPERAND3,OPERX
                         AND4,OPERAND5    THIS  IS  ONE  WAY
*
LABEL2      OP2          OPERAND1,OPERAND2, AND THIS        X
                         OPERAND3,OPERAND4  IS  ANOTHER
*                                                WAY
|           |            |                                |
|           |            |                                |
| column  1 | column  10 | column  16                 column  72 |
```

Figure 86. How to code comments and continuation lines

Operand descriptions

Following the assembler format table, each operand is named and described. Every operand description begins with an explanation of the operand's function. If the operand has more than one fixed value that can be supplied with it, the operand description also explains the effect that each value has on the action performed by the macroinstruction.

Operand format:

The operand description can include a description of the format in which the operand should be coded. This description is provided when the format is an exception to these general rules:

- When a quantity is indicated (for example, RECLEN=*data length*), you can specify the value with either unframed decimal integers, or an expression that is equated to such a value (for example, RECLEN=TEN, where TEN EQU 5*2), or the number of the register (enclosed by parentheses) that contains the value when the macroinstruction is executed. Register notation is restricted to registers 2–12 when specifying a quantity.
- When an address is indicated (for example, ACB=*acb address*) and the macroinstruction is a declarative macroinstruction, you can specify any relocatable expression that is valid for an A-type address constant. If the macroinstruction is an RPL-based or ACB-based macroinstruction, you can use any expression that is valid for an RX-type assembler instruction (such as an LA instruction). Registers 1–12 can be specified for any operand that designates the address of an RPL. Register notation for all other address operands is restricted to registers 2–12.

The valid notation for the operands of the manipulative macroinstructions (GENCB, MODCB, SHOWCB, and TESTCB) is not as straightforward. The rules of syntax for the manipulative macroinstructions are defined and tabulated in Appendix J, “Summary of operand specifications,” on page 857.

If the operand is unusually complex or if its function can be better explained with an example, the operand description can contain an example that shows how the operand is coded.

Examples

Following the operand descriptions are one or more examples. These examples show possible ways that the macroinstruction and its operands might be coded.

The way a macroinstruction can be specified can often be understood more readily from an example than it can from the assembler format table, because the latter

must show all possible ways to specify the macroinstruction. A macroinstruction that appears to be complex in the assembler format table usually appears much simpler when it is actually coded.

Completion information

All of the executable macroinstructions pass return codes in registers, and most indicate status information in various control block fields when they are posted complete. Descriptions of this status information can be found at the end of the macroinstruction description.

Description of the VTAM macroinstructions

The following sections provide detailed coding information for each VTAM macroinstruction. The macroinstruction descriptions are in alphabetical order.

Figure 7 on page 20 shows the categories of macroinstructions and identifies the macroinstructions included in each category.

ACB—Create an access method control block

Purpose

The ACB control block identifies the application program to VTAM and to the SNA network as a logical unit.

Usage

You must define each application program before the program can use VTAM to communicate with logical units throughout the network. You define an application program by coding an APPL definition statement for the program. The application program must then create an ACB that points to the same symbolic name of the program as the name specified by the APPL statement. When the ACB is opened, VTAM finds the APPL information for the program and associates that information (that is, associates the application program) with the ACB.

Each application program that uses VTAM must define and open an ACB. An application program can contain more than one ACB (thus breaking itself down into “subapplications”), but each ACB must indicate a different application program name (that is, identify a separate APPL definition statement).

After the ACB has been opened, requests for VTAM services, such as requests for session establishment, requests for communication operations, and requests for network management (if the application program is authorized), can be made. When the ACB is closed (with the CLOSE macroinstruction) such requests can no longer be made, and any sessions that were established are terminated.

Using the ACB, the application program can provide an address of a list of exit routine addresses. The routines represented in this list are invoked by VTAM when special events occur, such as error conditions or session-establishment requests.

Using the ACB, the application program can also cause VTAM to prevent or allow certain session-establishment requests that are directed to the ACB.

If this application program is an authorized communication network management (CNM) program, the ACB can identify an NIB to be used to allow access to the SSCP-LU session established when the ACB is opened.

The application program can also specify 4 bytes of user data placed in the ACB. This data is not examined by VTAM.

An ACB macroinstruction causes an ACB to be built during program assembly on a fullword boundary. The ACB can also be built during program execution with the GENCB macroinstruction. The ACB is modified during program execution with the MODCB macroinstruction, or by using the DSECT created by the IFGACB mapping macroinstruction, but only before the ACB opens. Do not modify the ACB while it is open. Not all ACB fields can be created or accessed by the manipulative macroinstructions.

The ACB control block referenced by the OPEN and CLOSE macroinstructions can reside in either 31-bit or 24-bit storage but must be consistent with the addressing mode of the application program. The MODE parameter is used to set the addressing mode of the ACB control block for these macroinstructions.

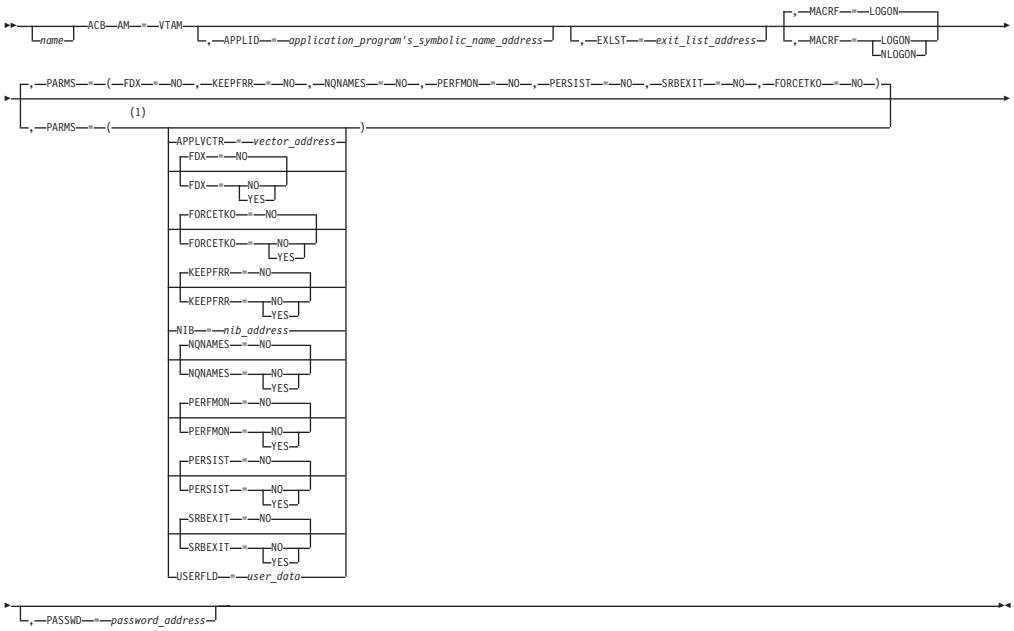
The ACB and its related storage (APPLID, password, EXLST, NIB, and Application-ACB vector list) must be allocated in the same storage key. This key can be the storage key of the program status word (PSW) at the time OPEN was issued, or the storage key of the task control block (TCB).

For each ACB opened to VTAM, there is approximately X'250' bytes of storage used as a VTAM work area. This work area is released when CLOSE ACB processing completes.

An application program can indicate its capacity to support persistent sessions by specifying (PARMS=(PERSIST=YES)). In addition, the application can indicate that the OPEN ACB associated with this ACB should be permitted to initiate MNPS forced takeover processing, if necessary, by specifying (PARMS=(FORCETKO=YES)).

For additional information about the ACB, see Chapter 4, “Opening and closing an application program,” on page 55.

Syntax



Notes:

1 You can code more than one suboperand on PARMS, but code no more than one from each group.

Input parameters

AM=VTAM
Identifies the ACB built by this macroinstruction as a VTAM ACB. This operand is required.

APPLID=*application_program's_symbolic_name_address*
During OPEN processing, links the ACB with a particular entry in VTAM's configuration tables that was created by an APPL definition statement. This both identifies the application program to VTAM and associates the application program with any options that might be indicated on the APPL statement.

If APPLID is coded, the symbolic name must match either the ACBNAME parameter (if coded) or the name field on the APPL definition statement. If ACBNAME has not been coded, APPLID must match the APPL statement's name field.

If the ACBNAME operand has been omitted, APPLID (and the name field of the APPL statement) must be a network-unique name for the application to have cross-domain sessions. If ACBNAME is coded, ACBNAME and APPLID must be unique only to their domain.

If an application program name is not specified on the APPLID operand of the ACB macroinstruction, VTAM uses a name supplied by the operating system. If the name supplied by the operating system does not match a name on the APPL definition statement, VTAM does not process the ACB.

If the application program is started by a job step that is a procedure invocation, the name is the procedure step name in the job control language (JCL) for the application program. Otherwise, it is the job step name in the JCL.

Format: Expressions involving registers cannot be used with the ACB macroinstruction.

Note: The area pointed to by this operand must begin with a 1-byte length field followed by the application program's symbolic name in EBCDIC. This is the symbolic name that appears in an APPL statement (name or ACBNAME operand), and must conform to the rules for coding this operand described in the z/OS Communications Server: SNA Resource Definition Reference. The length field specifies the length of the name. Any name that is longer than 8 bytes is truncated to 8 bytes. You can either pad the name to the right with enough blanks to form an 8-byte name (length field of 8), or you can set the length field to the actual length of the name you are providing and let VTAM do the padding. In the example at the end of this macroinstruction description, the first method is used.

The area pointed to by this operand must reside in 24-bit addressable storage.

EXLST=*exit_list_address*
Links the ACB to an exit list containing addresses of routines to be entered when certain events occur. This list is created by an EXLST (or GENCB) macroinstruction. See "EXLST—Create an exit list" on page 404 for a more complete description of this macroinstruction. You can also refer to Chapter 7, "Using exit routines," on page 219, for descriptions of these events.

More than one ACB can indicate the same exit list. For more information, see "EXLST—Create an exit list" on page 404. If no exit list is used, the application program is not notified that all the events described in the EXLST macroinstruction occurred.

The exit list identified in an ACB applies to all sessions with the application program. A separate exit list can be identified in the NIB used when the session is established to specify a DFASY, RESP, or SCIP exit routine (or any combination) to be used for that particular session. For details, refer to "Specifying the DFASY, RESP, and SCIP exit routines in an ACB or NIB" on page 229.

Format: Expressions involving registers cannot be used with the ACB macroinstruction.

MACRF
Indicates whether the application program can act (1) as the PLU in any LU-LU session other than those that it initiates itself through OPNDST OPTCD=ACQUIRE and (2) as the SLU in any LU-LU session.

An application program that is recovering for a failing application program must have the same MACRF value as the failing application program.

MACRF=LOGON
Indicates whether the application program can act (1) as the PLU in any LU-LU session other than those that it initiates itself through OPNDST OPTCD=ACQUIRE and (2) as the SLU in any LU-LU session.

MACRF=NLOGON
If NLOGON is specified, the only LU-LU sessions allowed for the application program are those initiated by the application program itself using OPNDST OPTCD=ACQUIRE. Thus, if NLOGON is specified, the application program cannot issue SIMLOGON or REQSESS, cannot have its SCIP exit routine scheduled with BIND requests sent by other LUs, and cannot receive CINITs (for instance, in a LOGON exit routine) because of sessions initiated by other LUs. NLOGON also prevents the operation of

SETLOGON OPTCD=START, STOP, and QUIESCE. See also Table 6 on page 82 for a summary of the interaction between SETLOGON and the ACB's MACRF operand.

Other application programs can interrogate for the setting of the MACRF operand by using INQUIRE OPTCD=APPSTAT. See the description of the INQUIRE macroinstruction.

PARMS=(APPLVCTR=*vector_address*)
Specifies the address of the application-ACB vector list. The application-ACB vector list contains a collection of vectors provided by the application to VTAM on the ACB macroinstruction. See "Vector lists" on page 58 for more information.

If the application is recovering after a node failure, its application-ACB vector list must be identical to that of the original application program. See "Opening the ACB during recovery from a node failure" on page 70 for more information.

PARMS=(FDX=YES|NO)
Specifies whether the application program supports LU 6.2 architecture protocol extensions for full-duplex and expedited data transmission. When starting the first session with a partner LU, the support or non-support information is used when negotiating LU-LU session support capabilities. If YES is specified, the application program handles requests for full-duplex conversations and receipt of expedited data RUs. If omitted, or if NO is specified, the full-duplex/expedited data extensions are not used on sessions with this application program.

If the application is recovering after a node failure, its support for LU 6.2 full-duplex protocols must match that of the original application. See "Opening the ACB during recovery from a node failure" on page 70 for more information.

PARMS=(FORCETKO)
Indicates whether the application wants the OPEN ACB that is associated with this ACB to trigger MNPS takeover processing when the existing application image within the sysplex is not pending some form of MNPS recovery. This parameter is valid only if PARMS=(PERSIST=YES) is also specified.

PARMS=(FORCETKO=YES)
The application wants MNPS forced takeover processing to be performed when the existing application image in the sysplex does not require MNPS recovery processing.

PARMS=(FORCETKO=NO)
The application does not require MNPS forced takeover processing to be performed. Normal OPEN ACB (which could include MNPS planned takeover processing) logic should be performed for this ACB.

PARMS=(KEEPFRR)
Specifies whether VTAM maintains the FRR stack for an application-dispatchable unit of work while VTAM processes the work.

PARMS=(KEEPFRR=YES)
VTAM does not disturb the contents of the FRR stack when VTAM API gains control. VTAM returns the FRR stack intact when VTAM returns control to the application.

PARMS=(KEEPFRR=NO)
VTAM purges the FRR stack and adds its own FRRs. VTAM removes the added FRRs before returning control to the application.

VTAM must have at least one slot on the FRR stack; otherwise, z/OS abends the SRB.

Note: In order to use the KEEPFRR function for macroinstructions issued with OPTCD=SYN under control of an SRB that is not running in cross-memory mode, use OPTCD=(SYN,KEEPSRB). For more information, see "Additional coding considerations for authorized path" on page 302.

PARMS=(NIB=*nib_address*)
Points to an NIB used to allow access to the SSCP-LU session. PARMS is initialized by a CNM application program prior to execution of the OPEN macroinstruction. This suboperand must be omitted for an application program that is not authorized for communication network management; in this case, the address field is set to 0. PARMS must specify a valid NIB address for an application program authorized for CNM. Refer to the OPEN macroinstruction ERROR field description of error codes 246 and 250 for further details. When an ACB is generated during program assembly for a CNM application program, the PARMS operand and the NIB suboperand initialize the ACBTNIB field with the NIB address.

When an ACB is generated during program execution, the GENCB macroinstruction allocates an ACB that includes the ACBTNIB field. However, manipulative macroinstructions cannot be used to initialize or modify the ACBTNIB field. Instead, the application program that dynamically initializes the ACB requires the IFGACB BAL mapping macroinstruction to map the ACB; the ACBTNIB label declared by the IFGACB macroinstruction is then used by the application program to initialize the field to the desired NIB address.

For more information about CNM application programs, see Chapter 9, "Handling errors and special conditions," on page 277.

PARMS=(NQNAMES)
Specifies whether the application program understands and is using network-qualified names.

If the application program is recovering after a node or application failure, it must have the same PARMS=(NQNAMES) value as the original application program. See "Opening the ACB during recovery from an application failure" on page 68 or "Opening the ACB during recovery from a node failure" on page 70 for more information.

PARMS=(NQNAMES=YES)
For application programs that also specify APPC=NO, and for all non LU6.2 sessions, if PARMS=(NQNAMES=YES), the application program supports network qualified names, and can use both network-qualified names and non-network-qualified names. For example, an OPNDST with a non-network-qualified name works regardless of whether the application program has specified that it supports network-qualified names.

VTAM examines the NIBNET field (the same field previously used by NIBMODE) in the ISTDNIB control block. If the contents on NIBNET are not equal to "RECORD ", and the field has not been set to blanks (X'40) or null (X'00), the value of NIBNET is used as the network identifier of the network qualified name (`netid.luname`). The luname is obtained from the field NIBSYM of ISTDNIB.

For application programs that specify APPC=YES then for all LU 6.2 sessions the following applies: the application program must provide network qualified names. That is, the PRL6NET field (NETID=) must be filled in with the network identifier of the target on all APPCCMDs that also specify RPL6LU (LUNAME=).

PARMS=(NQNAMES=NO)
The application program does not support network-qualified names and can use only non-network-qualified names. If a network identifier is provided in the NIB, it is ignored.

PARMS=(PERFMON)
Indicates whether the application is authorized to use the performance monitor interface.

PARMS=(PERFMON=YES)
The application is authorized to use the performance monitor interface.

PARMS=(PERFMON=NO)
The appplication is not authorized to use the performance monitor interface.

PARMS=(PERSIST)
Indicates whether the application can support persistent LU-LU sessions.

PARMS=(PERSIST=YES)
The application is capable of persistence, and VTAM tracks each request unit flowing on any LU-LU sessions established by the application.

PARMS=(PERSIST=NO)
The application is not capable of persistence. The latest RU information is kept.

PARMS=(SRBEXIT)
Specifies whether the application is authorized to use SRB processing in its exit routines. The SRBEXIT option enables an application to dynamically control whether SRB dispatching schedules user exits in SRB mode. SRBEXIT overrides what is coded on the APPL statement. If this operand is not specified, VTAM uses the APPL statement.

Note: If multiple applications open an ACB under one task, VTAM recognizes the SRBEXIT operand for the first application that opens the ACB. Therefore, the first open ACB determines whether exit routines operate in SRB or task control block (TCB) mode. This operand is ignored for subsequent applications opening ACBs under the same task. If you code SRBEXIT=YES on the first application opening the ACB, the subsequent applications opening ACBs under the same task must be authorized.

PARMS=(SRBEXIT=YES)
The application is authorized to use SRB processing in its exit routines.

PARMS=(SRBEXIT=NO)
The application is not authorized to use SRB processing in its exit routines.

PARMS=(USERFLD=user_data)
Can be any 4 bytes of data that the application program wants to associate with the ACB. The ACB macroinstruction stores this data in the ACBUSER field of the control block that the macroinstruction creates. The field can be referenced or changed at any time by the application program. VTAM does not examine this field.

The data can be coded in either character, fixed-point, or hexadecimal format, or, if an address is desired, it can be coded as an A-type or V-type address constant. Register notation cannot be used. If the operand is omitted, the user field is set to 0.

When an ACB is generated during program assembly, the PARMS operand and the USERFLD suboperand initialize the ACBUSER field with the user data. When an ACB is generated during program execution, the GENCB macroinstruction allocates an ACB that includes the ACBUSER field. However, manipulative macroinstructions cannot be used to initialize or modify the ACBUSER field. Instead, the application program that dynamically initializes the ACB requires the IFGACB BAL mapping macroinstruction to map the ACB; the ACBUSER label declared by the IFGACB macroinstruction is then used by the application program to initialize the field to the desired user data.

PASSWD=*password_address*
Allows an application program to associate its ACB with an APPL entry that is password protected. If a password is included in an APPL entry, any application program wanting to link its ACB to that entry must specify the entry's password in the ACB. The two passwords are compared when the application program opens the ACB. If the passwords do not match, the ACB is not opened. (The purpose of this password protection is to prevent a program from running as one of the installation's predefined application programs without the authorization of the installation.) If you omit this operand, the PASSWD address field is set to 0.

Note: The area pointed to by this operand must begin with a 1-byte length field followed by the EBCDIC password (in alphanumeric characters only). This is the password that is specified using the PRTCT operand of the APPL definition statement. It must conform to the rules for coding this operand described in the z/OS Communications Server: SNA Resource Definition Reference. The maximum length is 8 bytes. The truncation and use of the length field are the same as for the APPLID operand.

Format: Expressions involving registers cannot be used with the ACB macroinstruction.

Examples

```
ACB1     ACB    AM=VTAM,APPLID=NAME,PASSWD=PASFLD,                    C
                MACRF=LOGON,EXLST=EXLST1,                             C
                PARMS=(USERFLD=A(MYLU))
         .
         .
         .
         .
         .
         .
NAME     DC     AL1(L'NAMEF)
NAMEF    DC     C'PAYROLL'
PASFLD   DC     AL1(L'PASFLDF)
PASFLDF  DC     C'SECRET'
MYLU     DS     XL100
```

ACB1 generates an ACB that is associated with the PAYROLL APPL entry when the ACB is opened. SECRET is the password protecting that APPL entry. MACRF=LOGON means that once the application program has issued SETLOGON OPTCD=START, the LOGON and SCIP exit routines indicated in EXLST1 can be scheduled to establish sessions with the application program known as PAYROLL.

MYLU is a control block that the application program uses to keep status about itself.

Completion information

The following ACB fields are set by VTAM and can be examined by the application program during program execution.

Field name
Contents

ACBAMSVL
[3] Indicates the address of the access-method-support vector list. The vector list is located in read-only storage for the application program. This storage area is addressable from the address space in which the OPEN macroinstruction was issued. VTAM sets this field when the OPEN macroinstruction has completed successfully. For a description of the access-method-support vector list, refer to "Vector lists" on page 58.

ACBRIVL
[3] Indicates the address of the resource-information vector list. The vector list is located in read-only storage for the application program. This storage area is addressable from the address space in which the OPEN macroinstruction was issued. VTAM sets this field when the OPEN macroinstruction has completed successfully. For a description of the resource-information vector list, refer to "Vector lists" on page 58.

ERROR
Indicates why the ACB has not been opened or closed successfully. VTAM uses this field to return information to the application program upon completion of OPEN or CLOSE processing. You can use either SHOWCB or TESTCB to examine the codes in this field. The possible codes, along with their meanings, appear in the OPEN and CLOSE macroinstruction descriptions. See the description of the OPEN and CLOSE macroinstructions and Chapter 9, "Handling errors and special conditions," on page 277, for more information on this field.

OFLAGS
Indicates whether the ACB is opened or closed. VTAM uses this field to return information to the application program upon completion of OPEN or CLOSE processing. By specifying OFLAGS=OPEN on a TESTCB macroinstruction, you can determine whether the ACB is open. See the description of the OPEN and CLOSE macroinstructions and Chapter 9, "Handling errors and special conditions," on page 277, for more information on this field.

ACBPSINS
Indicates to opening applications that are capable of persistence that the opening application has taken over or recovered an application program that had enabled persistence.

3. This is a label in the IFGACB DSECT (shown in Appendix E, "Control block formats and DSECTs," on page 739), rather than a field name. An ACB operand does not exist for this field.

CHANGE—Terminate affinity between LU and generic resource application

Purpose

When VTAM establishes sessions between application programs and LUs, VTAM keeps track of the LUs that are currently in session with a generic resource application. VTAM can distinguish which applications are acting as generic resources, and is aware of the affinity between an application and any LU that has established a session with it.

The affinity between an LU and an application program is controlled by either VTAM or the application program; the controlling party *owns* the affinity. When the affinity is owned by the application program and there are no sessions with the partner LU, the CHANGE OPTCD=ENDAFFIN macroinstruction causes VTAM to *terminate* the affinity. When the affinity is owned by the application program and there are sessions with the partner LU, the CHANGE OPTCD=ENDAFFNF macroinstruction causes VTAM to terminate the affinity. See "Ownership of affinities between LUs and application programs" on page 79 for more information.

Usage

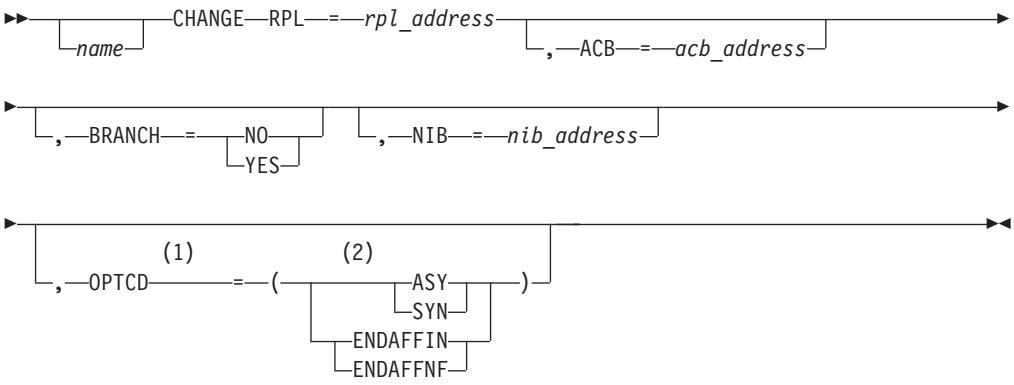
The CHANGE OPTCD=ENDAFFIN | ENDAFFNF macroinstruction uses the NIB to identify the name of the partner LU and the generic resource name.

Before issuing the CHANGE macroinstruction, the application program must perform the following steps:

- Set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853 for information about the register contents upon return of control.
- Terminate all sessions with partner LUs, unless using OPTCD=ENDAFFNF for a forced termination.

VTAM receives control from the CHANGE macroinstruction in the addressing mode of the application program that issued the macroinstruction, and returns control to the application program in that same mode.

Syntax

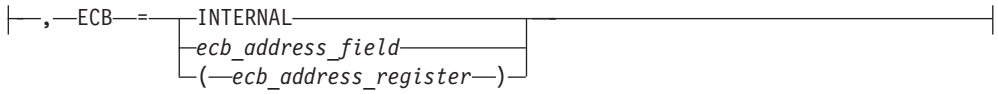


Notes:

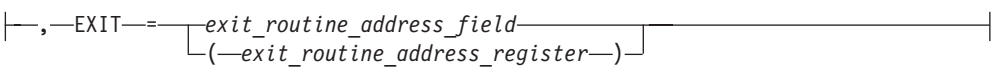
1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

ECB:



EXIT:



Input parameters

RPL=*rpl_address*
Indicates the address of the RPL that contains information to be used during the processing of this request.

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing CHANGE.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) CHANGE operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) CHANGE operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

OPTCD

OPTCD=SYN|ASY
If the SYN option code is set, control is returned to the application program when the CHANGE operation has been completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. After the CHANGE operation has been completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the CHANGE operation, you should not use the SYN option if suspending the CHANGE-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

ENDAFFIN
Indicates that the association between the issuing application and the specified network resource is to be terminated.

This OPTCD value requires specification of an NIB containing the name and network ID of the associated LU and the generic resource name under which the association was created.

ENDAFFNF
Indicates that the association between the issuing application and the specified network resource is to be terminated. ENDAFFNF will terminate the affinity even if sessions are currently active with this partner LU.

This OPTCD value requires specification of an NIB containing the name and network ID of the associated LU and the generic resource name under which the association was created.

NIB=*nib_address*
Specifies the address of a NIB whose NAME and NETID fields designate a partner LU and a GNAME= field that contains the generic name. These fields identify the association that is to be terminated.

Examples

```
        CHANGE ACB=CICS03,NIB=NIBGNRS,OPTCD=ENDAFFIN,...
⋮
NIBGNRS  NIB   NAME=NTWKLU,NETID=NETA,GNAME=CICS
```

This CHANGE macroinstruction will remove the association between an LU named NETA.NTWKLU and CICS03 using the generic name CICS.

Note: If the NAME=* is specified on the NIB, the CHANGE macroinstruction will remove all associations with the application program. This function is only intended for use by an application that is in the process of shutting down.

Completion information

The CHANGE operation is successfully complete when the LU-to-application association has been terminated.

After the CHANGE operation is completed, the following RPL fields are set:

- The value 25 (decimal) is set in the REQ field, indicating a CHANGE request.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651.

Registers 0 and 15 are also set as indicated in Chapter 9, “Handling errors and special conditions,” on page 277.

CHECK—Check request status

Purpose

When asynchronous handling is specified for an RPL-based request (ASY option code in effect), the application program receives control when VTAM accepts the

request, and the requested operation is scheduled. A CHECK macroinstruction must subsequently be issued for the RPL used for the request. CHECK cannot be issued for synchronous requests, because VTAM performs an internal operation analogous to CHECK before returning to the next sequential instruction.

Usage

The CHECK macroinstruction must be issued in an addressing mode consistent with the addressing mode of the application program at the time the original request is initiated. For example, issuing the CHECK macroinstruction in 24-bit addressing mode for a request issued in 31-bit addressing mode might have unpredictable results. The application program must ensure that the CHECK macroinstruction is issued in a consistent manner. All control blocks used by the CHECK macroinstruction must reside in 24-bit storage.

A CHECK macroinstruction can be issued if asynchronous ECB posting is used and, usually, only under the control of a TCB. A CHECK macroinstruction can be issued under an SRB only if the associated RPL has been posted complete. For example, CHECK for an RPL can be issued in the RPL user exit routine scheduled for that RPL because the RPL is posted complete before the user exit is scheduled. If CHECK is issued under an SRB before the RPL has been posted complete, the CHECK issues a WAIT, but the WAIT SVC is not allowed under an SRB. You can issue a CHECK macroinstruction in cross-memory mode against the ECB only after the request is complete and the ECB is posted. If the request is incomplete or the ECB is not posted, the user must issue the CHECK macroinstruction in non-cross-memory TCB mode.

Before issuing the CHECK macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

When CHECK executes for an RPL specifying an ECB, the following actions occur:

- If the operation checked does not complete, execution of the application program task under which the CHECK is issued suspends until the operation completes. (Asynchronous exit routines associated with this task can still run, as discussed in "Normal operating system environment for a VTAM application program" on page 30.)
- If the operation checked completes successfully (RTNCD,FDB2)=(X'00',X'00'), control returns to the next sequential instruction after CHECK.
- If the operation checked completes unsuccessfully, the LERAD or SYNAD exit routine is invoked, if available; otherwise, control returns to the next sequential instruction after CHECK.
- The ECB (internal or external) is cleared before control returns to the application program. (Clearing the ECB is necessary before an RPL-based macroinstruction using this RPL is issued.)

  Note: Do not clear the ECB specified in the RPL-based macroinstruction between the time the RPL-based macroinstruction is issued and the corresponding CHECK is issued. If the ECB is cleared during this interval, control cannot be returned to the application program after the CHECK is issued.
- The RPL checked is marked available for reuse by another request. (CHECK is the only way this can be done for asynchronous requests.)

When CHECK executes in an RPL exit routine for the associated RPL, the following actions occur:

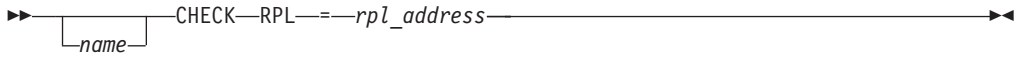
- The RPL checked is marked available for reuse by another request.
- If the operation checked completes successfully (RTNCD,FDB2) = (X'00',X'00'), control returns to the next sequential instruction after CHECK.
- If the operation checked completes unsuccessfully, the LERAD or SYNAD exit routine is invoked, if available; otherwise, control returns to the next sequential instruction after CHECK.

Notes:

1. When you use an RPL exit routine, issue the CHECK macroinstruction only in the RPL exit routine. If the CHECK is issued outside of the exit routine and the CHECK executes before the RPL exit routine is invoked, the CHECK fails with a return code of X'18' in register 0.
2. When a synchronous request is issued, VTAM uses the ECB-EXIT field to perform an internal function analogous to CHECK.

For detailed information on the use of the CHECK macroinstruction, refer to Chapter 3, "Organizing an application program," on page 33, and Chapter 4, "Opening and closing an application program," on page 55.

Syntax



Input parameters

RPL=*rpl_address*

Indicates the address of the RPL associated with the request whose completion status is being checked.

Format: Register notation (for registers 2–12) is valid.

Note: See the ECB and EXIT operands in the RPL macroinstruction description for more information about the RPL exit routine and the ECB.

Examples

```
CHK1     CHECK RPL=RPL1
```

If CHK1 is in the routine indicated by RPL1's EXIT field, and the operation requested through RPL1 ends with a logic or other error, the LERAD or SYNAD exit routine is scheduled.

If there is no RPL exit routine for RPL1, CHK1 causes program execution to stop until the operation requested through RPL1 has ended. If the operation ends with a logic or other error, CHK1 causes the LERAD or SYNAD exit routine to be invoked.

Completion information

When CHECK processing has completed, registers 0 and 15 are set as indicated in Chapter 9, "Handling errors and special conditions," on page 277. If an error occurred and an LERAD or SYNAD exit routine was invoked, these registers

contain the values set in them by the exit routine. Otherwise, VTAM places a general return code in register 15 and a recovery action return code in register 0.

CLOSE—Close one or more ACBs

Purpose

The following are the significant results of successfully executing the CLOSE macroinstruction (when persistence is not enabled):

- VTAM no longer accepts any requests that refer to the ACB specified in the CLOSE macroinstruction. This ACB is effectively disassociated from VTAM. Outstanding requests are posted complete.
- VTAM no longer maintains the association between the APPL entry known to VTAM and the ACB specified in this macroinstruction.
- VTAM terminates every session that exists between the application program logical unit (represented by the ACB) and other logical units. Before CLOSE terminates a session, all communication activity is stopped and all pending communication requests are canceled.

The preceding conditions do not apply when persistence is enabled. Instead, the following occurs:

- VTAM puts all active sessions in the recovery pending state.
- VTAM terminates all queued and pending active sessions and, if PSTIMER is set on SETLOGON OPTCD=PERSIST, VTAM starts timing. If doing MNPS, the MNPS state in the coupling facility is set to "SNPS recovery pending".

The CLOSE macroinstruction can be applied to more than one ACB. CLOSE must be issued in the mainline program. VTAM prevents attempts to issue CLOSE in an RPL exit routine or in any of the asynchronous EXLST exit routines, such as the TPEND exit routine.

For cross-memory API users, the following conditions must be met:

- CLOSE must be issued in non-cross-memory mode by mainline processing under TCB control.
- CLOSE must be issued in the address space that becomes the primary address space during a cross-memory VTAM API request.

See “Cross-memory application program interface (API) support” on page 317 for more information.

Usage

The ACB and its related storage (APPLID, password, EXLST, NIB, and application-ACB vector list) must be allocated in the same storage key. This key can be the storage key of the program status word (PSW) at the time OPEN was issued, or the storage key of the task control block (TCB).

The access-method-support and resource-information vector lists (described in “Vector lists” on page 58) must not be referenced after the CLOSE macroinstruction is issued.

For general information about the CLOSE macroinstruction, refer to Chapter 4, “Opening and closing an application program,” on page 55. For multitasking and multiple address space considerations involving the CLOSE macroinstruction, refer to Chapter 10, “Operating system facilities,” on page 295.

Control block fields referenced by the CLOSE macroinstruction can reside in either 31-bit or 24-bit storage but must be consistent with the addressing mode of the application program. The MODE parameter is used to set the addressing mode of the ACB control block for these macroinstructions.

Because further considerations apply, VTAM must issue the list or execute form of the CLOSE macroinstruction.

The standard form of the CLOSE macroinstruction expands at assembly time into (1) nonexecutable code that represents the parameters you specified on the macroinstruction and (2) executable code that causes the access method to be entered when the macroinstruction is executed. The nonexecutable code, called the parameter list, is assembled at the point in your application program where the macroinstruction appears.

List and execute forms of the CLOSE macroinstruction cause the assembler to:

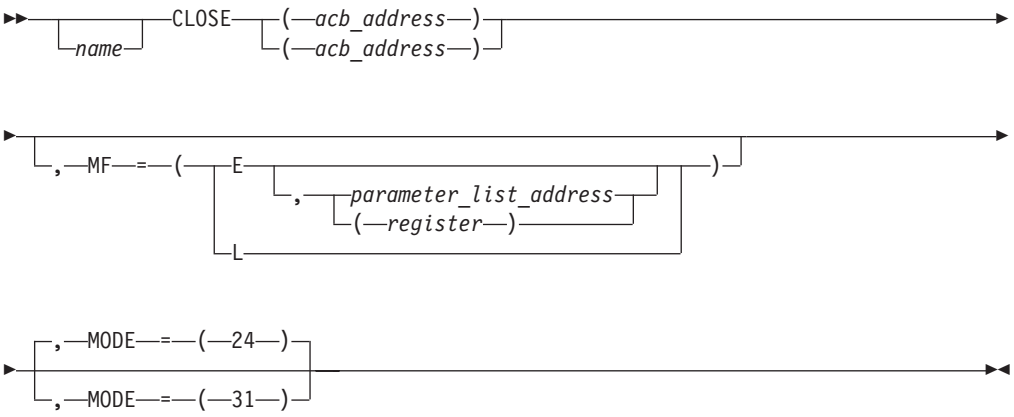
- Build the parameter list where the macroinstruction appears in your source code, but assemble no executable code (list form)
- Assemble code that modifies a parameter list and causes the access method to be entered during program execution (execute form).

Table 90 summarizes the actions of these various forms. It also indicates the types of programs that would use each form and shows the use of the MF operand. Below the table is the list and execute form of the close macroinstruction.

Table 90. Forms of the CLOSE macroinstruction

Form	During assembly	During execution	Useful for	Coded with
Standard	Parameter list built where macroinstruction appears in source code.	Access method entered.	Non-reentrant programs that are not sharing or modifying parameter lists.	No MF operand
List	Parameter list built where macroinstruction appears in source code.	No executable code (execute form required).	Non-reentrant programs that are sharing or modifying parameter lists.	MF=L
Execute	Code assembled (where macroinstruction appears in the source code) to **modify** the parameter list whose address you supply.	Parameter list modified and the access method entered.	Programs using the list form.	MF=(E,*address*)

Syntax



Input parameters

acb_address
Indicates the ACB disassociated from VTAM.

Format: If you code only one address, you can omit the parentheses. You can code up to 255 addresses. Separate each ACB address with two commas.

Note: Issue one CLOSE macroinstruction to close VSAM ACBs in addition to VTAM ACBs. Include DCBs.

MF=E
Indicates use of the execute form of the CLOSE macroinstruction and existing parameter list.

parameter_list_address
Indicates the location of the parameter list used by the access method.

The execute form allows you to modify the parameter list between the generation of that parameter list and the invocation of the access method routines that use the parameter list. Only the execute form provides a means for you to modify the parameter list after it is built.

(register)
Indicates the number of the register that will contain the parameter list address when the macroinstruction is executed.

MF=L
Indicates that the CLOSE macroinstruction creates a parameter list referred to by an execute-form instruction.

MODE
specifies the format of the CLOSE parameter list being generated.

24 specifies that a standard form (24-bit) parameter list address be generated. The parameter list must reside below 16 megabytes and point to an ACB residing below 16 megabytes.

31 specifies that a long form (31-bit) parameter list address be generated. This parameter value must be coded if the parameter list or the ACB control block resides above 16 megabytes.

Examples

```
CLOSE123 CLOSE (ACB1,,ACB2,,(7))
```

CLOSE123 closes ACB1, ACB2, and the ACB whose address is in register 7. All sessions with the application program logical units represented by these ACBs are terminated.

Completion information

When control is returned to the instruction following the CLOSE macroinstruction, register 15 indicates whether the CLOSE processing has been completed successfully.

Successful completion (meaning that all ACBs specified in the macroinstruction have been disassociated from VTAM) is indicated by a return code of 0.

Unsuccessful completion is indicated by the following register 15 values:

Value Meaning

4 (X'04')
One or more ACBs were not successfully closed. Depending on the type of error, the OFLAGS field can indicate that the ACB is closed even though the CLOSE has failed (for example, the ACB might never have been opened).

8 (X'08')
One or more ACBs were not successfully closed. Inspect the ERROR field for the cause of the failure. Another CLOSE macroinstruction can be used.

12 (X'0C')
One or more ACBs were not successfully closed. Another CLOSE macroinstruction cannot be issued.

Use the following guidelines to determine whether the CLOSE was successful:

1. Put 0 in register 15 before issuing CLOSE.
2. Issue CLOSE for only one VTAM ACB at a time.
3. If register 15 is 0, consider the CLOSE successful. If register 15 is not 0, consider the CLOSE unsuccessful, and examine the contents of the ACBs ERROR field.

If unsuccessful completion is indicated, the application program can examine the OFLAGS field in each ACB to determine which ACB was not closed. If you use the OFLAGS=OPEN operand on a TESTCB macroinstruction, an “equal” PSW condition code results if the previously opened ACB was not closed.

For each ACB, you can use either the SHOWCB or TESTCB macroinstruction to check the ERROR field and determine the cause of the error. For example:

```
SHOWCB
AM=VTAM,ACB=ACB1,FIELDS=ERROR,AREA=SHOWIT,             C
       LENGTH=4
```

Note: If the ACB address specified in the CLOSE macroinstruction does not indicate an ACB, or lies beyond the addressable range of your application program, the ERROR and OFLAGS fields in the ACB are unchanged.

The value set in the ERROR field of the ACB specified in the CLOSE macroinstruction indicates the specific nature of the error (if any) found.

ERROR field
Meaning

0 (X'00')
CLOSE successfully closed the ACB.

4 (X'04')
A CLOSE macroinstruction has been successfully issued for this ACB (or the ACB has never been opened in the first place).

20 (X'14')
CLOSE cannot be processed because of a temporary shortage of storage.

64 (X'40')
Outstanding OPNDST OPTCD=ACQUIRE is not released.

66 (X'42')
The ACB has been closed, but an apparent system error has prevented the successful termination of one or more of the sessions that the application program has. There is a logic error in VTAM; consult IBM Service. The LUs that have not had their sessions terminated are not available to other application programs, and LUs with which you were requesting a session when the CLOSE macroinstruction was issued are likewise unavailable. You can notify the VTAM operator (while the program is running) of the situation so that the operator can make the LUs available to other application programs.

70 (X'46')
CLOSE was not issued in the mainline program. OPEN and CLOSE cannot be issued in any exit routine.

76 (X'4C')
This application program is authorized to issue VTAM operator commands and receive VTAM messages. A CLOSE was issued, but messages are still queued for it, or VTAM is waiting for a reply, or both. See "Orderly closing of a program operator" on page 882 for more information.

80 (X'50')
VTAM is no longer included as part of the operating system.

96 (X'60')
An apparent system error occurred. Either there is a logic error in VTAM; or there is an error in your use of OPEN or CLOSE that VTAM did not properly detect. Save all applicable program listings and storage dumps, and consult IBM Service.

112 (X'70')
CLOSE was issued while the program was in the process of terminating abnormally. The CLOSE is not necessary because the ACB is closed by VTAM when the task terminates.

188 (X'BC')
The ACB is in the process of being opened or is in the process of being closed by another request.

CLSDST—Terminate sessions, application program is the PLU

Purpose

The CLSDST macroinstruction is used to terminate sessions in which the application program is acting as the PLU. CLSDST sends UNBIND requests from the PLU to the SLUs to terminate active sessions (sessions for which BIND has been sent). CLSDST sends TERMINATE for sessions which are queued. CLSDST also rejects CINIT requests received at the PLU for the specified SLU. The CLSDST

macroinstruction can also be used to initiate the next session the SLU has. In addition, the CLSDST macroinstruction can be used to terminate a queued session when OPTCD=TERMQ is used.

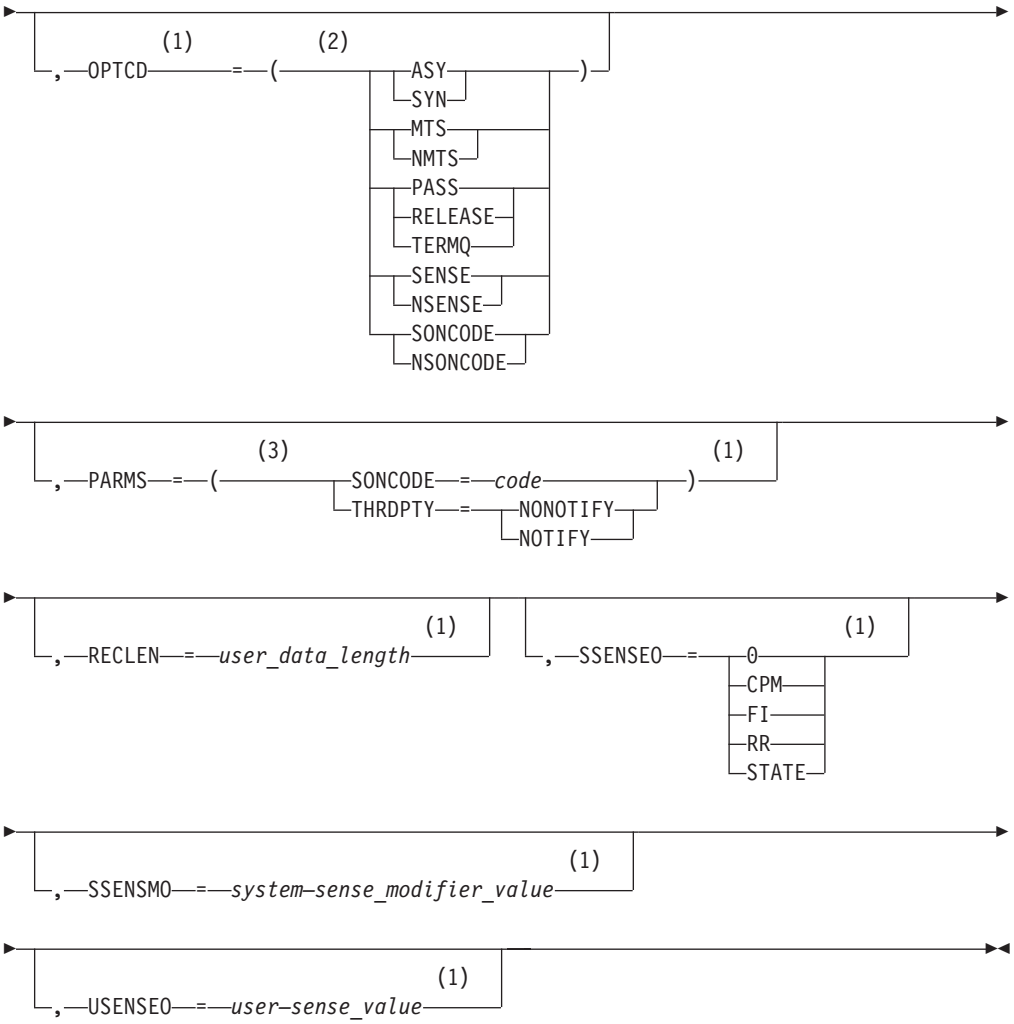
Usage

Before issuing the CLSDST macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

The CLSDST macroinstruction employs the RPL, and optionally the NIB, to identify the set of sessions to be terminated. A detailed description of the parameters and operation of the CLSDST macroinstruction is contained in Chapter 5, "Establishing and terminating sessions with logical units," on page 81.

VTAM receives control from the CLSDST macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

3 You can code more than one suboperand on PARMS, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing CLSDST is to perform.

The following RPL operands apply to the CLSDST macroinstruction:

AAREA=*target_plu's_symbolic_name_address*
Indicates the name of the target PLU of a CLSDST OPTCD=PASS. If NQNAMES=NO, the name must be 8 bytes long, padded to the right with blanks. If NQNAMES=YES, the name can be 8 bytes long, padded to the right with blanks or it can be an 8-byte long network identifier padded to the right

with blanks, followed by an 8-byte name padded with blanks to the right. The target PLU cannot be the application program that is issuing the CLSDST OPTCD=PASS.

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing CLSDST.

AREA=*user_data_address*
Indicates the location of the user data to be sent to the target PLU of CLSDST OPTCD=PASS. The contents and format of the user data are determined by the logical units. The user data is equivalent to the user data field of an Initiate request or a character-coded logon. User data is sent only if OPTCD=PASS is set.

ARECLEN=*target_plu's_name_length*
Indicates the length (in bytes) of the data contained in the area indicated by the AAREA parameter.

ARG=(register)
Indicates the register containing the CID of the session to be terminated.

Note:

1. The NIB and the ARG operands occupy the same physical field (RPLARG) in the RPL. If the last macroinstruction operand used to set or modify this field was ARG=(*register*), or if the field has been left unchanged since VTAM inserted a CID into it, VTAM recognizes that this field contains a CID. If the last operand used to set or modify this field was NIB=*address*, VTAM recognizes that the field contains an NIB address.
2. If your application uses the RPL DSECT, IFGRPL, you must clear the RPLNIB bit if a CID is being inserted into the RPLARG field.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) CLSDST operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an

asynchronous (OPTCD=ASY) CLSDST operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

NIB=*nib_address*
Indicates the NIB whose NAME field identifies the sessions to be terminated. (See "Scope of CLSDST" on page 92 for details.) If the NIB operand is not specified, the RPLARG field must contain the CID of the session.

If OPTCD=PASS is specified, the NIB optionally has a LOGMODE field which specifies the name of the session parameter set to be used in establishing a new session.

If OPTCD=TERMQ is specified, a NIB must be supplied, which identifies the NAME of the session partner, which identifies the session to be terminated.

If the application program issues CLSDST OPTCD=PASS with PARMS=(THRDPTY=NOTIFY), the USERFLD contents are passed as a correlator to the application program's NSEXIT exit routine, if it is scheduled.

Note: If your application uses the RPL DSECT, IFGRPL, you must set the RPLNIB bit if an NIB address is being inserted into the RPLARG field.

OPTCD=MTS
OPTCD=NMTS
If you code OPTCD=MTS, VTAM expects to find valid model terminal support (MTS) override data in an area pointed to by NIBMTSAR and formatted to match the ISTMTS DSECT. OPTCD=MTS is valid only in combination with OPTCD=PASS and requires the specification of the NIB operand. If you code OPTCD=NMTS, VTAM does not expect any MTS override data, and the NIB operand remains optional.

If you do not code either OPTCD=MTS or OPTCD=NMTS on this macroinstruction, VTAM uses the value left over from the previous use of the RPL.

Note: NIBMTSAR is an alternate name for the NIBNDAR field used by the OPNDST and OPNSEC macroinstruction to point to BIND image data. Therefore, do not code both MTSAREA and BNDAREA on the same macroinstruction.

OPTCD=NSENSE
OPTCD=SENSE
When a CLSDST macroinstruction is issued with OPTCD=RELEASE to reject a CINIT request, OPTCD=NSENSE or OPTCD=SENSE indicates whether values were specified with the SSENSEO, SSENSMO, and USENSEO operands. If OPTCD=NSENSE is coded, VTAM rejects the CINIT with a sense value of X'08010000'. If OPTCD=SENSE is specified, VTAM rejects the CINIT with the application-specified sense values in the SSENSEO, SSENSMO, and USENSEO fields of the RPL.

Note: Only a nonzero sense is allowed for OPTCD=SENSE. If you specify OPTCD=SENSE, and a sense code of X'00000000', CLSDST is rejected with RTNCD/FDB2=X'14',X'50' (RPL field not valid).

OPTCD=NSONCODE

OPTCD=SONCODE
If OPTCD=NSONCODE, VTAM uses an UNBIND SON code of X'01'. If OPTCD=SONCODE, VTAM uses the SON code specified in the RPL with the PARMS=(SONCODE=*code*) operand.

OPTCD=PASS
OPTCD=RELEASE
OPTCD=TERMQ
For CLSDST (PASS), two names can be network-qualified: the name of the LU whose session is to be terminated, and the name of the primary LU that the session is being passed to.

- If PARMS=(NQNAMES=NO) on the ACB macroinstruction and ARECLEN is greater than or equal to 8, the 8-byte name in AAREA is used as the target of the primary LU that the session is being passed to.
- If PARMS=(NQNAMES=YES) on the ACB macroinstuction and ARECLEN is greater than or equal to 16, the network-qualified name in AAREA is used as the target of the primary LU that the session is being passed to.
- If PARMS=(NQNAMES=YES) on the ACB macroinstuction and ARECLEN is greater than or equal to 8 but less than 16, the 8-byte name in AAREA is used as the target of the primary LU that the session is being passed to.

The format of the network-qualified name in AAREA is the 8-byte network identifier (padded with blanks, if necessary) followed by the 8-byte resource name (padded with blanks, if necessary).

If AAREA contains a network-qualified name, VTAM initiates a session between the PLU, specified as a network-qualified name in RPLAAREA, and the SLU this application program is currently in session with. The name of the SLU is specified as described under “Network-qualified names with CLSDST” on page 402.

When the session setup is completed, regardless of whether the setup is successful, the NSEXIT is scheduled, and the NOTIFY request or the NSPE request is presented to the exit. The control vectors X'0E' are removed from the NOTIFY request that is presented to the initiating application program; therefore, the NOTIFY request appears no different from the NOTIFY request in previous releases of VTAM. However, the network-qualified names are available and are pointed to by Word 7 on the parameter list passed to the NSEXIT.

The control vector X'59' for session authorization data, if present in the original CINIT, may not be passed to the initiating LU. This action is determined by settings for the SMEAUTH start option and the structure of the control vector as it is built by the session management exit. The SMEAUTH start option can override the session management exit's setting. For more information, refer to z/OS Communications Server: SNA Resource Definition Reference and z/OS Communications Server: SNA Customization.

If AAREA contains a non-network-qualified name, VTAM initiates a session between the PLU, specified as a non-network-qualified name in RPLAAREA, and the SLU this application program is currently in session with. The name of the SLU is specified as described under “Network-qualified names with CLSDST” on page 402. The name translations that occur for other non-network-qualified names (for example, for names in SIMLOGON) also occur for the name in RPLARREA.

If (PARMS=(THRDPTY=NOTIFY)) is specified, when the session setup is completed, regardless of whether the setup is successful, the NSEXIT is scheduled, and the NOTIFY request or the NSPE request is presented to the exit.

When RELEASE is set, VTAM determines the identity of the logical unit's next session partner (if any). When PASS is set, the application program determines the identity of the logical unit's next session partner; an Initiate request is sent to the SSCP to request a session between the SLU and the target PLU whose symbolic name is indicated in the AAREA field of the RPL used by CLSDST. If the AREA and RECLEN fields are also set, user data is sent to the target PLU in the CINIT request.

If OPTCD=TERMQ is coded, VTAM only terminates queued sessions (active sessions are not terminated). When TERMQ is specified, the session partner name must be used via the NIB. CID cannot be used for this option. VTAM sends a TERMINATE for this option.

OPTCD=SYN
OPTCD=ASY
If SYN option code is set, control is returned to the application program when the CLSDST operation has completed. If ASY option code is set, control is returned as soon as VTAM has accepted the request. Once the CLSDST operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the CLSDST operation, you should not use the SYN option if you are suspending the CLSDST-issuing task or if SRB for this time is undesirable. Use the ASY option code, instead.

PARMS=(SONCODE=*code*)
The application program sets the UNBIND SON code by specifying PARMS=(SONCODE=*code*) and OPTCD=SONCODE, where *code* is the 1-byte UNBIND type code used by VTAM on an UNBIND RU. See *SNA Formats* , which contains a description of the UNBIND RU, for definitions of the UNBIND type codes (SON codes). VTAM does not validate the code specified in this parameter.

If PARMS=(SONCODE=X'FE') is specified, system-sense and user-sense are set with the existing SSENSEO, SSENSMO, and USENSEO RPL fields.

PARMS=(THRDPTY=NOTIFY)
PARMS=(THRDPTY=NONOTIFY)
Indicates for CLSDST OPTCD=PASS whether the application program receives notification when the new session is established between the target PLU and the SLU (that is, when a positive response is received to the BIND for that session). If THRDPTY=NOTIFY is specified, and the session is established, the application program receives a Notify request in its NSEXIT exit routine. If the session setup fails, the application program receives an NSPE or Notify in its NSEXIT exit routine, regardless of the setting of this parameter.

RECLEN=*user_data_length*
Indicates how many bytes of user data are to be sent to the target PLU of a CLSDST OPTCD=PASS. The value in RECLEN can be no larger than 255. If RECLEN is set to 0, the AREA field is ignored and no user data is sent.

SSENSEO=0

SSENSEO=CPM
SSENSEO=FI
SSENSEO=RR
SSENSEO=STATE

.

This field can be used to provide application-specified sense values for negative responses to CINIT or for UNBIND. See the section on the TERMSESS macroinstruction for more information.

SSENSMO=*system-sense_modifier_value*
This field can be used to provide application-specified sense values for negative responses to CINIT or for UNBIND. See the TERMSESS macroinstruction for more information.

Specify any decimal integer 0–255 inclusive, or specify a 1-byte hexadecimal constant.

USENSEO=*user-sense_value*
This field can be used to provide application specified sense values for negative responses to CINIT or for UNBIND. See the TERMSESS macroinstruction for more information.

Specify any decimal integer 0–65535 inclusive, or specify a 2-byte hexadecimal or character constant.

Examples

```
CL1      CLSDST RPL=RPL1,                                          C
               ACB=ACB1,                                           C
               NIB=NIB3,                                           C
               AAREA=APPLNAME,                                     C
               AREA=LGNMSG,RECLEN=60,                              C
               ECB=POSTIT1,OPTCD=(ASY,PASS),                       C
               PARMS=(THRDPTY=NOTIFY)
         .
         .
         .
         .
         .
         .
POSTIT1  DS    F
NIB3     NIB   NAME=LU1,LOGMODE=BATCH
APPLNAME DC    CL8'PLOTTER'
LGNMSG   DC    CL60'LOGON FROM LU1'
```

CL1 terminates the session with the logical unit represented in NIB3 (LU1) after initiating a session between LU1 and the application program named PLOTTER. This macroinstruction also specifies a logon mode name (BATCH) and 60 bytes of information containing a user data field (LGNMSG). The user data field and the session parameters that VTAM derives from the logon mode name can be accessed (using INQUIRE) by the application program receiving the CINIT resulting from the Initiate before it issues OPNDST. The logon mode name (BATCH in this example) and class-of-service name are contained in the CINIT request and can be accessed by the target PLUs LOGON exit routine. The application program's NSEXIT routine is driven with a Notify request when the session between LU1 and PLOTTER has been established.

```
CL2      CLSDST RPL=RPL2,                                          C
               ARG=(3),               (SESSION TO BE TERMINATED)  C
               ECB=POSTIT2,                                        C
               OPTCD=(ASY,RELEASE,SENSE),                          C
               SSENSEO=5,SSENSMO=6                                 C
```

CL2 terminates the session whose CID has been placed in register 3. Unlike the first example, CL2 does not initiate a session with a specified PLU.

```
CL3      CLSDST RPL=RPL3,                                          C
               ACB=ACB1,                                           C
               NIB=NIB6,               (SESSION TO BE TERMINATED)  C
               AAREA=APPLNAME,         (APPLICATION TO RECEIVE CINIT C
                                        REQUEST)                   C
               RECLEN=0,               (NO USER DATA)              C
               ECB=POSTIT3,OPTCD=(ASY,PASS)
         .
         .
         .
         .
         .
         .
APPLNAME DC    CL8'PLOTTER'
POSTIT3  DC    F'0'
NIB6     NIB   NAME=LU3
```

CL3 terminates the session with the logical unit represented by NIB6 (LU3), and initiates a session between LU3 and the application program named PLOTTER. Because the RECLEN field is being set to 0, no user data field is sent to PLOTTER. The default logon mode name of 8 blanks is assumed.

Completion information

The CLSDST operation is successfully completed when either:

- The SSCP responds to the Terminate request generated by CLSDST for CLSDST OPTCD=TERMQ.
- The current session terminates for CLSDST OPTCD=RELEASE or PASS. For CLSDST OPTCD=PASS, completion does not depend on the action (accepting or rejecting the session) of the target PLU.

After the CLSDST operation is completed, the following RPL fields are set:

- The value 31 (decimal) is set in the REQ field, indicating a CLSDST request.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

If CLSDST is issued for a session that has been terminated, a return code (RTNCD,FDB2)=(0C,0B) can be posted.

If the macroinstruction returns an error code, the SSENSEI, SSENSMI, and USENSEI fields can be set indicating system-sense information, system-sense modifier, and user-sense information. See Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651 for more information about these fields.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

Network-qualified names with CLSDST

If the NIB is specified and if it contains a network identifier in ISTNIB, the network identifier is used along with the LU name in NIBSYM to create a network-qualified name that represents the name of the LU whose sessions are to be terminated. However, if APPC=YES, the sessions terminated do not include LU 6.2 sessions controlled by VTAM.

If no NIB is supplied, the RPLARG contains the CID of the one and only session to be terminated.

If the application program is specified with PARSESS=YES and a NIB is supplied, and if NIBCID in **not equal** to zero, NIBCID contains the CID of the one and only session to be terminated.

If the application program is specified with PARSESS=YES and a NIB is supplied, and if NIBCID is **equal** to zero, all sessions to the SLU or SLUs specified in NIBSYM are to be terminated. This can include sessions to multiple networks. However, if APPC=YES, the sessions terminated do not include LU 6.2 sessions controlled by VTAM.

EXECRPL—Execute a request

Purpose

When an RPL-based request fails for a temporary reason and the request might succeed if reissued, VTAM returns a recovery action return code of 8 in register 0 and in the RPL's RTNCD field. The portion of the application program receiving control (the SYNAD exit routine or the next sequential instruction) has the address of the RPL available to it in register 1. The program can issue an EXECRPL macroinstruction to retry the request without having to modify the request's RPL.

Usage

The operation performed by EXECRPL depends on the request code that is set in the RPLs REQ field. If the REQ field indicates a RECEIVE request, for example, the effect of EXECRPL is identical to that of a RECEIVE macroinstruction. (The REQ field is described in the RPL macroinstruction.) EXECRPL can be used to execute any RPL-based request except CHECK or another EXECRPL macroinstruction.

When EXECRPL is used for its intended purpose—that is, to re-execute a request that has failed with a recovery action return code of 8—the application program need not concern itself with the RPL contents when EXECRPL is issued. But if EXECRPL is used to retry a request that has failed with some other recovery action return code (or to repeat a request that has not failed at all), close attention must be paid to RPL fields that can be set by the application program and then modified by VTAM while the request is being processed.

For example, the RPL's RESPOND field for a SEND request is set by the application program (to indicate whether a response is to be returned) and is then reset by VTAM (to indicate the type of response returned). EXECRPL should not be issued for this RPL unless the RESPOND field is reset to its intended setting.

The following tables identify those fields that can be set by the application program and then reset by VTAM. These fields are identified with an “AV” under the appropriate macroinstruction.

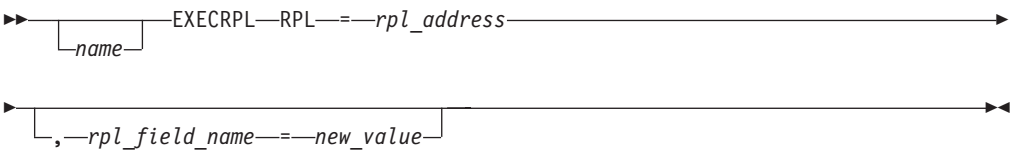
- Figure 89 on page 512 and Figure 90 on page 513
- Figure 178 on page 848, Figure 179 on page 849, and Figure 180 on page 850
- Figure 181 on page 851 and Figure 182 on page 852

We recommend that the application program establish a suitable limit on the number of times it uses the EXECRPL macroinstruction in an error recovery attempt.

Before issuing the EXECRPL macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from and returns control to the EXECRPL macroinstruction in the addressing mode of the application program that issued the macroinstruction.

Syntax



Input parameters

RPL=*rpl_address*
Indicates the location of the RPL defining the request to be reexecuted.

rpl field name=*new_value*
Indicates a field of the RPL to be modified and the new value that is contained or represented within it.

Format: For *rpl_field_name*, code the keyword of the RPL macroinstruction operand that corresponds to the RPL field to be modified. The *new_value* can be any value that could have been supplied with the keyword had the operand been issued in an RPL macroinstruction, or it can indicate a register that contains such a valid value.

Examples

```
RETRY1   EXECRPL RPL=(1)
```

A SYNAD exit routine has been entered for a retrievable error (register 0 is set to 8). The request is reexecuted as defined by the current contents of the RPL.

Completion information

After the EXECRPL operation is completed, the action taken by VTAM depends on the type of request that EXECRPL has processed. The manner in which the application program is notified of completion (ECB or EXIT), the RPL fields and return codes that are returned, and the data areas (if any) that are used depend on the contents of the RPL when EXECRPL was executed. If the request is successfully accepted, then registers 0 to 15 at the next sequential instruction after EXECRPL are set exactly as expected when the original macroinstruction was issued.

EXLST—Create an exit list

Purpose

The EXLST macroinstruction builds a list of exit routine addresses. Each operand in this macroinstruction represents a class of events for which an exit routine can be invoked by VTAM. The address supplied for each operand indicates the user-written routine to be given control when an event that it handles occurs. For

example, the SYNAD operand supplies the address of a routine that handles exception conditions (other than logic errors) for RPL-based macroinstructions, and the NSEXIT operand supplies the address of a routine that handles certain network services requests from VTAM. Refer to Chapter 7, "Using exit routines," on page 219, for detailed information about each of the exit routines that can be specified by EXLST, including the parameter lists passed to the exit routines and register usage for each exit routine.

Usage

The application program places the address of the exit list created by the EXLST macroinstruction in the EXLST field of an ACB or of an NIB. See the ACB and NIB macroinstructions for details.

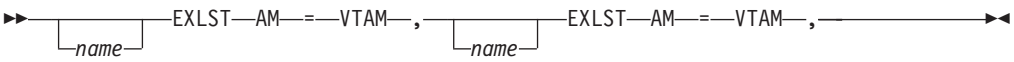
An EXLST macroinstruction causes an EXLST control block to be built during program assembly. The control block is built on a fullword boundary. The EXLST control block can also be built during program execution with the GENCB macroinstruction. The EXLST control block can be modified during program execution by using the MODCB macroinstruction or by using the DSECT created by the IFGEXLST mapping macroinstruction. After OPEN is done for an ACB that specifies an EXLST, or after OPEN (for a CNM application program), OPNDST, or OPNSEC is done for an NIB that specifies an EXLST, the exit routines defined by that EXLST cannot be modified or freed for the application program represented by the ACB or for the session represented by the NIB.

When you examine your program listing, you might discover that the assembler has reserved space for exit list addresses that you never specified. Unspecified exits are not, however, used by VTAM, and you cannot use MODCB to insert an address in a field you never specified in the EXLST (or GENCB) macroinstruction. An address of 0 can never be specified.

Note: Only exit routines that VTAM recognizes can be specified with the VTAM EXLST macroinstruction. For example, VSAM exit routines are not allowed.

The expansion of the EXLST macroinstruction is identical for 24- and 31-bit addressing mode application programs.

Syntax



Input parameters

AM=VTAM
Identifies the exit list generated by this macroinstruction as a VTAM exit list (as distinguished from a VSAM exit list). This operand is required.

ATTN=*exit_routine_address*
Indicates the address of a routine to be entered to notify a VTAM LU 6.2 application program of the occurrence of the following LU 6.2 related events:

- Session limit changes
- Incoming conversation requests
- Session deactivation.

Refer to the z/OS Communications Server: SNA Programmer's LU 6.2 Guide for information on the use of the ATTN exit routine.

DFASY=*exit_routine_address*
Indicates the address of a routine to be entered when an expedited-flow data-flow-control request is received. Data-flow-control requests received by the primary end of the session are SBI, QEC, RELQ, SIG, RSHUTD, and SHUTDC. Data-flow-control requests received by the secondary end of the session are SBI, QEC, RELQ, SIGNAL, and SHUTD.

The EXLST containing a DFASY exit routine address can be pointed to by an NIB, as well as by an ACB (see the EXLST operand of the NIB macroinstruction).

LERAD=*exit_routine_address*
Indicates the address of a routine to be entered when the application program makes an RPL-based request that results in a logic error. Before the LERAD exit routine is given control, VTAM sets a recovery action return code. These codes are explained in Chapter 9, “Handling errors and special conditions,” on page 277. The LERAD exit routine is entered for all recovery action return codes of 20 and 24 (decimal).

LOGON=*exit_routine_address*
Indicates the address of a routine to be entered when a CINIT request is sent to the application program as a result of a session-initiation request (such as a logon from a device-type logical unit).

LOSTERM=*exit_routine_address*
Indicates the address of a routine to be entered when, for example, a logical unit has requested that a session be terminated, or when there are not enough buffers available to queue data that has been received.

NSEXIT=*exit_routine_address*
Indicates the address of a routine to be entered when the application program receives certain network services request units.

RELREQ=*exit_routine_address*
Indicates the address of a routine to be entered when another application program requests a session with a device-type logical unit that is currently in session with your application program. This can occur when the other application program issues a SIMLOGON macroinstruction (with the RELRQ and Q options specified) for the device-type logical unit.

RESP=*exit_routine_address*
Indicates the address of a routine to be entered when certain normal-flow responses arrive on a session.

The EXLST containing the RESP exit routine address can be pointed to by an NIB, as well as by an ACB (see the EXLST operand of the NIB macroinstruction).

SCIP=*exit_routine_address*
Indicates the address of a routine to be entered when a BIND, UNBIND, Clear, STSN, SDT, or RQR request is received by an application program.

The EXLST containing the SCIP exit routine address can be pointed to by an NIB, as well as by an ACB (see the EXLST operand of the NIB macroinstruction).

Note: A BIND request does not cause an NIB-specified SCIP exit routine to be scheduled. Any application program that is to receive a BIND request (that is, acts as an SLU application program) must, therefore, have a SCIP exit routine defined in the EXLST associated with the program's ACB.

SYNAD=*exit_routine_address*
Indicates the address of a routine to be entered if an unrecoverable input or output error (physical error) or other unusual condition occurs during the processing of an RPL-based request. (Errors that result from requests that are not valid are handled by the LERAD exit routine.) Before the SYNAD exit routine is given control, VTAM sets a recovery action return code. These codes are explained in Chapter 9, “Handling errors and special conditions,” on page 277. The SYNAD exit routine is entered for all recovery action return codes of 4, 8, 12, and 16 (decimal).

TPEND=*exit_routine_address*
Indicates the address of a routine to be entered when any of the following occur:

- A VTAM operator issues a HALT command (or VARY INACT command for the application program).
- VTAM detects an internal problem that necessitates halting itself.
- VTAM abnormally ends
- An alternate application takes over sessions from an application that has enabled persistence.

GENCB—Generate a control block

Purpose

The GENCB macroinstruction builds an ACB, EXLST, RPL, or NIB. The advantage of using the GENCB macroinstruction is that the control blocks are generated during program execution. (With the ACB, EXLST, RPL, and NIB macroinstructions, the control blocks are built during program assembly.)

GENCB not only builds the control block during program execution, but can also build the control block in dynamically allocated storage. One advantage of this technique is that it can remove application program dependencies on the length of each control block.

Usage

The GENCB user specifies the type of control block to be built and the contents of some of its fields. The operands used to specify the field contents are exactly the same as those used in the declarative macroinstruction that builds the control block during assembly. For example, these macroinstructions build the same exit list:

```
GENCB BLK=EXLST,SYNAD=SYNADPGM,AM=VTAM
EXLST SYNAD=SYNADPGM,AM=VTAM
```

To build the control block in the application program's storage, the application program should either reserve enough storage during program assembly to accommodate the control block or issue an operating system storage manipulation macroinstruction to get the necessary storage. If an operating system macroinstruction is used, the location and length of this storage must be coded in the GENCB macroinstruction. Dynamic storage allocation for the control block occurs automatically if the location and length operands (WAREA and LENGTH) are omitted.

To free the storage obtained by the GENCB macro, the application program can issue the appropriate operating system instruction.

Ordinarily, the storage should be returned to subpool 0. However, if GENCB is issued in a task running in privileged state, return the storage to subpool 252.

Dynamic storage allocation can be successful only if (1) the program is operating in virtual mode and (2) enough unallocated virtual storage remains in the program's partition or address space to build the control block. See the description of the LENGTH operand for an explanation of how control block lengths are determined.

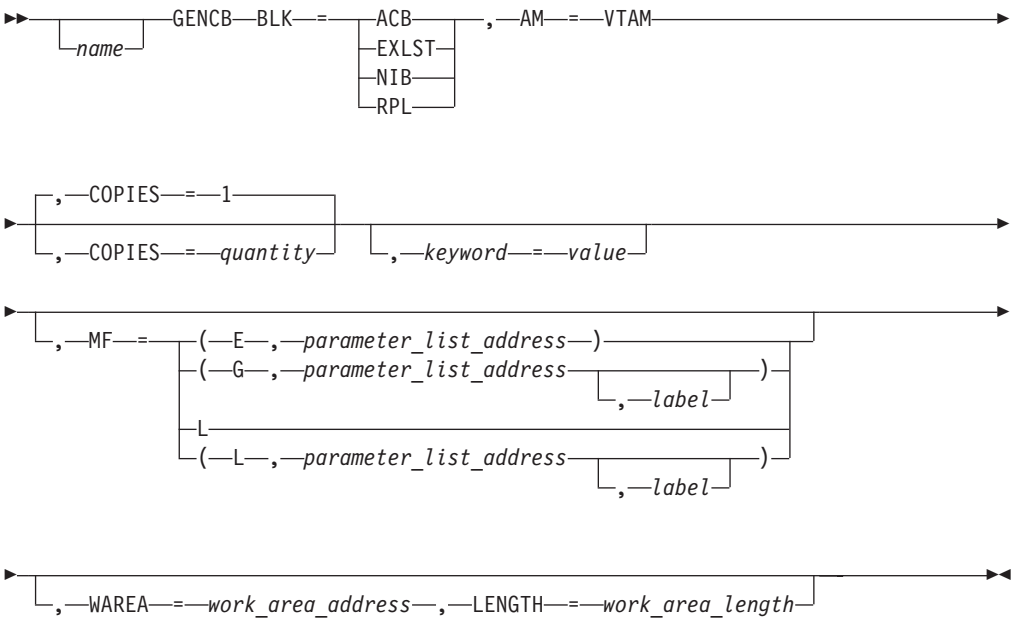
List, generate, and execute forms of the GENCB macroinstruction are available; they are designated by the MF operand. (See Appendix K, “Forms of the manipulative macroinstruction,” on page 865, for more information.) These forms must be used if the invoking routine is reentrant.

Because the GENCB operands can be specified in a large variety of formats, format specifications have been tabulated in Appendix J, “Summary of operand specifications,” on page 857, and do not appear in this macroinstruction description.

Note: Not all control block fields can be initialized by using GENCB. See Appendix J, “Summary of operand specifications,” on page 857, for those supported. You can initialize other fields at execution time by using the appropriate control block DSECTs described in Appendix E, “Control block formats and DSECTs,” on page 739.

The GENCB macroinstruction can be issued by an application program running in either 24- or 31-bit addressing mode. Refer to “31-bit addressing” on page 319 for information on 31-bit addressing. To use 31-bit addressing, the application must use the VTAM mapping macroinstructions as well as GETMAIN and FREEMAIN.

Syntax



Input parameters

AM=VTAM
Identifies this macroinstruction as a VTAM macroinstruction. This operand is required.

BLK
Indicates the type of control block to be generated.

BLK=ACB
An ACB control block is to be generated.

BLK=EXLST
An EXLST control block is to be generated.

BLK=NIB
An NIB control block is to be generated.

BLK=RPL
An RPL control block is to be generated.

COPIES=quantity
Indicates the number of control blocks to be generated.

The copies are identical in form and contents. They are placed contiguously in storage, whether that storage is the area indicated by the WAREA operand or in dynamically allocated storage.

The length returned in register 0 is the total length of the generated control blocks. The length of each block (the total length divided by the number of copies) can be used to determine the location of the beginning of each block.

Note: If this operand is not used, one control block is built.

keyword=value
Indicates a control block field and the value that is to be contained or represented within it.

For *keyword*, code a GENCB-supported keyword (see Appendix J, "Summary of operand specifications," on page 857) that can be used in the macroinstruction corresponding to the BLK operand. If BLK=ACB is used, for example, code the keyword of any GENCB-supported operand that can be used in the ACB macroinstruction. There is one exception: ARG=(*register*) can also be coded if BLK=RPL.

For *value*, indicate a register, or code any value that could be used if the operand were being specified in the ACB, EXLST, RPL, or NIB macroinstruction, or use one of the formats indicated in Appendix J, "Summary of operand specifications," on page 857.

Note: If no keywords are included, the following types of control blocks are built.

- ACB: All fields are set to their default values as indicated in the ACB macroinstruction description, except MACRF which is set to NLOGON.
- RPL: All fields are set to their default values as indicated in the RPL macroinstruction description.
- EXLST: All fields are set to 0.
- NIB: All fields are set to their default values as indicated in the NIB macroinstruction description.

LENGTH=*work_area_length*

Indicates the length (in bytes) of the storage area designated by the WAREA operand. If this length is insufficient, register 15 contains the value 4, and register 0 contains the value 9.

Warning: To avoid having to recode your application program should you wish to run it under a different operating system or a different release of VTAM, use the manipulative macroinstructions to obtain the control block lengths. You do this by specifying ACBLEN, EXLLEN, RPLLEN, or NIBLEN in either a SHOWCB or TESTCB macroinstruction.

GENCB fails if LENGTH specifies a smaller value than the length of the control block to be generated. For example, if you use the IFGACB DSECT to determine the length of the ACB, and if your program is assembled on one release of VTAM but is executed on a subsequent release in which the ACB is larger, the GENCB fails.

To obtain the length of an ACB, the following SHOWCB could be coded:

```
SHOWCB FIELDS=ACBLEN,AREA=WORKAREA,LENGTH=4,AM=VTAM
```

To test the length of an exit list in your particular operating system, the following TESTCB could be coded:

```
TESTCB EXLLEN=(7),AM=VTAM
```

If you are generating more than one control block, remember that the total length of each control block is the length indicated by the control block's length field (ACBLEN, EXLLEN, RPLLEN, NIBLEN) plus the number of bytes required for fullword alignment. (EXLSTs are variable in length; when no specific EXLST is specified, the length returned by SHOWCB or tested by TESTCB is the maximum possible length for your operating system.)

MF=E, G, or L

Indicates that an execute, generate, or list form of GENCB is to be used. Omitting this operand causes the standard form of GENCB to be used. See Appendix K, "Forms of the manipulative macroinstruction," on page 865, for a description of the execute, generate, and list forms of GENCB.

WAREA=*work_area_address*

Indicates the location of the storage area in the application program where the control block is to be built. The work area must be aligned on a fullword boundary. If this operand is specified, the LENGTH operand must also be specified.

If the WAREA and LENGTH operands are omitted, VTAM obtains dynamically allocated storage from the operating system and builds the control block there. Assuming that GENCB is completed successfully (this is indicated by a return code of 0 in register 15), the address of the generated control block (or blocks) is placed in register 1, and their total length is placed in register 0.

Examples

```
GEN1     GENCB  AM=VTAM,BLK=ACB,                                    C
               APPLID=(3),EXLST=(6),                               C
               WAREA=BLOKPOOL,LENGTH=(4)
         .
         .
         .
BLOKPOOL DS    32D
```

GEN1 builds an ACB in statically reserved storage (BLOKPOOL). When GEN1 is executed, register 3 must contain the address of an application program's symbolic name, and register 6 must contain the address of the exit list to be pointed to by the ACB.

```
         L     10,WORKAREA         (REG10=ACB LENGTH)
         GETMAIN R,LV=(10)
         LR    5,1                 (REG5=ACB ADDRESS)
GEN2     GENCB  AM=VTAM,BLK=ACB,                                   C
               WAREA=(5),LENGTH=(10)
```

In this example, the application program is building an ACB in dynamically allocated storage obtained by itself. Using the procedure described in the preceding LENGTH operand description, the application program has obtained the length of an ACB and placed it in a fullword called WORKAREA. The instructions preceding GEN2 obtain the correct amount of storage, and GEN2 builds the ACB in that storage.

```
GEN3     GENCB BLK=RPL,COPIES=10,AM=VTAM
```

GEN3 creates 10 RPLs in dynamically allocated storage. The address of the beginning of these RPLs is returned in register 1, and the total length is returned in register 0. This length includes all padding for fullword alignment; the RPLLEN field indicates the length of each unpadded RPL. Each RPL is built as though an RPL macroinstruction with no operands had been issued.

Completion information

After GENCB processing is finished and control is returned to the application program, register 15 indicates whether the operation is completed successfully. If the operation is completed successfully, register 15 is set to 0. If it is completed unsuccessfully, register 15 is set to either X'04', X'08', or X'0C'. If register 15 is set to X'04' or X'0C', register 0 is also set indicating the specific nature of the error (see Appendix I, “Return codes for manipulative macroinstructions,” on page 855). After successful completion, register 1 has the address of the generated control blocks, and register 0 has their total length (including padding to a fullword boundary).

INQUIRE—Obtain logical unit information or application program status

Usage

Several types of INQUIRE exist. The setting of the RPL's option code determines which one is used.

The following descriptions indicate the purpose and use of these options; see the operand descriptions for details regarding how each is specified. For restrictions on the use of the operands, see Table 91. Also, for large networks, see INQUIRE OPTCD=APPSTAT on page 413.

Table 91. Permissible option codes in the INQUIRE macroinstruction

OPTCD= keyword	Same-domain request	Cross-domain request	Issued by primary	Issued by secondary
APPSTAT (Note 3)	YES	YES	YES	YES
CIDXLATE	YES	YES	YES	YES
COUNTS	YES	YES	YES	YES
DEVCHAR	YES	YES (Note 1)	YES	YES
LOGONMSG (Note 1)	YES	YES	YES	NO (Note 2)
NQN (Note 3)	YES	YES	YES	YES
PERSESS	YES	YES	YES	YES
SESSKEY (Note 4)	YES	YES	YES	YES
SESSNAME	YES	YES	YES	YES
SESSPARM (Note 1)	YES	YES	YES	NO
STATUS	YES	YES	YES	YES
TERMS	YES	YES	YES	YES
TOPLOGON (Note 1)	YES	YES	YES	NO (Note 2)
USERVAR	YES	NO	YES	YES

The following notes refer to the preceeding table.

Notes:

1. Application program can issue only after the CINIT is queued and before the OPNDST or CLSDST macroinstruction is issued to accept or reject the pending active session.
2. Application program can issue, but VTAM can return (RTNCD,FDB2)=(X'00',X'07') indicating that the requested information is not available.
3. These operands are the only options that could cause an external RU to flow into the network.
4. Application program can issue only after the session is established. If the session is not established, VTAM can return (RTNCD,FDB2)=(X'00',X'07') indicating that the requested information is not available.

Before issuing the INQUIRE macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from the INQUIRE macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

Note: The INQUIRE macroinstruction for OPTCD=NQN differs from other VTAM macroinstructions in that whether a network-qualified name is presented or not is not always dependent on the value specified for PARMS=(NQNAMES=YES) on the ACB macroinstruction. Therefore, an application program can use this type of INQUIRE to translate names without having to change the entire application program to use network-qualified names. Translation is performed on this type of INQUIRE macroinstruction by examining the NIBNET field in the NIB. If NIBNET is **not** zeros, blanks, or "RECORD ", the network identifier of the NIB is used in the INQUIRE macroinstruction (real-to-symbolic translation).

APPSTAT
: INQUIRE checks a specified application program and determines its capability to establish sessions. In addition, if AREA and AREALEN are included, the network-qualified name of the specified application program is returned. This option can be used to determine the name of the VTAM application currently associated with a USERVAR.

CIDXLATE
: Given a session CID, INQUIRE provides the symbolic name of the logical unit that has this session with the application program. Conversely, given the symbolic name of a logical unit with which this application program has one or more sessions, INQUIRE provides the CID of a session with that logical unit.

COUNTS
: For the ACB specified, INQUIRE provides the number of active sessions and the number of queued CINIT requests for the application program.

DEVCHAR
: INQUIRE obtains certain predefined device characteristics of a logical unit. In general, however, session parameter information (obtained, for example, by INQUIRE OPTCD=SESSPARM) should be used to determine the way to operate a session with the logical unit.

LOGONMSG
: INQUIRE obtains the user data (logon message) portion of a queued CINIT that resulted from an initiation of a session between a logical unit and this application program. The user data was part of the original session-initiation request (such as a logon).

NQN Provides name translation for either a symbolic-to-real or real-to-symbolic request.

PERSESS
: INQUIRE enables VTAM to create an NIB for each session pending recovery. The NIB provides the information needed to restore the individual session.

SESSKEY
: This provides the session cryptography key and the initial chaining value of a session.

SESSNAME
Allows you to determine the network qualified name of the real instance that a specified LU is currently in session with.

SESSPARM
INQUIRE obtains either a set of session parameters from the logon mode table associated with the specified logical unit, or obtains the session parameter from a CINIT queued for the application program.

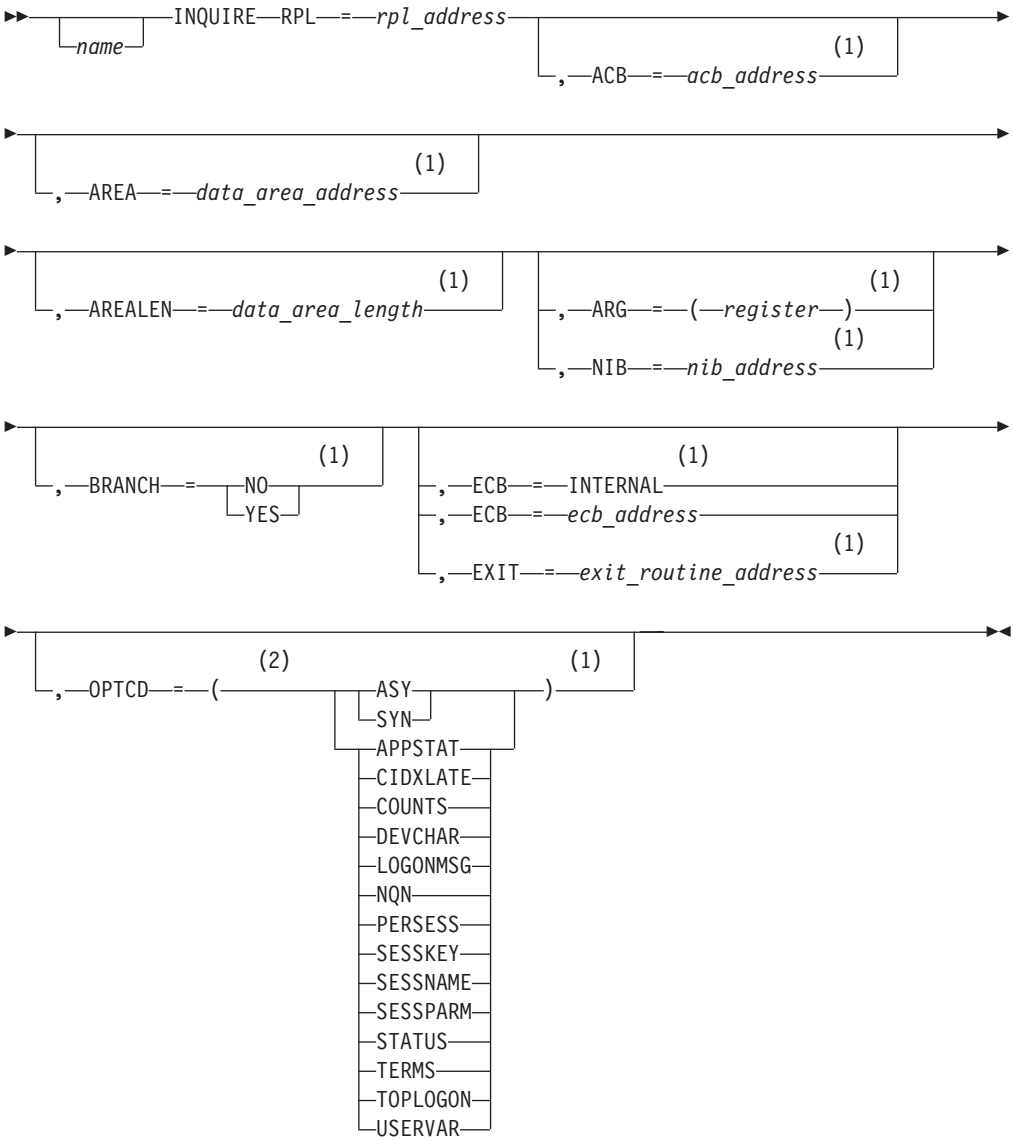
STATUS
INQUIRE returns the status of an LU or an application program; it checks the LU or application and determines its capability to establish sessions. In addition, if AREA and AREALEN are included, it returns the network-qualified name of the LU or application program.

TERMS
For a given resource known to VTAM, INQUIRE builds an NIB or list of NIBs for associated logical units. Examples of resources that can be specified are: logical units, physical units, lines, local terminals, application programs, resources in other domains, NCPs, non-SNA major nodes, channel-attached major nodes, switched major nodes, and CDRSC major nodes. Refer to the z/OS Communications Server: SNA Resource Definition Reference for more information on these resources. During VTAM definition, the user can define a resource (such as a PU) that has one or more logical units subordinate to it. These definitions create entries for the resource and the dependent LUs that are subordinate to the resource that is defined to VTAM. If the application program builds one NIB that indicates such an entry in its NAME field, it can then issue INQUIRE to generate NIBs for all of the logical units associated with the entry. Thus, the application program need not be aware of the identities or the number of these logical units before establishing sessions with them. This allows the user, through the VTAM operator or VTAM definition procedures, to change the set of logical units after the application program has been assembled. For additional information, refer to "Using INQUIRE OPTCD=TERMS to generate NIBs" on page 271.

TOPLOGON
Returns the name of the session partner and the CID for the pending active session represented by the oldest queued CINIT received by the application program.

USERVAR
Although USERVAR is allowed to be issued , the actual macroinstruction invocation always returns a successful RTNCD,FDB2 and returns the translated name as the same name passed as input. OPTCD=NQN is the suggested option to be used for name translation requests.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing INQUIRE is to perform.

The following RPL operands apply to the INQUIRE macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing INQUIRE.

AREA=*data_area_address*
Indicates where the information produced by INQUIRE is to be placed.

The AREA operand must be coded with each INQUIRE macroinstruction and must contain a valid storage pointer regardless of OPTCD except for OPTCD=APPSTAT.

AREALEN=*data_area_length*
Indicates the maximum number of bytes of data that the data area can hold; if the data to be placed there exceeds this value, a special condition results, (RTNCD,FDB2)=(X'00',X'05'), and the RECLEN field indicates the required length (except when OPTCD=APPSTAT). INQUIRE can be reissued using the correct length.

With OPTCD=APPSTAT, AREALEN must be defined to be at least 16 bytes to hold the network-qualified name.

ARG=(register)
Indicates the register containing the CID of the session. Register notation must be used if the CID is to be placed in the ARG field with this INQUIRE macroinstruction. This operand applies to the DEVCHAR, CIDXLATE, SESSPARM, LOGONMSG, and SESSKEY forms of INQUIRE.

Note:

1. The NIB and the ARG operands occupy the same physical field (RPLARG) in the RPL. If the last macroinstruction operand used to set or modify this field was ARG=(*register*), or if the field has been left unchanged since VTAM inserted a CID into it, VTAM recognizes that this field contains a CID. If the last operand used to set or modify this field was NIB=*address*, VTAM recognizes that the field contains an NIB address.
2. If your application uses the RPL DSECT, IFGRPL, you must clear the RPLNIB bit if a CID is being inserted into the RPLARG field.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) INQUIRE operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an

asynchronous (OPTCD=ASY) INQUIRE operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

NIB=*nib_address*
Indicates the NIB whose NAME field identifies the session or logical unit. This operand applies to the LOGONMSG, DEVCHAR, TERMS, APPSTAT, SESSPARM, SESSKEY, and CIDXLATE forms of INQUIRE. See the OPTCD descriptions of these forms of INQUIRE for information about whether the NAME (NIBSYM) or CID fields in the NIB can be used. For DEVCHAR, LOGONMSG, SESSPARM, SESSKEY, and CIDXLATE, NIB=*address* and ARG=(*register*) are mutually exclusive methods of identifying the logical unit, or specific session.

Note: If your application uses the RPL DSECT, IFGRPL, you must set the RPLNIB if an NIB address is being inserted into the RPLARG field.

OPTCD
See Table 91 on page 412 for possible restrictions on the use of the OPTCD operand. The following describes each of the possible parameters to OPTCD. Only one of the following options can be specified.

OPTCD=APPSTAT
INQUIRE checks the status of a given application program and returns a value in the RPL's FDBK field which gives information about that application program's capability to establish sessions. INQUIRE can also return information about the application program's capability to establish cross-network sessions. Refer to the SETLOGON macroinstruction for further information.

The following are the RPL FDBK flag values indicating the status of the specified application program.

RPLFDB3 value
Explanation

X'00' The application program is active. The application program has opened its ACB with MACRF=LOGON and has issued SETLOGON OPTCD=START. It has not subsequently issued SETLOGON OPCTD=QUIESCE or SETLOGON OPCTD=STOP. Therefore, the application program can act as the PLU in a session.

X'04' The application program is inactive. If this is a same-domain request, the application program's ACB is not open. If this is a cross-domain request, the application program's ACB is not open, or an inactive CDRSC, representing the application, was found in the host of the program that issued the INQUIRE request. No sessions can be established with the application program.

X'08' The application program has opened its ACB with MACRF=NLOGON. Therefore, it does not accept sessions that it did not itself initiate by OPNDST OPTCD=ACQUIRE, and it cannot act as the SLU in a session.

X'0C' The application program has opened its ACB with MACRF=LOGON and has not yet issued SETLOGON OPTCD=START or has issued a subsequent SETLOGON OPCTD=STOP. If SETLOGON OPCTD=START has not yet been issued, the application program is not yet enabled for sessions in which it acts as the SLU and attempts to initiate such sessions are rejected.

If SETLOGON OPCTD=STOP is issued, the program requests to temporarily stop establishing sessions. This indicator is by convention only; that is, the SSCP does not take any action to prevent the initiation of sessions with this application program just because this indicator is set. The effect of SETLOGON OPTCD=STOP can be reversed by issuing SETLOGON OPTCD=START.

X'10' VTAM returns this code in two cases. The application either issued SETLOGON OPTCD=QUIESCE, or the application is pending recovery. In either case, no new sessions can be established. To reverse SETLOGON OPTCD=QUIESCE, the application program closes and reopens the ACB.

To leave the recovery pending state, the application program issues OPEN ACB.

RPLRTNCD/ RPLFDB2
Explanation

X'000A'
Application program not connectable

X'144C'
Search argument for INQUIRE or INTRPRET not valid

X'1453'
Logical unit not found

Note: For more information on RPL fields, see Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651, or Appendix E, "Control block formats and DSECTs," on page 739.

The RPL's ACB field must contain the address of an opened ACB. This is the ACB of the application program issuing the INQUIRE.

The RPL's NIB field must point to a NIB whose NAME field contains the symbolic name of the application program whose status is desired.

If PARMS=(NQNAMES=YES), and if the NIB is specified and it contains a network identifier in the NIBNET field, the network identifier is used along with the LU name in NIBSYM to find the application.

If AREA and AREALEN are specified with OPTCD=APPSTAT, VTAM will attempt to give the network-qualified name (network identifier and real name) for the specified application program. If a USERVAR or ALIAS is given as input, the name of the application associated with the USERVAR will be returned. The network-qualified name will be placed in RPLAREA. RPLRLEN will contain the number of bytes placed in RPLAREA.

Note: The preferred function for symbolic-to-real name translation is INQUIRE OPTCD=NQN.

When VTAM is able to process the request successfully, RPLAREA will contain the network identifier followed by the real name of the application. The netid will be in the first 8 bytes, followed by the real name in the second 8 bytes. Blanks (X'40') are appended to the right of the network identifier or name, if needed, to guarantee the 8-byte length of each field. AREA/AREALEN should be defined to hold at least 16 bytes of data.

The network-qualified name is not returned when AREA and AREALEN are included with INQUIRE OPTCD=APPSTAT in the following cases:

- The work area designated by AREA is not large enough (AREALEN must be at least 16 bytes to hold the network identifier and real name)
- An error was encountered while processing INQUIRE (RPLRTNCD=X'00')
- The named resource is an inactive CDRSC and the real network identifier or name is not available
- The named resource was found but is not an application.

In these cases, RPLAREA will not contain the network-qualified name. In addition, RTNCD/FBD2 will not report any new or additional error codes. If RPLRTNCD=X'00', the RPL will still contain information about the application's status. The INQUIRE will complete as if AREA had never been specified as part of the invocation.

OPTCD=ASY

Control is returned to the application program as soon as VTAM has accepted the INQUIRE operation request. Once the INQUIRE operation has been completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=ASY.

OPTCD=CIDXLATE

To determine the name of the logical unit associated with a CID, the RPLARG field must contain the CID when the INQUIRE macroinstruction is executed.

- If PARMS=(NQNAMES=NO) on the ACB macroinstruction and AREALEN is greater than or equal to 8, the 8-byte name is stored in AREA.
- If PARMS=(NQNAMES=YES) on the ACB macroinstruction and AREALEN is greater than or equal to 16, the network-qualified name is stored in AREA.

The format of the returned data is the 8-byte network identifier (padded with blanks, if necessary) followed by the resource name (padded with blanks, if necessary).

To use INQUIRE to determine the CID of a session with a logical unit, the RPL's NIB field must contain the address of an NIB. The NAME field of that NIB must in turn contain the symbolic name to be converted. The CID is placed in the data area indicated by the RPL's AREA field. The AREALEN field must be set to 4. If the logical unit is not currently in session, a CID is not returned, and an error code is set (RTNCD,FDB2)=(X'00',X'07').

If the INQUIRE macroinstruction specifies the name of a logical unit with which the application program has established parallel sessions, the CID returned identifies one of the current parallel sessions. The exact session whose CID is returned is unpredictable.

If PARMS=(NQNAMES=YES), then the following logic is used in finding the CID:

- If NIBNET is zeros, blanks, or “RECORD ”, the LU name in the NIB field NIBSYM is used to find the CID.
- If NIBNET is **not** zeros, blanks, or “RECORD”, the LU name in the NIB field NIBSYM and the network identifier in the NIB field NIBNET are used to find the CID.

OPTCD=COUNTS

For the ACB specified in the RPL, INQUIRE provides the number of active sessions and the number of queued CINIT requests for the application program.

The RPL's ACB field must contain the address of the ACB. The AREA and AREALEN fields must indicate a 4-byte area where the information is to be

placed. VTAM places the number of active sessions in the first 2 bytes and the number of queued CINITs in the second 2 bytes.

OPTCD=DEVCHAR

The logical unit (which can be a non-SNA device) for which the device characteristics are to be retrieved is specified either by placing the CID of an active or pending active session into the RPL's ARG field or by specifying an NIB in the RPL. For application programs not capable of parallel sessions (PARSESS=NO on the APPL definition statement), if a NIB is specified, the NIBSYM field must specify the name of the logical unit. For application programs capable of parallel sessions, if an NIB is specified, the operation of the DEVCHAR option depends on the setting of the NIBCID field. If the NIBCID field contains 0, the NIBs NAME (NIBSYM) field must specify the name of the logical unit. If the NIBCID field is not 0, the logical unit is the one associated with the specified session.

The device characteristics are placed in an 8-byte program storage area whose location is set in the AREA field. The AREALEN field must be set to 8. See the description of the ISTDVCHR DSECT in Appendix E, "Control block formats and DSECTs," on page 739, for a complete description of the DEVCHAR information.

OPTCD=LOGONMSG

The RPL's ACB field must indicate the ACB to which the CINIT is directed. The AREA and AREALEN fields must indicate the location and length of the storage area where the user data field (the logon message) from the Initiate that led to the CINIT is to be placed. The particular user data field to be retrieved is specified either by placing the CID of the associated pending active session into the RPLARG field or by specifying an NIB in the RPL.

For application programs not capable of parallel sessions (PARSESS=NO on the APPL definition statement), if an NIB is specified, the NIBSYM field must specify the name of the logical unit associated with the pending active session (and thus uniquely identifies the queued CINIT containing the user data).

For application programs capable of parallel sessions (PARSESS=YES on the APPL definition statement), if an NIB is specified, the operation of the LOGONMSG option depends upon the setting of the NIBCID field. If the NIBCID field contains a value of 0, INQUIRE obtains the data associated with the oldest queued CINIT for a session with the named logical unit; if the NIBCID field contains a value other than 0, INQUIRE obtains the data from the specific queued CINIT represented by the CID.

If PARMS=(NQNAMES=YES), then the following logic is used in finding the CINIT:

- If NIB is specified and if it contains a network identifier in the NIBNET field, the network identifier is used along with the LU name in NIBSYM to find the CINIT.
- If NIBNET is zeros, blanks, or "RECORD ", then only NIBSYM is used to find the CINIT.

Note: The information necessary to be supplied for the ACB, NAME, CID, and AREALEN fields can be obtained from the LOGON exit routine's parameter list. Refer to "LOGON exit routine" on page 238 for further information.

VTAM indicates the actual length of the user data in the RPL's RECLEN field. If the length of data is AREALEN, RECLEN is posted with the required length; conditional completion is indicated, (RTNCD,FDB2)=(X'00',X'05'); and no data

is supplied to the application program. INQUIRE OPTCD=LOGONMSG can then be reissued specifying a larger AREALEN.

Note: The user data portion of a CINIT cannot be obtained after an OPNDST or CLSDST has been issued for the pending active session with which CINIT is associated.

OPTCD=NQN

If a symbolic-to-real translation request is made, the application program supplies a non-network-qualified symbolic name and asks VTAM to translate the symbolic name into a real network identifier and a resource name. If a real-to-symbolic translation request is made, the application program supplies a network-qualified name (a network identifier and a resource name) and asks VTAM to translate the real name into a non-network-qualified symbolic name if one is defined to that VTAM.

VTAM performs a symbolic-to-real translation if NIBNET is filled in with "RECORD ", blanks, or null. In other words, VTAM translates the symbolic name into a network-qualified name. The AREALEN field must be greater than or equal to 16.

VTAM performs a real-to-symbolic translation if NIBNET is filled in with a network identifier. In other words, VTAM translates the network-qualified name into a symbolic name. The AREALEN field must be greater than or equal to 8.

OPTCD=PERSESS

This option causes VTAM to create an NIB for each record application program interface session that is pending recovery in the area provided by the user (RPLAREA). The NIB provides the application with information that is used to restore the session. This information includes:

- The network ID of the LU
- The VTAM CID

  Note: The CID of a session restored for persistent sessions is different from the CID of the original session.
- The value of the previous NIB USERFLD
- An indication of whether the application is the PLU or the SLU
- An indication of whether this is the last NIB in the list
- The setting of the session's continue-any mode and continue-specific mode for the three data types: DFSYN, DFASY, and RESP
- A pointer to the restore parameter list.

  The restore parameter list indicates where to locate:
  - BIND information and user data structure subfields if BINUSEL is non-zero (for LU 6.1 and LU 6.2)
  - Session qualifier pair (for LU 6.1)
  - MODENAME (for LU 6.2)
  - Session instance identifier (for LU 6.2).

When INQUIRE OPTCD=PERSESS is issued, AREA indicates where the information produced by the macroinstruction is placed. See "Restoring sessions pending recovery" on page 137 for more information.

If the data to be provided exceeds the AREALEN value, one of two special conditions results, (RTNCD,FDB2)=(X'00',X'05') or (RTNCD,FDB2)=(X'00',X'0D'). VTAM sets the return code to (RTNCD,FDB2)=(X'00',X'05') if AREALEN is

insufficient to hold the information for at least one session pending recovery (in this case, RPLAREA will not contain any data). VTAM sets the return code to (RTNCD,FDB2)=(X'00',X'0D') if AREALEN is insufficient to hold the information for all sessions pending recovery. See “Restoring sessions” on page 139 for more information. The RECLEN field indicates the length that was used (in this case, RPLAREA will contain the NIBs for as many sessions as the RPLAREA is capable of holding). If AREALEN is large enough to hold the data, a different condition results, (RTNCD,FDB2)=(X'00',X'00'). If INQUIRE OPTCD=PERSESS is issued and no sessions are pending recovery, another condition results, (RTNCD,FDB2)=(X'00',X'07').

VTAM builds NIBs and returns them to the issuer of the macroinstruction.

- If PARMS=(NQNAMES=YES) on the ACB macroinstruction, the field NIBSYM contains the LU name and the field NIBNET contains the network identifier.
- If PARMS=(NQNAMES=NO) on the ACB macroinstruction, the field NIBSYM contains the LU name and the field NIBNET contains zeros.

OPTCD=SESSKEY

For a particular session, INQUIRE returns cryptographic session information which consists of a 16-byte field containing the session cryptography key (the first 8 bytes) and the initial chaining value (ICV) (the second 8 bytes). The AREA field indicates the location of this 16-byte field. The AREALEN field must be set to 16. The session is specified by either the CID in the RPLARG field or by a CID or NAME field in the NIB.

For application programs not capable of parallel sessions (PARSESS=NO on the APPL definition statement), if an NIB is specified, the NIB's NAME (NIBSYM) field must specify the name of the logical unit. For application programs capable of parallel sessions, if an NIB is specified, the operation of the SESSKEY option depends on the setting of the NIBCID field. If the NIBCID field contains 0, the NIB's NAME (NIBSYM) field must specify the name of the logical unit, and the session cryptography information is retrieved for a session with the named logical unit; if multiple sessions exist with the logical unit, the session for which the information is retrieved is unpredictable. If the NIBCID field is not 0, the session cryptography information is retrieved for the session identified by the CID.

If PARMS=(NQNAMES=YES), then the following logic is used in finding the session:

- In the NIBNET field, the network identifier is used along with the LU name in NIBSYM to find the session.
- If NIBNET is zeros, blanks, or “RECORD ”, only NIBSYM is used to find the session.

OPTCD=SESSNAME

Allows you to determine the network-qualified name of the generic resource application that is associated, using the generic name, with a specific LU.

The NIBSYM field contains the name of the LU; NIBNET is the network identifier of the LU. NIBGENN provides the generic resource name used to establish the session.

VTAM determines which application is associated with the specified LU and generic resource name and returns the name and netid of the application in the AREA field. The AREALEN field must be set to 16.

If the logical unit initiates its sessions using the application network names, rather than the generic resource name, error code (RTNCD,FDB2)=(X'14',X'88') is set indicating that no LU-to-application associations match the given criteria.

If the application program does not support network-qualified names and NETID is not specified, VTAM will use the first occurrence of the partner name.

OPTCD=SESSPARM

To determine the session parameters associated with a specified logon mode name, the NIB field of the RPL must point to an NIB whose LOGMODE operand identifies the logon mode name used. The logon mode name that is specified in the NIB is used to search the logon mode table defined for the logical unit named in the NIB. If a match is found, the session parameters associated with the logon mode name are returned in the AREA field of the RPL. The AREALEN field must be set to at least 36 (decimal).

To determine the session parameters associated with a queued CINIT, either the RPLARG field can specify the CID of the pending active session or the RPL's NIB field can specify an NIB that in turn specifies the pending active session. In either case, the session parameters (including the user data field from Initiate) from the queued CINIT are returned in the AREA field. AREALEN must be large enough to contain the session parameters from the queued CINIT. If the NIB technique is used, the LOGMODE field must be set to 0 (indicating that the session parameters of a queued CINIT are requested). For application programs not capable of parallel sessions (PARSESS=NO on the APPL definition statement), if an NIB is specified, the NIB's NAME (NIBSYM) field must specify the name of the logical unit. For application programs capable of parallel sessions, if a NIB is specified, the operation of the SESSPARM option depends on the setting of the NIBCID field. If the NIBCID field contains 0, the NIBSYM field must specify the name of the logical unit, and the session parameters are retrieved from the CINIT for the oldest pending active session with that logical unit. If the NIBCID field is not 0, the session parameters are retrieved from the CINIT identified by the CID.

See the description of the LOGMODE operand of the NIB macroinstruction for more information. For more information on session parameters, refer to Appendix F, “Specifying a session parameter,” on page 793.

OPTCD=STATUS

INQUIRE checks the status of an application or LU and returns a value in the RPL's FDBK field. Specify this option code to receive the status of an LU. If the requested resource is an application, or if an error is detected, the result is the same as though an INQUIRE OPTCD=APPSTAT was issued. If the resource is an LU, one of the following RPL FDBK values is returned:

RPLFDB3 value	Explanation
X'80'	The resource is active. For an independent LU, the PU is either active or connectable.
X'84'	The resource is not active. The resource currently has a status other than active.

OPTCD=SYN

Control is returned to the application program when the INQUIRE operation has been completed. Once the INQUIRE operation has been completed, the

ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN.

Because it might take VTAM a relatively long time to complete the INQUIRE operation, you should not use the SYN option if suspending the INQUIRE issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

OPTCD=TERMS

The RPL's NIB field must point to a NIB whose NAME field contains the name of a resource known to VTAM at the time INQUIRE is executed. If the name in the NIB is an LU, an NIB is built for that LU. If the name is for a resource other than an LU, an NIB is built for that resource and all dependent LUs that are subordinate to that resource.

The AREA and AREALEN fields designate the location and length of the work area where the NIBs are built. The work area must be set to binary zeros by the application program before INQUIRE is issued.

VTAM indicates the total length of the NIBs in the RPL's RECLEN field.

If the application program wants the NIBs to be built in dynamically allocated storage (obtained by the application program), INQUIRE should be issued twice. For the first INQUIRE, set AREALEN to 0. This INQUIRE is completed with (RTNCD,FDB2)=(X'00',X'05') (insufficient length), and RECLEN indicates the required length. Obtain the storage and reissue INQUIRE with AREALEN set to the proper length.

Each NIB contains the symbolic name of a logical unit, with flags for the LISTEND field set in such a way as to group the NIBs together into an NIB list. In addition, except for a cross-domain LU, the same device characteristics are placed in each NIB as would be obtained for each logical unit by issuing INQUIRE OPTCD=DEVCHAR for that logical unit. The device characteristics field in the NIB might not always be set for a cross-domain LU.

After you have set the NIB fields to their desired values, the NIBs are ready to be used for session establishment.

If PARMS=(NQNAMES=YES), then the following logic is used in finding the LU:

- If the NIB is specified and if it contains a network identifier in the NIBNET field, the network identifier is used along with the LU name in NIBSYM to find the LU.
- If NIBNET is zeros, blanks, or “RECORD ”, only NIBSYM is used to find the LU.

OPTCD=TOPLOGON

The ACB field of the INQUIRE's RPL must indicate the ACB whose CINIT queue is to be examined. The symbolic name of the logical unit associated with the oldest queued CINIT is returned in the data area indicated by you in the RPL's AREA field. The AREALEN field must be set to 8. The CID of this pending active session is returned in the RPLARG field.

If no CINITs are queued, INQUIRE is posted complete with (RTNCD,FDB2)=(X'00',X'07').

- If PARMS=(NQNAMES=NO) and AREA is greater than or equal to 8, the non-network-qualified name is returned. The format of the returned data in RPLAREA is an 8-byte name, (padded with blanks if necessary).

- If PARMS=(NQNAMES=YES) and AREA is greater than or equal to 16, the network-qualified name is returned. The format of the returned data in RPLAREA is an 8-byte network identifier, (padded with blanks if necessary), followed by the resource name, (padded with blanks if necessary).

OPTCD=USERVAR

Use OPTCD=NQN in place of OPTCD=USERVAR. Although OPTCD=USERVAR can be issued, the actual macroinstruction invocation always returns a successful RTNCD,FDB2 and returns the translated name as the same name passed as input. NQN is the suggested option to be used for name translation requests.

RPLAREA points to an 8-byte field that is the name of the USERVAR. Set RPL AREALEN to 8 for a USERVAR name.

Examples

```
INQ1     INQUIRE RPL=RPL1,OPTCD=APPSTAT,NIB=NIB1
         .
         .
         .
TST1     TESTCB RPL=RPL1,FDBK=0
         BE    ACTIVE
         .
         .
         .
NIB1     NIB   NAME=PGM1
```

INQ1 determines whether PGM1 is active and accepting session-establishment requests. The answer is returned in RPL1's FDBK field. TST1 and the branch instruction cause a branch to ACTIVE if the application program is active and accepting session-establishment requests.

```
INQ2     INQUIRE RPL=RPL2,OPTCD=LOGONMSG,                              C
               ACB=ACB1,NIB=NIB2,                                      C
               AREA=LGNMSG,AREALEN=100
         .
         .
         .
NIB2     NIB   NAME=LU2
LGNMSG   DS    CL100
```

INQ2 obtains the user data portion of the Initiate that was sent from or on behalf of the logical unit whose symbolic name is contained in NIB2 and that requested a session with the application program represented by ACB1. This data is placed in the area designated as LGNMSG.

Completion information

The INQUIRE operation is successfully completed when the information has been placed into the application program's storage area.

When the INQUIRE operation completes, the following RPL fields are set:

- The value 26 (decimal) is set in the REQ field indicating an INQUIRE request.
- If INQUIRE OPTCD=APPSTAT completes normally, as indicated in register 15, VTAM sets the FDBK field to one of the values listed under APPSTAT, discussed earlier in this macroinstruction description.
- If INQUIRE (all versions except OPTCD=APPSTAT) has been completed normally, the RECLEN field indicates the number of bytes of data that are placed in the work area designated by the AREA field. If INQUIRE completed

successfully, but the FDB2 field indicates that the work area is too small, (RTNCD,FDB2)=(X'00',X'05'), RECLEN indicates the required length.

- If INQUIRE OPTCD=TOPLOGON completes normally, as indicated in register 15, VTAM sets the ARG field with the CID of the oldest queued CINIT.
- If the macroinstruction returns an error code, the SSENSEI, SSENSMI, and USENSEI fields might be set indicating system-sense information, system-sense modifier, and user-sense information. See Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651 for more information about these fields.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651. VTAM also sets registers 0 and 15 as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

INTRPRET—Interpret an input sequence

Purpose

For each device-type logical unit, INTRPRET allows an application program to use interpret tables to convert an input character sequence into an output character sequence.

Usage

This function uses interpret tables that are specified by the user and maintained by VTAM, rather than tables that are created and maintained by each application program.

During VTAM definition, the user identifies each device-type logical unit in the domain and optionally associates an interpret table with each one. The interpret table contains one or more variable-length sequences that the device-type logical unit is capable of sending—such as graphic characters, tab characters, or program function key characters. With each of these sequences, the user specifies a corresponding 8-byte sequence (or the address of a user-written routine that generates an 8-byte sequence). An application program issuing INTRPRET identifies the device-type logical unit and provides a particular sequence received from the logical unit; VTAM, if it finds that sequence in the interpret table for that logical unit, returns the corresponding sequence to the application program.

Note: Do not define an interpret table for other application programs, or use it to interpret data from a logical unit in another domain. Refer to the z/OS Communications Server: SNA Resource Definition Reference for details on interpret tables.

As an example, assume that the user defines the following interpret table for logical unit T3270:

Symbol	T3270 interpret table
LGN	LOGON
#	REPEATLT
@	LIST

If the application program specifies T3270 and provides any sequence beginning with # to INTRPRET, INTRPRET would return the corresponding sequence—in this case, REPEATLT—to the application program.

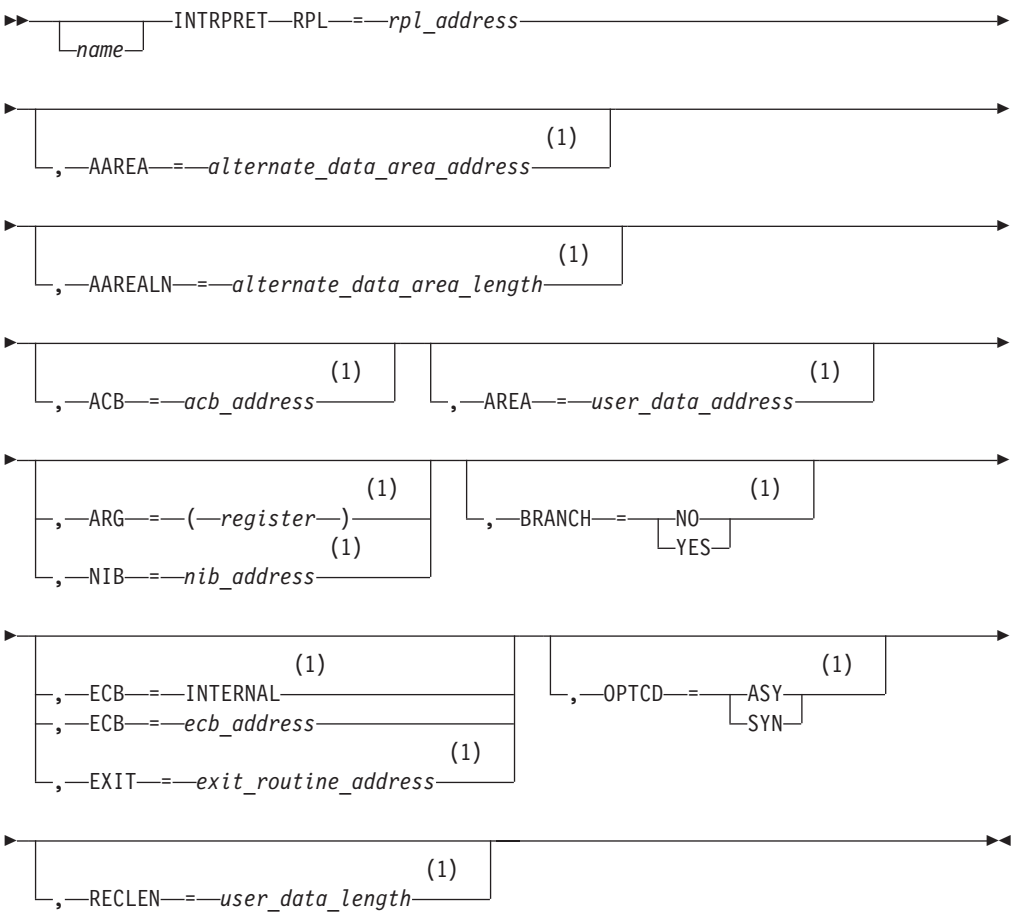
Before issuing the INTRPRET macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from the INTRPRET macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

For output, the name placed in the AAREA field depends on whether PARMS=(NQNAMES=YES) or PARMS=(NQNAMES=NO) on the ACB macroinstruction:

- If PARMS=(NQNAMES=NO) on the ACB macroinstruction, the 8-byte name is returned.
- If PARMS=(NQNAMES=YES) on the ACB macroinstruction, the 16-byte network-qualified name is returned. The format of the returned data is the 8-byte network identifier (padded with blanks if necessary) followed by the 8-byte resource name (padded with blanks if necessary).

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing INTRPRET is to perform.

The following RPL operands apply to the INTRPRET macroinstruction:

AAREA=*alternate_data_area_address*
Indicates the data area where VTAM is to place the interpreted sequence.

AAREALN=*alternate_data_area_length*
Indicates the length of the data area where VTAM is to place the interpreted sequence. If NQNAMES=NO, this value should be at least 8. If NQNAMES=YES, this value should be at least 16.

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing INTRPRET.

AREA=*user_data_address*
Indicates the data area containing the sequence being submitted to VTAM for interpretation.

ARG=(register)
Indicates the register containing the CID of the session with the device-type logical unit. VTAM searches for an interpret table for the associated device-type logical unit. If the ARG operand is not specified, the RPLARG field must contain an NIB address.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) INTRPRET operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) INTRPRET operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

NIB=*nib_address*
Indicates the NIB whose NAME field identifies the device-type logical unit and if using NQNAMES, indicates the NIB whose NIBNET field contains the network identifier of the logical unit. VTAM searches for an interpret table for this device-type logical unit. If the NIB operand is not specified, the RPLARG field must contain a CID.

Note: If your application uses the RPL DSECT, IFGRPL, you must set the RPLNIB bit if an NIB address is being inserted into the RPLARG field.

OPTCD=SYN
OPTCD=ASY
If the SYN option code is set, control is returned to the application program when the INTRPRET operation has been completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. After the INTRPRET operation has been completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the INTRPRET operation, you should not use the SYN option if suspending the INTRPRET-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

RECLEN=*user_data_length*
Indicates how many bytes are being submitted to VTAM for interpretation. If a sequence in the interpret table is shorter than the value in RECLEN, and yet is identical to the corresponding initial part of the data pointed to by AREA, the sequences are considered matched. Refer to the z/OS Communications Server: SNA Resource Definition Reference for coding information on the logon-interpret routine. RECLEN must be set to 255 (decimal) or less.

Examples

```
INT1     INTRPRET RPL=RPL1,                                          C
               NIB=NIB6,AREA=INSEQ,RECLEN=(3),                       C
               AAREA=OUTSEQ,AAREALN=8
         .
         .
         .
RPL1     RPL
INSEQ    DS    CL180
NIB6     NIB   NAME=LU1
OUTSEQ   DS    CL8
```

An application program has read a block of data from LU1 and issues INT1 to interpret that data. NIB6 identifies the LU (and therefore, the interpret table to be used), AREA indicates the data area containing the data to be interpreted (INSEQ), and RECLEN indicates that the amount of data to be interpreted is in register 3. If INTRPRET uses the same RPL that was used to read the data, the NIB-ARG field, the AREA field, and the RECLEN field are already correctly set.

Upon completion of INT1, the corresponding sequence is placed in the data area identified by the AAREA field (OUTSEQ). Although two separate data areas have been provided in this example for the "input" data (INSEQ) and the "output" data (OUTSEQ), there is no reason why the same data area could not be used.

Completion information

The INTRPRET operation is successfully completed as soon as the output character sequence has been placed in the application program's storage area.

When the INTRPRET operation is completed, these RPL fields are set:

- The value 27 (decimal) is set in the REQ field, indicating an INTRPRET request.
- If the FDB2 field indicates that INTRPRET failed because the data to be placed in the AAREA work area, (RTNCD,FDB2)=(X'00',X'05'), does not fit, the ARECLEN field contains the number of bytes required to hold the data. If INTRPRET is completed successfully, the ARECLEN field indicates how many bytes of data have actually been placed in the AAREA work area.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.
- If the macroinstruction returns an error code, the SSENSEI, SSENSMI, and USENSEI fields can be set indicating system-sense information, system-sense modifier, and user-sense information. See Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651 for more information about these fields.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

ISTGLBAL—Declare and set macro global variables

Purpose

Use the ISTGLBAL macroinstruction when assembling an application program to allow the program to discover the characteristics of the VTAM product associated with the macroinstruction definition library that contains ISTGLBAL.

Usage

Invoke ISTGLBAL by the IFGRPL and IFGACB macroinstructions as an inner macroinstruction.

The ISTGLBAL macroinstruction declares and sets two types of global variables:

- GBLC variable, which indicates the release level of VTAM
- GBLB variables, which provide a list of functions available for this release of VTAM.

To use the ISTGLBAL macroinstruction, you must be familiar with the GBLB, GBLC, SETB, and SETC assembler language instructions.

Instructions for coding the ISTGLBAL macroinstruction, with examples, are contained in "ISTGLBAL macroinstruction" on page 274.

Syntax

MODCB—Modify the contents of control block fields

Purpose

MODCB modifies the contents of one or more fields in an ACB, EXLST, RPL, or NIB control block. MODCB works with control blocks created either with declarative macroinstructions or with the GENCB macroinstruction.

Usage

The user of the MODCB macroinstruction indicates the location of the control block, the fields within the control block to be modified, and the new values that are to be placed or represented in those fields.

Most fields whose contents can be set with the ACB, EXLST, or NIB macroinstruction can also be modified by the MODCB macroinstruction. The operands used to do this are the same as those used when the control block is created.

The following restrictions apply to the use of MODCB:

- An ACB cannot be modified after an OPEN macroinstruction has been issued for it.
- An exit list (EXLST) cannot have exits added to it with the MODCB macroinstruction. If an exit list field is not specified in the EXLST macroinstruction, do not attempt to modify that field with a MODCB macroinstruction. MODCB can, however, be used to change dummy exit addresses to valid addresses. This must be done before OPEN is issued for the ACB that references the EXLST control block, or before an OPEN (for a CNM application program), OPNDST, or OPNSEC is issued using the NIB for referencing the EXLST.
- An RPL cannot be modified while a request using the RPL is pending, that is, while the RPL is active.
- An NIB should not be modified if it was referenced by an RPL that is still active.
- The AM field of the ACB, EXLST, and RPL control blocks cannot be modified. Once a control block has been generated in a VTAM-compatible form, it cannot later be modified for use with another access method.

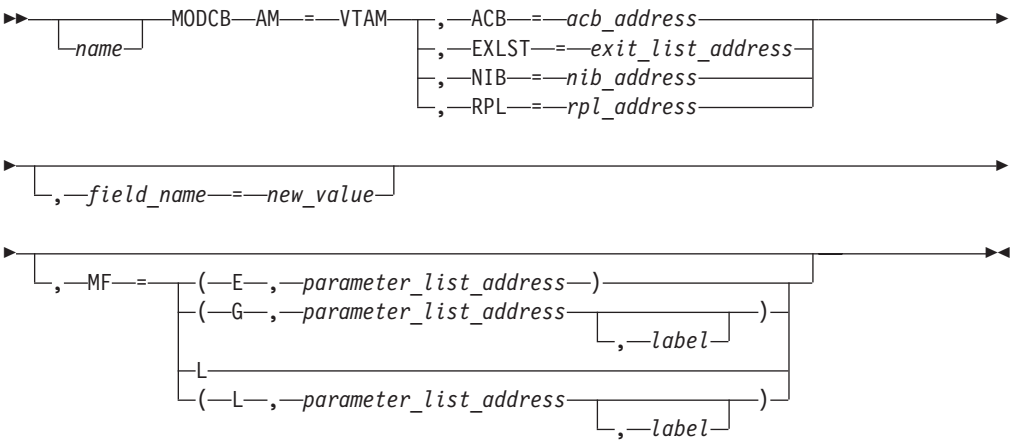
List, generate, and execute forms of the MODCB macroinstruction are available; they are designated by the MF operand. (See Appendix K, “Forms of the manipulative macroinstruction,” on page 865, for more information.)

Because the various MODCB operand values can be specified in a large variety of formats, the operand format specifications have been tabulated in Appendix J, “Summary of operand specifications,” on page 857, and do not appear here.

Note: Not all control block fields can be modified by using MODCB. Refer to Appendix J, “Summary of operand specifications,” on page 857, to determine fields that are modifiable. Other fields can be modified at execution time by use of the appropriate control block DSECTs as described in Appendix E, “Control block formats and DSECTs,” on page 739.

An application program running in either 24- or 31-bit addressing mode can issue the MODCB macroinstruction. To use 31-bit addressing, the application program must use the VTAM mapping macroinstructions as well as GETMAIN and FREEMAIN. Refer to "31-bit addressing" on page 319 for information about 31-bit addressing.

Syntax



Input parameters

AM=VTAM
Identifies this macroinstruction as a VTAM macroinstruction. This operand is required.

ACB=*acb_address*

EXLST=*exit_list_address*
NIB=*nib_address*
RPL=*rpl_address*
Indicates the type and location of the control block whose fields are to be modified.

field_name=*new_value*
Indicates a field in the control block to be modified and the new value that is to be contained or represented within it.

For *field_name*, code the keyword of any MODCB-supported operand (see Appendix J, "Summary of operand specifications," on page 857, for more information) that can be coded in the macroinstruction corresponding to the ACB, EXLST, RPL, or NIB operand. If RPL=RPL1 is coded, for example, the keyword of any MODCB-supported operand in the RPL macroinstruction can be coded. ARG=(*register*) can also be coded.

For *new_value*, code any value that could be used in an ACB, EXLST, RPL, or NIB macroinstruction, or use one of the formats indicated in Appendix J, "Summary of operand specifications," on page 857.

MF=E, G, or L
Indicates that an execute, generate, or list form of MODCB is used. Omitting this operand causes the standard form of MODCB to be used. See Appendix K, "Forms of the manipulative macroinstruction," on page 865, for a description of the execute, generate, or list forms of MODCB.

Examples

```
MOD1     MODCB RPL=(5),OPTCD=(ASY,SPEC,CS),AM=VTAM
```

MOD1 activates the ASY, SPEC, and CS option codes in an RPL. The settings for the other option codes are not affected. The address of this RPL must be in register 5 when MOD1 is executed.

Completion information

After MODCB processing is completed, register 15 indicates whether the operation completed successfully. If the operation completed successfully, register 15 is set to 0; if it completed unsuccessfully, register 15 is set to either X'04', X'08', or X'0C'. If register 15 is set to X'04' or X'0C', register 0 is also set indicating the specific nature of the error (see Appendix I, "Return codes for manipulative macroinstructions," on page 855, for more information).

NIB—Create a node initialization block

Purpose

The NIB is used during the initiation and establishment of LU-LU sessions to identify and define those sessions. Similarly, it is used to define the SSCP-LU session for a CNM application program. The NIB is also used to identify queued, pending active, or active sessions that the application program wants to terminate. Finally, it is used by the INQUIRE and INTRPRET macroinstructions to specify parameters that cannot be specified in the RPL.

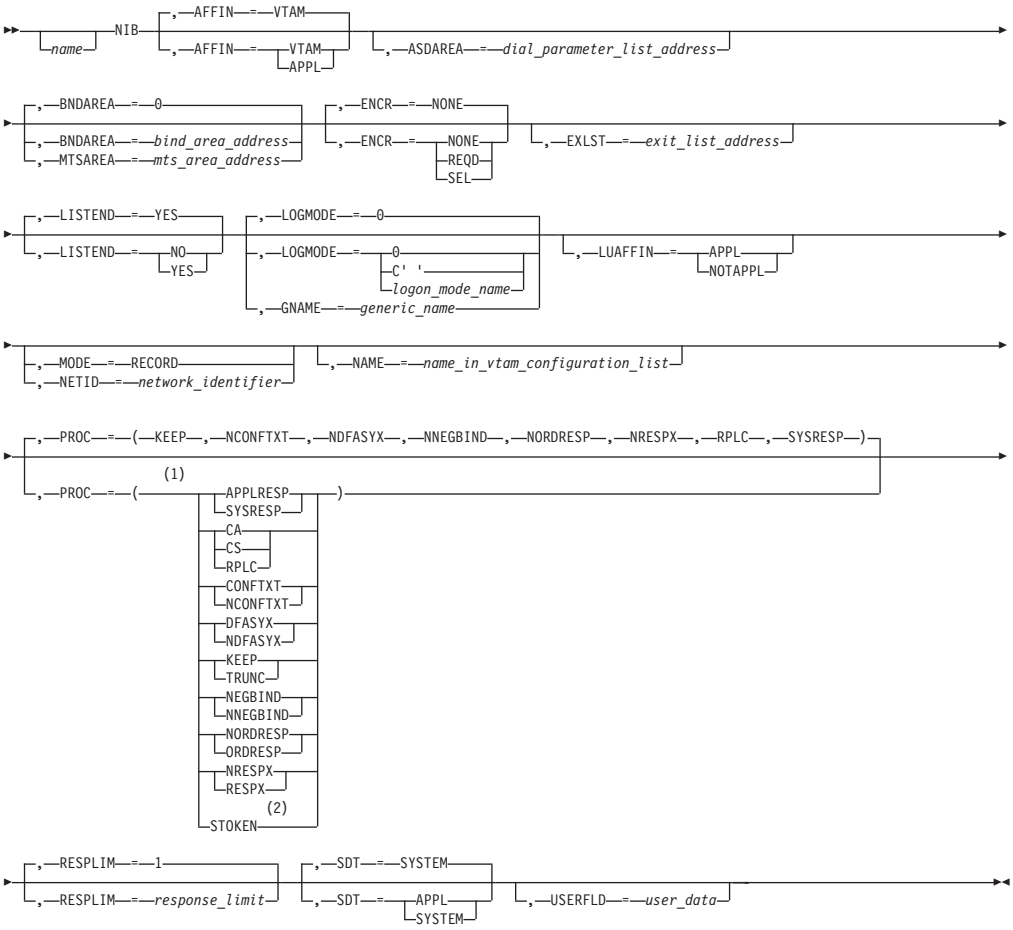
Usage

For certain macroinstructions, NIBs can be grouped together into lists. When requests are directed to an NIB that is the first in an NIB list, VTAM considers all of the logical units (LUs) represented in the NIB list to be the objects of the request, not just the LU represented by the first NIB.

The NIB macroinstruction causes the NIB control block to be built during program assembly; it is built on a fullword boundary. The NIB macroinstruction is not executable. An NIB can be built during program execution with a GENCB macroinstruction. It can be subsequently modified using the ISTDNIB and ISTDPROC DSECTs, and by the MODCB macroinstruction. MODCB does not, however, support all NIB fields.

The expansion of the NIB macroinstruction is identical for 24- and 31-bit addressing mode application programs.

Syntax



Notes:

1 You can code more than one suboperand on PROC, but code no more than one from each group.

2 PROC=STOKEN and the NAME parameter are mutually exclusive.

Input parameters

AFFIN

Indicates whether VTAM or the application program is to be the owner of any affinities that are established for this instance of a generic resource being established by the SETLOGON OPTCD=GNAMEADD macroinstruction. The AFFIN parameter is mutually exclusive with the LUAFFIN parameter.

AFFIN=VTAM

Indicates that VTAM is to be the owner of any affinities not explicitly owned by the application program.

AFFIN=APPL

Indicates that the application program is to be the owner of all affinities that are established.

ASDAREA=*dial_parameter_list_address*

Specifies the address of the dial parameter list that is supplied by the application. The ASDAREA operand sets the NIBASDP and NIBASDPA fields. The NIBASDP field indicates that during session establishment the application supplies the dial parameters. The NIBASDPA field holds the dial parameter list

address. You must use a fully initialized NIB for each established session. For additional information see Appendix E, "Control block formats and DSECTs," on page 739.

Note: VTAM can overwrite the NIBASDPA field during certain macroinstruction calls that use the NIB. For example, the persistent LU-LU session function uses the same 4-byte field as NIBASDPA to store the address of the recovery data when restoring sessions pending recovery.

BNDAREA=*BIND_area_address*

Permits the application program to explicitly specify a set of session parameters VTAM uses in constructing a BIND request or BIND response that is sent to establish a session.

BNDAREA=*bind_area_address*

Indicates the location of a BIND area containing session parameters to be used by OPNDST and OPNSEC macroinstructions. Session parameters specified in a BIND area generally override session parameters available from any other source. See Figure 177 on page 818 for details. See also "NIB LOGMODE and BNDAREA operands" on page 124 for details.

Note: BNDAREA and MTSAREA are mutually exclusive keywords on this macroinstruction.

BNDAREA=0

BNDAREA=0 must be specified, explicitly or by default, in the NIB used by a CNM application program to gain access to the SSCP-LU session.

ENCR

Indicates the level of cryptography that the application program requests for the session being established by OPNDST or OPNSEC. If either REQD or SEL is specified, the session is not established unless both ends of the session are capable of cryptography. Note that the actual level of cryptography used on the session can be higher than that specified by the ENCR operand, because the VTAM operator, the VTAM network definition, and the other LU in the session can all request upgrading of the cryptography level. See "Establishing cryptographic sessions" on page 131 for more information.

ENCR=NONE

No enciphering is requested. ENCR=NONE must be specified, explicitly or by default, in the NIB used to give a CNM application program access to the SSCP-LU session.

ENCR=REQD

All data requests (STYPE=REQ, CONTROL=DATA) are sent enciphered.

ENCR=SEL

Data requests are to be enciphered on the basis of the RPL's CRYPT field setting. See the RPL macroinstruction description for further information on the CRYPT field.

EXLST=*exit_list_address*

Indicates an EXLST control block that contains the address of a DFASY, RESP, or SCIP exit routine (or contains the addresses of any combination of these exit routines).

Exit routines indicated by an NIB (NIB-oriented exit routines) are scheduled when VTAM receives input (of the appropriate type) from the session established by means of the NIB through an OPNDST, OPNSEC, or OPEN macroinstruction. If input is received and no appropriate NIB-oriented exit

routine was specified, VTAM then satisfies any appropriate pending RECEIVE macroinstructions or schedules the appropriate ACB-oriented exit routine (if any). For details, refer to “Specifying the DFASY, RESP, and SCIP exit routines in an ACB or NIB” on page 229, and refer to Figure 39 on page 179 through Figure 42 on page 182.

Note: If you omit this operand, the NIB's EXLST field is set to 0.

GNAME=*generic_name*
is used to specify the generic name of the application. This field is used by the SETLOGON OPTCD=GNAMEADD, GNAMEDEL, or GNAMESUB, INQUIRE OPTCD=SESSNAME, and CHANGE OPTCD=ENDAFFIN macros. This name must be unique.

Note:

1. The NAME operand is required with SETLOGON OPTCD=GNAMESUB.
2. The GNAME parameter and the LOGMODE parameter are mutually exclusive.

LISTEND
Allows the application program to group NIBs into lists.

LISTEND=NO
Iindicates that this NIB and the NIB immediately following it in storage are part of an NIB list. Any number of NIBs can be grouped together by specifying LISTEND=NO for each one except the last. NIB lists can only be used with OPNDST OPTCD=ACQUIRE, or with SIMLOGON.

LISTEND=YES
Indicates that this NIB is the last in a list (or is an isolated NIB not part of a list).

Example: The following use of the LISTEND operand effectively groups the Boston NIBs into one group, the Chicago NIBs into another, and defines the Portland NIB as a “list” of one.

```
BOSTON   NIB   NAME=BOSTON1,LISTEND=NO
         NIB   NAME=BOSTON2,LISTEND=YES
CHICAGO  NIB   NAME=CHICAGO1,LISTEND=NO
         NIB   NAME=CHICAGO2,LISTEND=NO
         NIB   NAME=CHICAGO3,LISTEND=YES
PORTLAND NIB   NAME=PORTLAND,LISTEND=YES
```

LOGMODE
is used to specify the logon mode name. If this operand is either omitted or specified as 0, the LOGMODE field of the NIB is set to binary zeros.

The LOGMODE parameter and the GNAME parameter are mutually exclusive.

LOGMODE=*logon_mode_name*
The LOGMODE field in an NIB is used by the INQUIRE OPTCD=SESSPARM, OPNDST, REQSESS, CLSDST OPTCD=PASS, and SIMLOGON macroinstructions to indirectly specify a set of session parameters. See “NIB LOGMODE and BNDAREA operands” on page 124 for details.

LOGMODE=0
must be specified, explicitly or by default, in the NIB used to give a CNM application program access to the SSCP-LU session.

LUAFFIN
Indicates a specific affinity ownership condition between the application and a

specific partner LU for the session that is being started by the OPNDST or OPNSEC macroinstruction. The LUAFFIN parameter is mutually exclusive with the AFFIN parameter.

LUAFFIN=NOTAPPL
Indicates that VTAM is to be the owner of the affinity for a specific LU on the OPNDST or OPNSEC macroinstruction. This option is not honored if a session already exists with this LU, with the application already established as the owner of the affinity.

LUAFFIN=APPL
Indicates that the application program is to be the owner of the affinity for a specific LU on the OPNDST or OPNSEC macroinstruction. This option is not honored if a session already exists with this LU, with VTAM already established as the owner of the affinity.

MODE=RECORD
This operand is allowed by VTAM for migration purposes.

MODE=RECORD is optional and can only be specified if NETID is not specified on the NIB macroinstruction. MODE=RECORD is not valid when NETID is specified on the NIB macroinstuction.

MTSAREA=*MTS_area_address*
Specifies the address of an area containing MTS override data. This data is required when you specify OPTCD=MTS on a REQSESS or CLSDST PASS macroinstruction. You must format the specified area to match the ISTMTS DSECT. For additional information about the ISTMTS DSECT, see Appendix E, "Control block formats and DSECTs," on page 739.

Note: MTSAREA and BNDAREA are mutually exclusive keywords on this macroinstruction. The MTSAREA keyword is an alternate name for the BNDAREA keyword. The NIBMTSAR field set by the MTSAREA keyword is an overlay of the NIBNDAR field set by the BNDAREA keyword.

NAME=*session_parameter_name*
Associates the NIB with its session parameter.

Example:

```
NAME=LU13
```

Although this operand is sometimes optional, the NAME field generally should be set by the time a CLSDST, OPNDST, SIMLOGON, INTRPRET, REQSESS, TERMSESS, SESSIONC, OPNSEC, or INQUIRE or SETLOGON OPTCD=GNAMESUB macroinstruction is issued that uses this NIB. When OPNDST OPTCD=(ACCEPT,ANY) is issued, the NAME field need not be set, because VTAM places the name of the LU in this field. See the individual macroinstruction descriptions to determine whether alternatively the RPLARG or whether the NIBCID field is used.

If you omit this operand, the entire 8-byte NAME field is set to EBCDIC blanks.

This operand must be omitted in the NIB used by a CNM application program to gain access to the SSCP-LU session.

If this application uses NQNAMES=YES, this name is used with the NETID operand.

If this application uses NQNAMES=NO and NETID is not specified, VTAM will use the first located instance of the name provided associated with the application program.

If CHANGE OPTCD=ENDAFFIN|ENDAFFNF is issued using this NIB, you can specify NAME=* to end all affinities associated with the issuing application program. This function is intended for use by an application only immediately prior to closing its ACB.

When issuing SETLOGON OPTCD=GNAMESUB, this operand should be set to the application program network name of the generic resource instance specified in the VTAM node identification block (NIB).

Note: The NAME parameter and PROC=STOKEN are mutually exclusive.

NETID=*network_identifier*
Specifies an optional 1–8 character network name, (padded with blanks), that identifies the network in which the partner LU, identified by the NAME operand, resides. If you do not code NETID, no network identifier is associated with this application program. The value for NETID is stored in the NIBNET field in ISTDNIB.

The NETID operand and the MODE operand are mutually exclusive.

PROC=*processing_option*
Indicates options that VTAM is to follow for subsequent RPL-based requests involving the session established by an OPEN (for a CNM application program), OPNDST, SETLOGON, or OPNSEC using this NIB.

Format: Code as indicated in the preceding assembler format table. The parentheses can be omitted if only one option code is selected:

```
NIB   NAME=LU3,NETID=NETA,PROC=(DFASYX,RESPX,CONFTXT)
```

The following describes each of the possible parameters for PROC.

PROC=CA
CA, CS, and RPLC apply for a session when input received from that session satisfies a RECEIVE. These PROC options override the CS and CA option codes that might have been specified in the RECEIVE macroinstruction or the RPL referred to by the RECEIVE macroinstruction, but not the CS and CA option of any other type of macroinstruction.

With PROC option CA, the session should be put into continue-any mode after this RECEIVE is completed for the type of input that satisfies the RECEIVE. It can be used as long as a session responds to a RECEIVE OPTCD=ANY macroinstruction. This might be the case if the session normally sends no more than one request per transaction.

PROC=CS
The session should be put into continue-specific mode after this RECEIVE is completed for the type of input that satisfies the RECEIVE. It can be used when a session is to be prevented from completing any subsequent RECEIVE OPTCD=ANY macroinstructions. This might be the case if the session normally sends multiple requests per transaction.

PROC=CONDCS
The session should be put into continue-specific mode after this RECEIVE OPTCD=ANY is completed for the type of input that satisfies the RECEIVE only if more input data remains. More input data is considered to be remaining if RECLEN > AREALEN and KEEP was indicated (see PROC=KEEP). If no input data remains, the session is to be prevented

from completing any subsequent RECEIVE OPTCD=ANY macroinstructions for the remaining data. This might be the case if the session normally sends multiple requests per transaction and all the input data can normally be contained in the RECEIVE's buffer. Specifying this options can allow the application the application to avoid the overhead of a RESETSR macroinstruction after every RECEIVE OPTCD=ANY.

In order to ensure that the application receives the input data in the correct order, EXIT should be specified on the RECEIVE. Also note that if a session does send in multiple requests per transaction, a single session can use up a greater number of RECEIVE OPTCD=ANY macros. Consequently, an application that utilizes this function may want to increase the number of RECEIVE OPTCD=ANY macros it has outstanding.

PROC=RPLC
The CS, CA or CONDCS option code in the RECEIVE macroinstruction or the RPL referred to the RECEIVE macroinstruction should be used when switching continue modes.

PROC=KEEP
VTAM fills the input data area and saves any remaining data for subsequent RECEIVE macroinstructions.

PROC=TRUNC
VTAM fills the input data area and discards any remaining data. No error condition is indicated.

The presence of excess data can be determined by comparing the RPL's AREALEN field (input area size) with the RECLEN field (amount of incoming data). If the value in RECLEN exceeds the value in AREALEN, excess data has been kept (and is used to satisfy the next appropriate RECEIVE). An example is shown in “Handling overlength input data” on page 182.

The NIB's TRUNC-KEEP processing option is effective only if the NIBTK option code is set in the RPL. If the KEEP or TRUNC option codes are set in the RPL, then the NIB's TRUNC-KEEP processing option is overridden.

PROC=CONFTXT
The buffers used to hold data are cleared before they are returned to their buffer pools. The data is considered as “confidential”.

PROC=NCONFTXT
The buffers used to hold data are not cleared before they are returned to their buffer pools.

PROC=DFASYX
When DFASYX is set for the sessions' NIB and no other restrictions prevent the scheduling of the ACB-oriented DFASY exit-routine, the exit routine is scheduled. If the exit routine cannot be scheduled or the session is established with PROC=DFASYX, a RECEIVE OPTCD=ANY, RTYPE=DFASY is not valid. See Figure 40 on page 180 for more information about the DFASY exit routine.

PROC=NDFASYX
The DFASY exit routine is not scheduled.

PROC=NEGBIND
When a PLU application program specifies NEGBIND in an OPNDST macroinstruction, VTAM sends a negotiable BIND request to the SLU.

PROC=NNEGBIND
When a PLU application program specifies NNEGBIND in an OPNDST macroinstruction, VTAM sends non-negotiable BIND request.

When an SLU application program receives a negotiable BIND request in its SCIP exit routine, the session parameters can be checked to determine if the suggested parameters are agreeable. If the application program agrees with the session parameters sent by the PLU, the application program issues an OPNSEC macroinstruction that refers to an NIB with NNEGBIND specified, or with NEGBIND specified and BNDAREA=0. If the application program wants to suggest new session parameters, it issues an OPNSEC macroinstruction with the NIB specifying NEGBIND and BNDAREA containing the address of the new session parameters.

If a list of OPNDST OPTCD=ACQUIRE is used, only one of the NIBs in the list can specify PROC=NEGBIND.

PROC=RESPX
When RESPX is set for the sessions' NIB and no other restrictions prevent the scheduling of the ACB-oriented RESP exit routine, the exit routine is scheduled. If the exit routine cannot be scheduled or the session is established with PROC=RESPX, a RECEIVE OPTCD=ANY, RTYPE=RESP is not valid. See Figure 41 on page 181 for more information about the RESP exit routine.

PROC=NRESPX
The RESP exit routine is not scheduled.

PROC=ORDRESP

PROC=NORDRESP
Indicate whether certain designated normal-flow responses are to be handled by VTAM in a manner similar to the handling of normal-flow (DFSYN) requests. The ORDRESP and NORDRESP options are used in conjunction with the QRESP and NQRESP options in the RPL. For SEND of normal-flow data-flow-control requests, this operand also controls whether the POST and RESPOND fields are examined. Refer to "Controlling the handling of normal-flow responses" on page 161 for details.

PROC=APPLRESP
The application program must respond to the expedited-flow data-flow-control request using a SEND STYPE=RESP macroinstruction. While either a positive or negative response can be sent, a definite response type 1 (FME) must be used in either case. The sequence number and request code (CONTROL) used in the response can be obtained from the RECEIVE RPL or the VTAM read-only RPL supplied in the application exit routine.

For XRF only, during a session takeover, VTAM does not provide reasons for expedited flow requests to the failed active subsystem. The alternate application is responsible for providing these responses, and PROC=APPLRESP must be coded for all XRF sessions.

PROC=STOKEN
specifies that, for SETLOGON OPTCD=GNAMEADD processing, VTAM is to pass the value of the NIBSTKN field in ISTDNIB to the operating system for generic resource workload balancing. *Be sure that NIBSTKN is set when specifying PROC=STOKEN.*

When specified, the PROSTOKN indicator will be set on in the generated NIB.

Note: The NAME parameter and PROC=STOKEN are mutually exclusive.

PROC=SYSRESP
VTAM responds to the request.

RESPLIM=*response_limit*
Indicates the maximum number of responded output requests that can be pending at one time for a session. (A responded output request is a SEND POST=RESP, STYPE=REQ, and CONTROL specifying data or a normal-flow data-flow-control request.) If RESPLIM=0 is coded, VTAM imposes no limit on the number of pending responded output requests. The maximum value that can be coded is decimal 65535. RESPLIM=1 is the default.

SDT
For an application program acting as the primary end of a session, this operand indicates whether the application program or VTAM is to send the first SDT request on the session.

For an application program acting as the secondary end of a session, this operand when coded for an OPNSEC macroinstruction indicates whether the application program or VTAM responds to an SDT request (by issuing SESSIONC CONTROL=SDT, STYPE=RESP).

The use of this operand is determined by the transmission services profile specified in the session parameters used for the session (see Appendix F, "Specifying a session parameter," on page 793). The operand is ignored if the transmission services profile does not include SDT.

For the NIB used to give a CNM application program access to the SSCP-LU session, SDT=SYSTEM must be specified, explicitly or by default.

SDT=SYSTEM
VTAM automatically sends an SDT request as part of the session-establishment process before posting the OPNDST RPL complete.

SDT=APPL
VTAM does not send an SDT request until the application program tells it to do so (by issuing SESSIONC CONTROL=SDT,STYPE=REQ).

USERFLD=*user_data*
Indicates any 4 bytes of data that the application program wants to associate with a session-initiation request, or with the session itself.

The USERFLD in the NIB associated with a SIMLOGON, REQSESS, or CLSDST OPTCD=PASS is retained by VTAM and is subsequently returned to the application program that issued the session-initiation request in the exit parameter list for the LOGON exit routine (for SIMLOGON), SCIP exit routine (for REQSESS), or NSEXIT exit routine (for SIMLOGON, REQSESS, or CLSDST OPTCD=PASS). Refer to "Exit routines related to session establishment and termination" on page 100.

The USERFLD in the NIB associated with an OPEN (for a CNM application program), OPNDST, or OPNSEC macroinstruction is saved by VTAM for the session established by one of these macroinstructions and is returned in the RPL USER field whenever an RPL-based operation (for example, RECEIVE) completes for that session or an exit routine (for example, RESP) is entered for that session.

Note: The USERFLD associated with an OPNDST or OPNSEC macroinstruction overrides that field specified in a SIMLOGON or REQSESS macroinstruction. Thus, if the same 4 bytes of user data are to be saved by VTAM for the session as were saved for the session-initiation request, the OPNDST or OPNSEC macroinstruction must specify the same value originally specified in the corresponding SIMLOGON or REQSESS macroinstruction.

Format: Code the 4 bytes of data in either character, fixed-point, or hexadecimal format, or, if an address is desired, code it as an A-type or V-type address constant. Register notation cannot be used.

Example:

```
USERFLD=C'LU01'
USERFLD=F'100'
USERFLD=00043E0
USERFLD=A(RTN1)
USERFLD=V(EXTRTN)
```

If you omit this operand, the USERFLD field is set to 0.

Examples

```
NIB1     NIB   NAME=LUABC,NETID=NETA,USERFLD=A(LUABCV),                C
               PROC=(RESPX,TRUNC),LISTEND=YES
```

NIB1 represents a device-type LU named NETA.LUABC. When OPNDST is issued referencing this NIB, a session is established with LUABC and the processing options RESPX and TRUNC apply for that session. The address of LUABCV (which can be an application program storage area representing LUABC) is saved by VTAM and is returned to the application program whenever an operation related to that session completes.

NIB fields set by VTAM

The NIB operands already described are supplied by the application program and cause the NIB fields to be set when the NIB macroinstruction is assembled. Additional NIB fields are set by VTAM; these fields can be examined by the application program during program execution. (Some of these fields can also be set by the application program.) VTAM uses these fields to return information to the application program upon completion of processing for OPNDST, OPNSEC, and OPEN (for a CNM application program). These fields are:

Field name
Contents

CID A field containing a 32-bit value identifying the session established by using this NIB. This communication ID (CID) is also placed in the ARG field of the RPL used by the OPNDST or OPNSEC macroinstruction (if one, and only one, session was established). When NIB lists are used, the CID placed in the ARG field is not meaningful. The CID can be examined with the SHOWCB and TESTCB macroinstructions or with the ISTDNIB DSECT. See Appendix E, "Control block formats and DSECTs," on page 739, for more information. (If the session is not established, the CID field is not modified.) This field can be set by VTAM or the application program.

Note: The algorithm used to construct the CID can vary between releases of VTAM. Therefore, the application program should not become dependent on any perceived values.

CON An indicator that is set to show that the session represented by this NIB has been established. You can examine this field following OPNDST or OPNSEC by coding CON=YES in a TESTCB macroinstruction; an "equal" PSW condition code indicates that the CON field is set to YES, and the session is established. This field is useful if you are using OPNDST with a list of NIBs to establish more than one session. Examination of each NIBCON field tells you which sessions were successfully established and which were not. This field is set to NO if the session is not established. This field is set only by VTAM.

Note: You must check the CON field to determine if the session was actually established. A return code indication is not sufficient.

DEVCHAR
An 8-byte field describing certain characteristics of the SLU in the session established by OPNDST. (If the session is not established, the DEVCHAR field is not modified.) This field can be examined with either the SHOWCB or TESTCB macroinstruction or with the ISTDVCHR DSECT. The ISTDVCHR DSECT is described in Appendix E, "Control block formats and DSECTs," on page 739. This field is set only by VTAM.

NIBNACLQ
An indicator that is set to show whether a CINIT is still queued after an OPNDST OPTCD=ACCEPT macroinstruction fails to accept that queued CINIT. If this bit is on (1), it means that the queued CINIT was canceled by VTAM. The session-initiation request must be repeated; for example, a terminal operator has to log on again. If the bit is off (0), an error was detected prior to finding a pending active session with a matching CINIT. Refer to RPLRTNCD and RPLFDB2 for specific error recovery actions.

NAME
An 8-byte field containing the name of the secondary logical unit with which OPNDST OPTCD=(ACCEPT,ANY) has established a session. The NAME field is also filled in if OPNDST or OPNSEC uses the NIB CID field to identify a pending active session; in this case also, the name is that of the session partner LU. This field can be set by VTAM or the application program.

NETID
The name in the network where the session parameter LU resides.

NIBPSPLU
An indicator that shows whether the application is a PLU or an SLU for a particular session. A setting of 1 indicates that the application program is a PLU; 0 indicates that the application program is an SLU. INQUIRE OPTCD=PERSESS and OPNDST OPTCD=RESTORE provide the information.

NIBPSRSP
An indicator that shows the RESP data mode of the session pending recovery. A setting of 0 indicates that the RESP data mode is continue any at the time that the session is retained. A setting of 1 indicates the data mode is continue specific. INQUIRE OPTCD=PERSESS and OPNDST OPTCD=RESTORE provide the information.

NIBPSDFS
An indicator that shows the DFSYN data mode of the session pending recovery. A setting of 0 indicates that the DFSYN data mode is continue any at the time the session is retained. A setting of 1 indicates the data

mode is continue specific. INQUIRE OPTCD=PERSESS and OPNDST OPTCD=RESTORE provide the information.

NIBPSDFA
An indicator that shows the DFASY data mode of the session pending recovery. A setting of 0 indicates that the DFASY data mode is continue any at the time the session is retained. A setting of 1 indicates the data mode is continue specific. INQUIRE OPTCD=PERSESS and OPNDST OPTCD=RESTORE provide the information.

NIBRPARM
An address that points to the restore parameter list. The first word points to the portion of the BIND for the session being recovered that is mapped by ISTDBIND. The second word of this parameter list points to the session state control vector information for the session being recovered.

NIBSLWRK
Indicates SIMLOGON was successful for this NIB.

NIBNNAMS
Indicates that the association established with a partner LU is known by the application's network name.

NIBAFFIN
When the LUAFFIN parameter has been specified on the NIB for an OPNDST RPL or an OPNSEC RPL, the value NIBAFFIN will be set upon completion of the RPL to indicate whether VTAM owns the affinity or the application owns the affinity. If NIBAFFIN is set on, it indicates that the application owns the affinity.

OPEN—Open one or more ACBs

Purpose

The OPEN macroinstruction opens an ACB so that the ACB and all subsequent requests referring to it can be identified by VTAM as applying to a specific application program. During OPEN processing, the application program cannot issue requests that reference the ACB. You indicate the ACB that is to be opened in the OPEN macroinstruction coding.

Usage

After the OPEN macroinstruction completes successfully, VTAM sets the fields of the ACB that point to the access-method-support and resource-information vector lists. The application program can reference these fields until a CLOSE macroinstruction or equivalent (for example, ABEND) occurs. “Vector lists” on page 58 describes the vector lists.

Because an application program can start before VTAM starts, an application program might issue an OPEN macroinstruction before VTAM is active. In this case, OPEN fails and the application program is informed that VTAM is not active. This is the only way an application program can determine whether VTAM is active.

An application that has enabled persistence can use OPEN to recover sessions pending recovery. See “Using persistent LU-LU session support” on page 66 for more information.

OPEN must be issued in the mainline program.

For cross-memory API users, the following conditions must be met:

- OPEN must be issued in non-cross-memory mode by mainline processing under TCB control.
- OPEN must be issued in the address space that becomes the primary address space during a cross-memory VTAM API request.

See "Cross-memory application program interface (API) support" on page 317 for more information.

The ACB and its related storage (APPLID, password, EXLST, NIB and Application-ACB vector list) must be allocated in the same storage key. This key can be the storage key of the program status word (PSW) at the time OPEN was issued, or the storage key of the task control block (TCB).

VTAM prevents attempts to issue OPEN in an RPL exit routine or in any of the other asynchronous exit routines. For additional information about the OPEN macroinstruction, see Chapter 4, "Opening and closing an application program," on page 55.

VTAM supports application programs in 31-bit addressing mode and residing in 31-bit storage. Control block fields referenced by the OPEN macroinstruction can reside in either 31-bit or 24-bit storage but must be consistent with the addressing mode of the application program. The MODE parameter is used to set the addressing mode of the ACB control block for these macroinstructions.

Because further considerations apply, you must issue the list or execute forms of the OPEN macroinstruction.

The standard form of the OPEN macroinstruction expands at assembly time into (1) nonexecutable code that represents the parameters you specified on the macroinstruction and (2) executable code that causes the access method to be entered when the macroinstruction is executed. The nonexecutable code, called the parameter list, is assembled at the point in your application program where the macroinstruction appears.

The list and execute forms of the OPEN macroinstruction cause the assembler to:

- Build the parameter list where the macroinstruction appears in your source code, but assemble no executable code (list form).
- Assemble code that modifies a parameter list and cause the access method to be entered during program execution (execute form).

Table 92 summarizes the actions of these various forms. It also indicates the types of programs that would use each form and shows how the MF operand is used for each form.

Table 92. Forms of the OPEN macroinstruction

Form	During assembly	During execution	Useful for	Coded with
Standard	Parameter list built where macroinstruction appears in source code	Access method entered	Non-reentrant programs that are not sharing or modifying parameter lists	No MF operand

Table 92. Forms of the OPEN macroinstruction (continued)

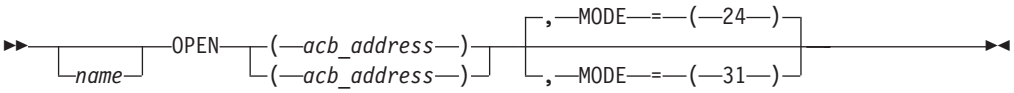
Form	During assembly	During execution	Useful for	Coded with
List	Parameter list built where macroinstruction appears in source code	No executable code (execute form required)	Non-reentrant programs that are sharing or modifying parameter lists	MF=L
Execute	Code assembled (where macroinstruction appears in the source code) to **modify** the parameter list whose address you supply	Parameter list modified and the access method entered	Programs using the list form	MF=(E,*address*)

The OPEN macroinstruction can specify up to 255 ACB addresses and is used to construct a data management parameter list. The parameter list is assembled where the macroinstruction appears in the source code.

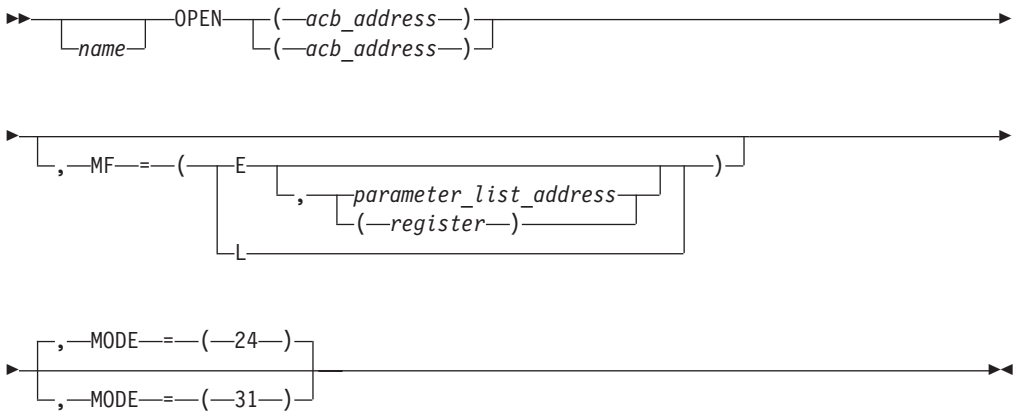
The list consists of a one-word entry for each ACB in the parameter list; the three low-order bytes are used for the ACB address. The end of the list is indicated by a 1 in the high-order bit of the last entry's high-order byte. The length of a list generated by a list-form instruction must be equal to the maximum length required by an execute-form instruction that refers to the same list.

Syntax

This standard form of OPEN is valid.



This is the list and execute form.



Input parameters

acb_address
Indicates the ACB that is to be associated with an APPL entry.

Format: If you specify more than one ACB, separate each with two commas. No more than 255 ACB addresses may be specified.

You can omit the parentheses if you code only one address.

Note: VSAM ACB addresses can also be used in the OPEN macroinstruction. Users can code DCB addresses. You can combine the addresses of different types of control blocks in one OPEN macroinstruction.

MF=E
Indicates that the execute form of the OPEN macroinstruction and an existing parameter list are used. The execute form allows you to modify the parameter list between the generation of that parameter list and the invocation of the access method routines that use the parameter list. Only the execute form provides a means for you to modify the parameter list after it has been built.

parameter_list_address
Indicates the location of the parameter list to be used by the access method.

(register)
Indicates the number of the register that will contain the parameter list address when the macroinstruction is executed.

MF=L
Indicates that the OPEN macroinstruction is used to create a parameter list referred by an execute-form instruction.

MODE
specifies the format of the OPEN parameter list being generated.

24 specifies that a standard form (24-bit) parameter list address be generated. The parameter list must reside below 16 megabytes and point to an ACB residing below 16 megabytes.

31 specifies that a long form (31-bit) parameter list address be generated. This parameter value must be coded if the parameter list or the ACB control block resides above 16 megabytes.

Examples

```
OPEN123 OPEN (ACB1,,ACB2,,(7))
```

OPEN123 opens ACB1, ACB2, and the ACB whose address is contained in register 7. Each of these ACBs is linked with an APPL definition statement.

Completion information

When control is returned to the instruction following the OPEN macroinstruction, register 15 indicates whether the OPEN processing was completed successfully. Successful completion means that all ACBs specified in the OPEN macroinstruction were opened; unsuccessful completion means that at least one ACB was not opened. Successful completion is indicated by a return code of 0 in register 15. The following register 15 values indicate unsuccessful completion:

4 (X'04')
All ACBs were successfully opened, but warning messages were issued for one or more VSAM ACBs.

8 (X'08')
One or more ACBs were not successfully opened. If the error condition indicated by the unopened ACB's ERROR field can be eliminated, another OPEN macroinstruction can be issued for the unopened ACBs.

12 (X'0C')
One or more ACBs were not successfully opened. Another OPEN macroinstruction cannot be issued for the unopened ACBs.

If unsuccessful completion is indicated, the application program should examine the OFLAGS field in each ACB to determine which one (or ones) could not be opened. Test each OFLAGS field by coding an ACB address and OFLAGS=OPEN in a TESTCB macroinstruction; if the resulting PSW condition code indicates an equal comparison, that ACB has been opened:

```
TESTCB   ACB=ACB4,OFLAGS=OPEN
```

If an unequal comparison is indicated, meaning that the ACB has not been opened, the ERROR field can be checked to determine the reason. Like OFLAGS, ERROR is not a field that the application program should modify (that is, there is no ERROR operand for the ACB macroinstruction, and thus none for the MODCB macroinstruction), but the application program can obtain the contents of this field with the SHOWCB or TESTCB macroinstruction. For example:

```
SHOWCB ACB=ACB1,FIELDS=ERROR,AREA=SHOWIT,                    C
       LENGTH=4,AM=VTAM
```

Note: If the ACB is open, or if the address specified in the OPEN macroinstruction either does not indicate an ACB or lies beyond the addressable range of your application program, the ERROR and OFLAGS fields in the ACB are unchanged. Thus, if you find one of the following return codes in the ACB's ERROR field and none of the specified causes apply, perhaps you are actually examining a field whose contents have not been modified by OPEN. An open ACB or an ACB address that is not valid results in register 15 being set to a nonzero value, however.

A list of the values that can be set in the ERROR field of an ACB follow (ACBERFLG is the actual field name). Because most of these error conditions result from an error in your application program or in the system programmer's

definition of VTAM, little can be done during program execution when these return codes are encountered. If, however, you are attempting to open more than one ACB, you might want to check the ERROR field of each ACB. All ACBs whose ERROR fields are set to 0 have been opened successfully, and your application program can proceed using those ACBs.

The value set in the ERROR field of the ACB specified in the OPEN macroinstruction indicates the specific nature of the error (if any) found. Except where noted, all values apply to all operating systems.

ERROR field
Meaning

0 (X'00')
OPEN successfully opened this ACB.

4 (X'04')
The ACB has been opened.

20 (X'14')
OPEN cannot be processed because of a temporary shortage of storage.

36 (X'24')
The OPEN ACB failed for one of the following reasons:

- The password specified by the ACB did not match the corresponding password in the APPL entry.
- The ACB did not specify a password and the APPL contains one.
- The security management product determined that the user is not authorized to open the ACB.

70 (X'46')
OPEN was issued in an exit routine.

80 (X'50')
VTAM has not been included as part of the operating system. The fault lies in the system definition procedures.

82 (X'52')
VTAM is included as part of the operating system, but the VTAM operator issued a HALT command, and VTAM has shut down. You cannot attempt to establish a session or communicate with any LUs.

84 (X'54')
Either the address supplied in the ACB's APPLID field lies beyond the addressable range of your application program, or no entry is found in the VTAM configuration tables that matches the name indicated by the ACB's APPLID field (or supplied by the operating system). If the OPEN macroinstruction is specified correctly, your system programmer might have:

- Failed to include your application program's symbolic name during VTAM definition
- Improperly handled the symbolic name.

Refer to the description of the APPLID operand in “ACB—Create an access method control block” on page 375.

86 (X'56')
A match for your application program's symbolic name is found, but it is for an entry other than an APPL. If you specified this name in the ACB's APPLID field, verify that you have the correct name and handled this

name properly (see the APPLID operand of the ACB macroinstruction). If the symbolic name is supplied by the operating system, the supplied name is suspect.

88 (X'58')
Another ACB, already opened by VTAM, indicates the same application program symbolic name that this ACB does. The system programmer might have assigned the same symbolic name to two application programs. This is valid only if the programs are not open concurrently. Possibly the system operator initiated your program at the wrong time.

90 (X'5A')
No entry is found in the VTAM configuration tables that matches the name indicated by the ACB's APPLID field (or supplied by the operating system). This error might have occurred for one of the following reasons:

- The VTAM operator deactivated the APPL entry.
- The APPL entry was never created.
- If VTAM is trying to recover for persistent sessions, the application is not in pending recovery state.

92 (X'5C')
VTAM is included as part of the operating system but inactive.

94 (X'5E')
The address supplied in the ACB's APPLID field lies beyond the addressable range of your application program.

95 (X'5F')
The VTAM transient being used by the application for an OPEN ACB does not match the level of VTAM.

96 (X'60')
An apparent system error occurred. Either there is a logic error in VTAM, or there is an error in your use of OPEN or CLOSE that VTAM did not properly detect. Save all applicable program listings and storage dumps, and consult IBM Service.

98 (X'62')
The APPLID length byte is incorrectly specified.

100 (X'64')
The address supplied in the ACB's PASSWD field lies beyond the addressable range of your application program.

102 (X'66')
The PASSWD length byte is incorrectly specified.

104 (X'68')
The APPLID field in the ACB identifies an application program that is defined with AUTH=PPO in its APPL definition statement. Another program with the same authorization is active. Only one program defined with AUTH=PPO can be active at a time.

106 (X'6A')
The address supplied in the ACB's vector list field lies beyond the addressable range of your application program.

108 (X'6C')
The VTAM ACB vector list length byte is incorrectly specified.

112 (X'70')
You attempted to open an ACB that is in the process of being closed. This

can occur when a VTAM application program job step or subtask is canceled or terminates abnormally. The process of closing the ACB can continue after the job step or subtask has actually terminated. Subsequently, if the job step is restarted or the subtask is reattached before the ACB closing process has been completed, an OPEN macroinstruction that is then issued for that ACB fails.

114 (X'72')
This code occurs when an OPEN ACB fails for an LU 6.2 application with VERIFY=OPTIONAL or VERIFY=REQUIRED for one of the following reasons:

- The security management product is not installed.
- The security management product is not active.
- The security management product resource class APPCLU is not active.
- The application represented by the ACB is not in the security management product Started Procedures Table.

116 (X'74')
VTAM rejected the takeover by an alternate application because the original application did not enable persistence, although it is capable of persistence.

118 (X'76')
OPEN failed because the specified application is in a recovery pending state and PERSIST=YES is not specified on the ACB that is being opened. The OPEN may also fail if the application is in pending terminate state and an active CDRSC with the same name has been found in the sysplex.

120 (X'78')
ACB option mismatch between original application and opening takeover or recovery application. One or more of the following can apply:

- MACRF mismatch—both values must be either LOGON or NLOGON; they cannot differ.
- NQNAMES mismatch—both applications must be specified as NQNAMES=YES or NQNAMES=NO; they cannot differ.
- PERSIST mismatch—both applications must be specified as PERSIST=YES.
- FDX mismatch—both applications must be specified as FDX=YES or FDX=NO; they cannot differ.
- LIMQSINT mismatch—both application APPL statements must agree in their specification of a LIMQSINT parameter; either both must specify a value or neither may. (The actual timer values specified do not need to be identical, however.)
- APPC mismatch — both application APPL statements must specify either APPC=YES or APPC=NO; they cannot differ.
- ENCR or MAC mismatch—the recovering application APPL statement must specify a security level equal to or higher than the setting in effect for the original application
- GNAME capability mismatch—the original application was supporting a generic name but the VTAM node of the recovering application is not connected to a generic resource structure or its structure name differs from the original structure name.
- SECLVL mismatch—both application APPL statements must specify the same SECLVL setting values; they cannot differ.

- VERIFY mismatch—both application APPL statements must specify the same VERIFY setting values; they cannot differ.

122 (X'7A')
The OPEN ACB by this performance monitor application failed because a performance monitor application was already active, and only one performance monitor application can be active.

124 (X'7C')
SNPS takeover request denied because the active application does not allow itself to be taken over when it is still active.

140 (X'8C')
PERFMON=YES is coded on the ACB but the application is not CNM and POA authorized.

188 (X'BC')
The ACB is in the process of being opened or closed by another request.

244 (X'F4')
The application program is not authorized for SRBEXIT=YES. A request to open an ACB whose corresponding APPL definition statement specifies SRBEXIT=YES is rejected unless the application program is APF authorized, or using key 0–7, or in supervisor state.

246 (X'F6')
NIB storage address not valid. A CNM authorized application program either failed to supply an NIB pointer in the NIB field of the ACB, or the NIB address supplied lies beyond the addressable range of the application program.

250 (X'FA')
NIB options not valid. Either an application program without CNM authorization (specified in its associated VTAM resource definition) supplied an NIB pointer in its ACB; or, if CNM authorized, the application program failed to supply valid NIB options on the NIB macroinstruction.

254 (X'FE')
Duplicate unsolicited RU routing requested. The CNM routing table indicated that this application program was to receive the same unsolicited formatted requests that were already being routed to another active CNM authorized application program. Only one application program can be actively receiving a particular type of RU (for example, RECFMS) at a time.

The OFLAGS field in the ACB is set to B'xxx1xxxx' if the ACB opens successfully.

Use the following guidelines to produce a system-independent determination of a successful or unsuccessful OPEN:

- Put 0 in register 15 before issuing OPEN.
- Issue OPEN for only one VTAM ACB at a time.
- If register 15 is 0, consider the OPEN successful.
- If register 15 is not zero, consider the OPEN unsuccessful, and examine the contents of the ACB's ERROR field.

For a CNM application program, the CID of the SSCP-LU session is returned in the CID field of the NIB specified in the ACB's PARMS=(NIB=*nib address*) operand.

OPNDST—Establish sessions (application as PLU) or recover sessions

Purpose

The OPNDST macroinstruction requests VTAM to establish a session between the application program (operating as the PLU) and an LU.

Usage

No matter how a session is initiated, OPNDST is used to accept the resulting pending-active session in which the application program acts as the PLU. OPNDST is also used to initiate a session with an LU and accept the resulting pending-active session in one operation. This is known as acquiring a session.

Additionally, OPNDST is used by an application program (operating as a PLU or an SLU) to restore sessions pending recovery.

OPNDST OPTCD=ACQUIRE can establish more than one session if a list of NIBs is supplied. For a detailed description of the parameter specification and operation of the OPNDST macroinstruction, refer to "OPNDST macroinstruction" on page 89.

Before issuing the OPNDST macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

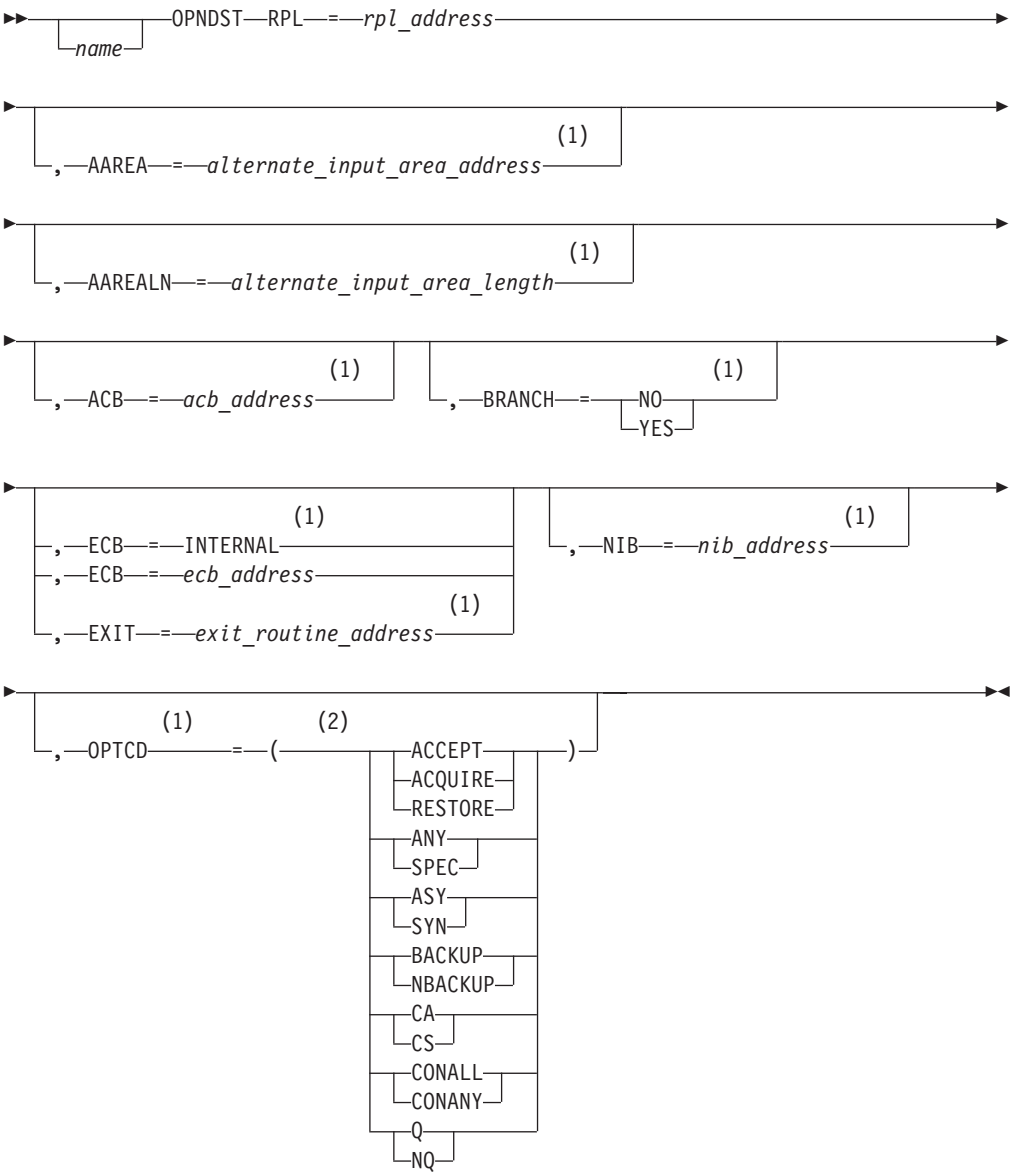
VTAM receives control from the OPNDST macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

The target of this macroinstruction might be a network-qualified name. For more details see "NIB—Create a node initialization block" on page 433.

The OPNDST macroinstruction can specify whether a session request is for a primary or backup XRF session. It must also specify a correlation ID in the NIB BNDAREA for OPNDST ACQUIRE on a primary XRF session. This correlation ID allows VTAM to pair up a primary XRF with a backup XRF session. Note that the negotiable BIND is not supported for XRF sessions.

When the application is participating as a member of a generic resource, it can specify whether it wants to own the affinity for the new session. The application does this using the LUAFFIN keyword of the NIB macroinstruction. For more details see "NIB—Create a node initialization block" on page 433.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing OPNDST is to perform.

The following RPL operands apply to the OPNDST macroinstruction:

AAREA=*alternate_input_area_address*
Specifies an address into which a response to a negotiable BIND is placed. AAREA is ignored if AAREALN is 0 or if PROC=NNEGBIND in the NIB.

When OPTCD=RESTORE, AAREA points to the area that the application program allocates to hold the recovery data that an application may use to resynchronize the sessions during session recovery.

AAREALN=*alternate_input_area_length*
Specifies the length of the address pointed to by AAREA. A valid BIND response can be up to 255 bytes in length.

When OPTCD=RESTORE, the area obtained must be large enough to hold all the recovery data. If it is not, a special condition results (RTNCD,FDB2)=(X'00',X'05'). OPNDST calculates the length needed and returns that value to the RECLEN field. The sessions are not restored until the supplied area is large enough.

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing OPNDST.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) OPNDST operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) OPNDST operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

NIB=*nib_address*
Indicates the address of the NIB whose parameters are to be associated with the session to be established. Refer to the NIB macroinstruction description in this chapter to see which NIB parameters are valid for OPNDST.

When OPTCD=RESTORE, the address points to the NIB list created by either the INQUIRE command or the application. The OPNDST macroinstruction uses this information to determine which sessions to restore.

OPTCD=ACCEPT
When ACCEPT is set, a session is established with a logical unit with which a pending active session exists.

If PARMS=(NQNAMES=YES) on the ACB macroinstruction, and if the NIB is specified with a network identifier in the NIBNET field, the network identifier is used along with the LU name in NIBSYM to find the queued CINIT.

OPTCD=ACQUIRE
VTAM issues session-initiation requests for the LUs indicated in the NIBs specified by the RPL's NIB field. Only sessions with available LUs are established.

If PARMS=(NQNAMES=YES) on the ACB macroinstruction, and if the NIB is specified with a network identifier in the NIBNET field, the network identifier is used along with the LU name in NIBSYM to build a session initiation request.

OPTCD=RESTORE
The PLU and SLU applications that are capable of persistence use the RESTORE option to recover sessions pending recovery. The application program can use the information found in control vector hex 29 to recover the individual sessions.

The NIB will contain a pointer to the restore parameter list. The restore parameter list indicates where to locate:

- Control vector 29, see Table 110 on page 747 for more information.
- BIS data (for LU 6.1 and LU 6.2)
- BID data (for LU 6.1 and LU 6.2)
- FMH-5 data (for LU 6.2).

For more information about BIS, BID, and FMH-5, data see “Data tracking” on page 137.

For XRF requests, the NIB BNDAREA field must point to a BIND specifying a control vector or vectors included with the XRF session activation control vector (including correlation ID) initialized appropriately. In addition, OPTCD=BACKUP must be specified on the OPNDST OPTCD=ACQUIRE if the session establishment request is for a backup XRF session. The request fails if the SLU cannot support XRF.

OPTCD=BACKUP and OPTCD=NBACKUP do not apply for OPNDST OPTCD=ACCEPT and are ignored. However, if a SIMLOGON OPTCD=BACKUP is issued, it drives a LOGON exit routine that contains OPNDST OPTCD=ACCEPT.

Note: When using OPNDST OPTCD=(ACQUIRE,ASY) an application program task could be suspended if an RPL exit routine is not provided. See “Initializing a session” on page 44 for details.

OPTCD=BACKUP
OPTCD=NBACKUP
This parameter applies only when trying to initiate an XRF backup session. A backup XRF session must not be requested by the application until the OPNDST for the primary XRF session has been posted complete. BACKUP indicates that the OPNDST request is to initiate a backup XRF session. OPTCD=NBACKUP is the default and results in a session initiation request that does not specify a backup XRF session.

OPTCD=CONALL

OPTCD=CONANY
When CONANY is set and OPNDST OPTCD=ACQUIRE is issued, a session is established with the first available logical unit of the NIB list indicated in the RPL's NIB field. When CONALL is set, a session is established for all the available LUs in the list.

OPTCD=CA
OPTCD=CS
Specifies the initial setting of the session's CS-CA mode for all data types (DFSYN, DFASY, and RESP). When CA is set, data obtained on a session can satisy a RECEIVE OPTCD=ANY or RECEIVE OPTCD=SPEC macroinstruction. When CS is set, only RECEIVE (OPTCD=SPEC) macroinstructions can obtain data.

OPTCD=Q
OPTCD=NQ
This option applies only when OPTCD=ACCEPT is in effect. When OPTCD=ACQUIRE is in effect, this option is ignored. When Q is set, VTAM queues the OPNDST until a session is initiated that results in a CINIT that OPNDST can accept. When NQ is set, VTAM terminates the OPNDST macroinstruction immediately if there is no suitable pending active session to accept.

OPTCD=SPEC
OPTCD=ANY
When SPEC is set (used only in conjunction with ACCEPT), a specific session (designated by the NIB's NAME or CID field) is established after the session becomes pending active. When ANY is set, a session is established for any pending active session that is created.

OPTCD=SYN
OPTCD=ASY
If SYN option code is set,control is returned to the application program when the OPNDST operation has completed. If ASY option code is set, control is returned as soon as VTAM has accepted the request. Once the OPNDST operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the OPNDST operation, you should not use the SYN option if suspending the OPNDST-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

Examples

Note: To avoid obscuring the differences between the different types of OPNDST, the same technique is used to set the RPL fields in each example (namely, inserting RPL-modifiers on the OPNDST macroinstruction). RPL fields could just as well have been set with the MODCB macroinstruction, with assembler instructions, or with the RPL macroinstruction itself.

Case 1: This is an “ACQUIRE CONALL” OPNDST.

```
ACQALL   OPNDST RPL=RPL1,NIB=NIBLST1,ACB=ACB1,                    C
               OPTCD=(ACQUIRE,CONALL)
         .
         .
```

```
         .
NIBLST1  NIB   NAME=LU1,LISTEND=NO
         NIB   NAME=LU2,LISTEND=NO
         NIB   NAME=LU2,LISTEND=YES
```

ACQALL establishes sessions between each of the available logical units of NIBLST1 and the application program represented by ACB1.

Note: Two sessions are established with LU2 only if both the application program and LU2 are capable of parallel sessions.

Case 2: This is an "ACQUIRE CONANY" OPNDST.

```
ACQANY   OPNDST RPL=RPL2,NIB=NIBLST2,ACB=ACB1,                        C
               OPTCD=(ACQUIRE,CONANY)
         .
         .
         .
NIBLST2  NIB   NAME=LUX,LISTEND=NO
         NIB   NAME=LUY,LISTEND=NO
         NIB   NAME=LUZ,LISTEND=YES
```

ACQANY establishes a session between one of the LUs of NIBLST2 (LUX, LUY, or LUZ) and the application program. The CON and CID fields are set in the NIB representing the established session. RPLARG field also contains the CID of the established session.

Case 3: This is an "ACQUIRE One NIB" OPNDST.

```
ACQONE   OPNDST RPL=RPL3,NIB=NIB3,ACB=ACB1,                           C
               OPTCD=ACQUIRE
         .
         .
         .
NIB3     NIB   NAME=LU35,LISTEND=YES
```

ACQONE establishes a session with LU35, if LU35 is available.

Case 4: This is an "ACCEPT ANY" OPNDST.

```
ACPTANY  OPNDST RPL=RPL4,NIB=NIB6,ACB=ACB1,                           C
               OPTCD=(ACCEPT,ANY,NQ)
         .
         .
         .
NIB6     NIB   LISTEND=YES
```

ACPTANY establishes a session with any one LU for which a session-initiation request has been issued that resulted in a CINIT queued to the application program. The symbolic name of this LU (along with the CID of the session) is placed in NIB6. Because NQ is specified, the request is posted complete with (RTNCD,FDB2)=(X'00',X'06') if no such queued CINIT exists.

Case 5: This is an "ACCEPT SPEC" OPNDST.

```
ACPTSPC  OPNDST RPL=RPL5,NIB=NIB7,ACB=ACB1,                           C
               OPTCD=(ACCEPT,SPEC,Q)
         .
         .
         .
NIB7     NIB   NAME=LU77,LISTEND=YES
```

ACPTSPC establishes a session between LU77 and the application program when a CINIT is received for such a session.

Completion information

The OPNDST operation is successfully completed when the specified sessions are established. (For selective and required cryptographic sessions, this is after the receipt of the response to CRV.)

After the OPNDST operation is completed, the following NIB fields are set:

- The CID of the session is placed in the CID field. If recovering for persistent sessions, the CID will be different than the CID of the session before the failure.
- The CON field is set to YES if the session was established; otherwise, it is set to NO. This field can be examined by coding CON=YES on a TESTCB macroinstruction.

  Note: You must check the CON field to determine whether the session was actually established. A return code indication is not sufficient.
- If the ACCEPT option was in effect, the symbolic name of the SLU is placed in the NAME field.
- If the ACCEPT option was in effect and NQNAMES=YES is specified, the symbolic name of the network containing the SLU is placed in the NETID field.
- Certain characteristics of the SLU are indicated in the DEVCHAR field. The DEVCHAR codes are explained in Appendix E, "Control block formats and DSECTs," on page 739.
- An indicator (NIBNACLQ) is set indicating whether the pending active session still exists. See the description of (RTNCD,FDB2)=(X'08',X'00') in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651, for an example of the use of this indicator.

The following fields are set in the RPL:

- The value 23 (decimal) is set in the REQ field, indicating an OPNDST request.
- If only one session has been established, the CID is placed in the ARG field.
- The address of the NIB or NIB list (as supplied by you in the NIB field) is returned in the AREA field. The NIB field is overlaid when the CID is placed in the ARG field, because the NIB and ARG fields occupy the same physical location in the RPL.
- If a negotiable BIND had been sent (PROC=NEGBIND in the NIB), ARECLEN specifies the length of the received positive response to that BIND, which has been moved into the area pointed to by AAREA. If ARECLEN is larger than AAREALN, the response is truncated to fit into AAREA.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651. If the return codes indicate that the OPNDST has failed, do not issue a CLSDST for the session, as it is not established. However, if OPNDST OPTCD=ACCEPT was used, and the NIB indicates that a pending active session still exists, then the OPNDST should be retried, or CLSDST should be issued to terminate the pending active session.
- The SSENSEI, SSENSMI, and USENSEI fields are set if (RTNCD,FDB2)=(X'10',X'01') (OPNDST for a logical unit failed).

  Note: The USERFLD field is not set for OPNDST.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

OPNSEC—Establish a session, application program acts as the SLU

Purpose

The OPNSEC macroinstruction is issued by a VTAM application program that has received a BIND request in a SCIP exit routine, and wants to respond positively to the BIND and thus establish a session with the PLU that sent the BIND. The application program that issues OPNSEC is the SLU in the session.

For a detailed description of the operation of the OPNSEC macroinstruction, refer to "OPNSEC macroinstruction" on page 96.

Usage

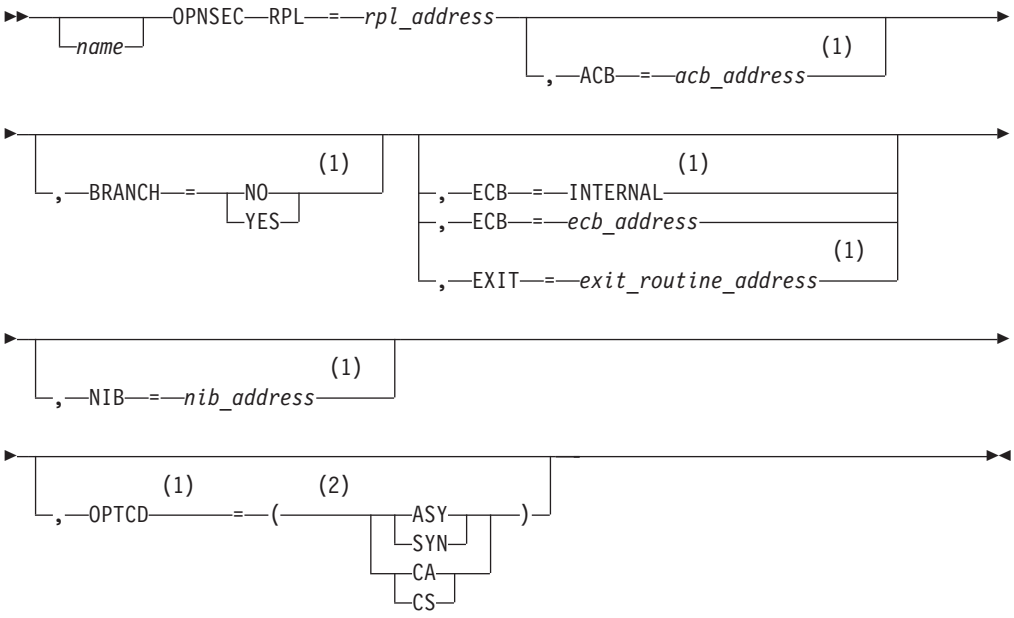
Before issuing the OPNSEC macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from the OPNSEC macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

When the OPNSEC successfully completes, VTAM indicates in the NIB the type of encryption (selective or required) that is specified by the application.

When the application is participating as a member of a generic resource, it can specify whether it wants to own the affinity for the new session. The application does this using the LUAFFIN keyword of the NIB macroinstruction. For more details see "NIB—Create a node initialization block" on page 433.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing OPNSEC is to perform.

The following RPL operands apply to the OPNSEC macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing OPNSEC.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) OPNSEC operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) OPNSEC operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

NIB=*nib_address*
Indicates the NIB whose parameters are to be associated with the session requested by the BIND request. Refer to the NIB macroinstruction description in this chapter to see which NIB parameters are valid for OPNSEC.

OPTCD=CA
OPTCD=CS
Specifies the initial CA-CS setting of all input types (DFSYN, DFASY, and RESP) for the session. When CA is set, input received satisfies a RECEIVE OPTCD=ANY or OPTCD=SPEC. When CS is set, input received satisfies only a RECEIVE OPTCD=SPEC.

OPTCD=SYN
OPTCD=ASY
When SYN option code is set, control is returned to the application program when the OPNSEC operation has been completed. If ASY option code is set, control is returned as soon as VTAM has accepted the request. After the OPNSEC operation has been completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the OPNSEC operation, you should not use the SYN option if suspending the OPNSEC-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

Examples

```
OPNS1    OPNSEC RPL=RPL1,OPTCD=CA,NIB=NIB1                          C
:
RPL1     RPL   AM=VTAM
NIB1     NIB   NAME=APPLPRI
```

Executing OPNS1 signifies acceptance of the session parameters received as a result of a BIND request and agreement to act as the secondary end of the session with APPLPRI.

Completion information

For selective and required cryptographic sessions, the OPNSEC operation is completed when the response to CRV is sent. For other sessions, completion occurs when the positive response to BIND is sent. Be aware that the RPL exit routine can be driven after other exit routines dealing with the same session are driven (for example, SCIP with start data traffic).

After the OPNSEC macroinstruction is completed, the following fields in the NIB are set:

- The CID field contains the 4-byte communication identifier.
- The CON field is set to YES if the session is established; otherwise, it is set to NO.
- The NAME field contains the symbolic name of the PLU from the BIND request.
- If NQNAMES=YES, the NETID field contains the network identifier of the PLU from the BIND request.

After the OPNSEC operation is completed, the following RPL fields are set:

- The value 42 (decimal) is set in the REQ field, indicating an OPNSEC request.
- The CID is placed in the RPLARG field.
- The address of the NIB that was specified in the NIB field is returned in the AREA field. The NIB field is overlaid by the ARG field that contains the CID. The NIB and the ARG fields occupy the same space in the RPL.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

  If the macroinstruction returns an error code, the SSENSEI, SSENSMI, and USENSEI fields can be set indicating system-sense information, system-sense modifier, and user-sense information. More information about these fields can be found in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

  Note: The USERFLD is not set for OPNSEC.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

RCVCMD—Receive a message from VTAM

Purpose

After an application program issues a VTAM operator command (VARY, DISPLAY, MODIFY, or REPLY) using a SENDCMD macroinstruction, a RCVCMD macroinstruction is used to receive the requested information. In addition, unsolicited VTAM messages, such as those indicating an unexpected failure in the network, can be received with this macroinstruction.

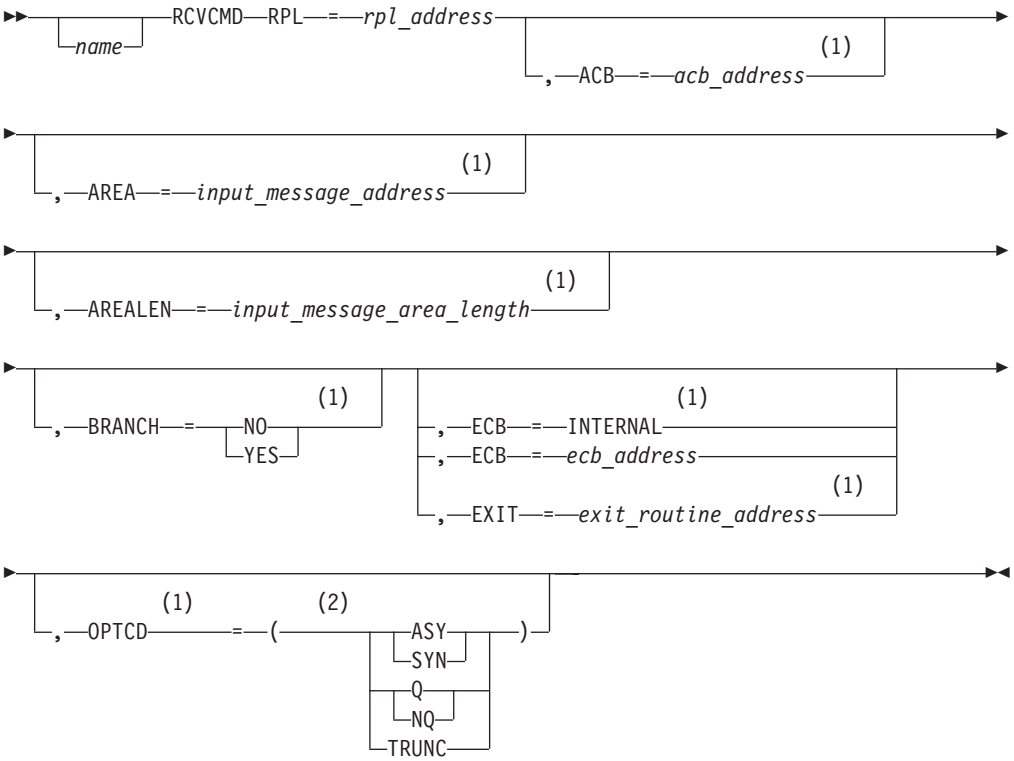
Usage

Before issuing the RCVCMD macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

For information on writing an application program that can issue VTAM operator commands and receive VTAM messages, see Appendix L, "Program operator coding requirements," on page 875.

VTAM receives control from the RCVCMD macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing RCVCMD is to perform.

The following RPL operands apply to the RCVCMD macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing RCVCMD.

AREA=*input_message_address*
Must contain the address of the area in the application program where the incoming message header, optional message identification, and the message text are to be placed. After the message has been moved to this area, the RPL's RECLEN field is set by VTAM with the total number of bytes of received data that have been moved into the AREA field. The AREA field is ignored if AREALEN=0.

AREALEN=*input_message_area_length*
Contains the length (in bytes) of the message area pointed to by AREA. The length specified should be 8 bytes longer than the longest message text anticipated to provide enough space for the message header and optional reply ID. The AREA field must be at least 4 bytes and no longer than 130 bytes.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) RCVCMD operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) RCVCMD operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

OPTCD=Q
OPTCD=NQ
Indicates the action to be taken if no input is available when the RCVCMD macroinstruction is executed. OPTCD=Q means that the macroinstruction is completed when the requested input eventually arrives. OPTCD=NQ means that the macroinstruction is completed immediately with (RTNCD,FDB2)=(X'00',X'06') if the input is not available.

Note: After a CLOSE macroinstruction fails because operator messages are still queued for the application program, RCVCMD macroinstructions can still be issued, but they are not queued. After the last message is received, RCVCMD is posted with (RTNCD,FDB2)=(X'14',X'70') if OPTCD=Q is specified, or (RTNCD,FDB2)=(0,6) if OPTCD=NQ is specified. This indicates that no more messages are queued. CLOSE can be reissued.

OPTCD=SYN
OPTCD=ASY
If SYN option code is set, control is returned to the application program when the RCVCMD operation has completed. If ASY option code is set, control is returned as soon as VTAM has accepted the request. Once the RCVCMD operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the RCVCMD operation, you should not use the SYN option if suspending the RCVCMD-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

OPTCD=TRUNC
Indicates that input data that is too lengthy is truncated whenever the RCVCMD macroinstruction is issued.

Examples

```
RCVCMD1   RCVCMD RPL=RPL1,AREA=MSGBUF,AREALEN=126,                    C
               OPTCD=(TRUNC,Q)
```

RCVCMD1 is completed when an incoming message is received from VTAM. After RCVCMD1 is completed, the application program can examine the contents of MSGBUF to determine the message received. Any messages that exceed 126 bytes are truncated to 126 bytes.

Completion information

The RCVCMD operation is successfully completed when the message or reply is received, the data (if any) is placed in the input area, and the appropriate information is set in the RPL. If NQ is specified and no input is available, RCVCMD is completed immediately.

After the RCVCMD operation is completed, the following RPL fields can be set by VTAM:

- The value 40 (decimal) is set in the REQ field, indicating an RCVCMD request.
- The RECLEN field indicates the length of the message placed in the input area pointed to by the AREA field. The length specified includes 4 bytes for the header and 4 additional bytes (if required) for the reply identifier. The reply-requested bits in the status field of the header can be tested to determine if this field is present.

  If a reply is requested for this message, the AAREALN field indicates the maximum length the reply can be. This reply length includes header, message ID, and text.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651.

Registers 0 and 15 are also set as indicated in Chapter 9, “Handling errors and special conditions,” on page 277.

RECEIVE—Receive input on a session

Purpose

The RECEIVE macroinstruction moves a data request received by VTAM into an input area in the application program or sets RPL fields to indicate receipt of a data-flow-control request or receipt of a response. The data request, data-flow-control request, or response was previously sent from a logical unit (on an LU-LU session) or from the SSCP (on the SSCP-LU session, if using the CNM interface). If data is received, it is placed in the input area designated by the application program.

Usage

Figure 87 illustrates the major options for a RECEIVE macroinstruction. Figure 39 on page 179 illustrates how RECEIVE macroinstructions are completed by VTAM. See Appendix A, "Summary of control block field usage," on page 633 for information about which RPL fields are automatically set by VTAM.

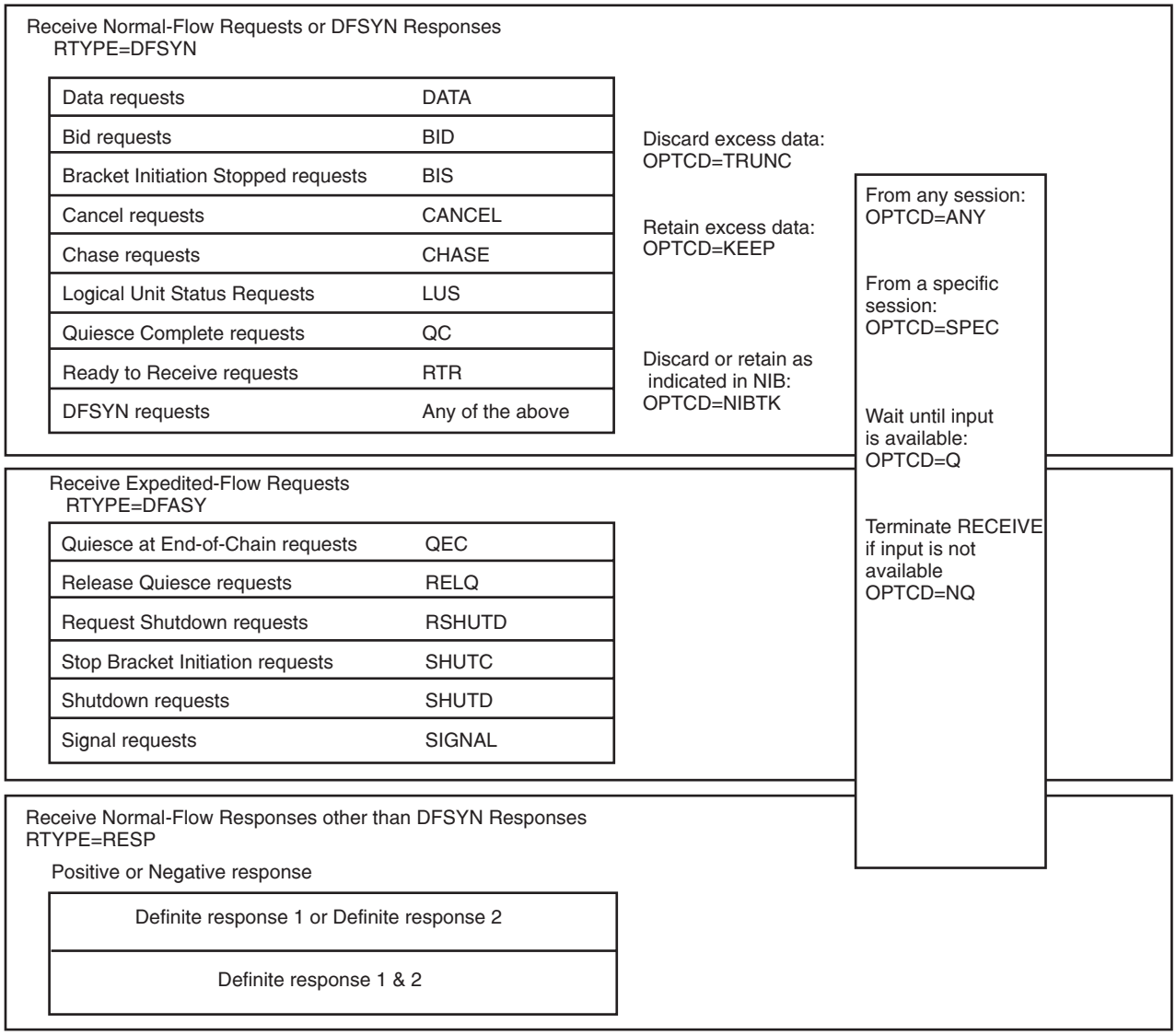


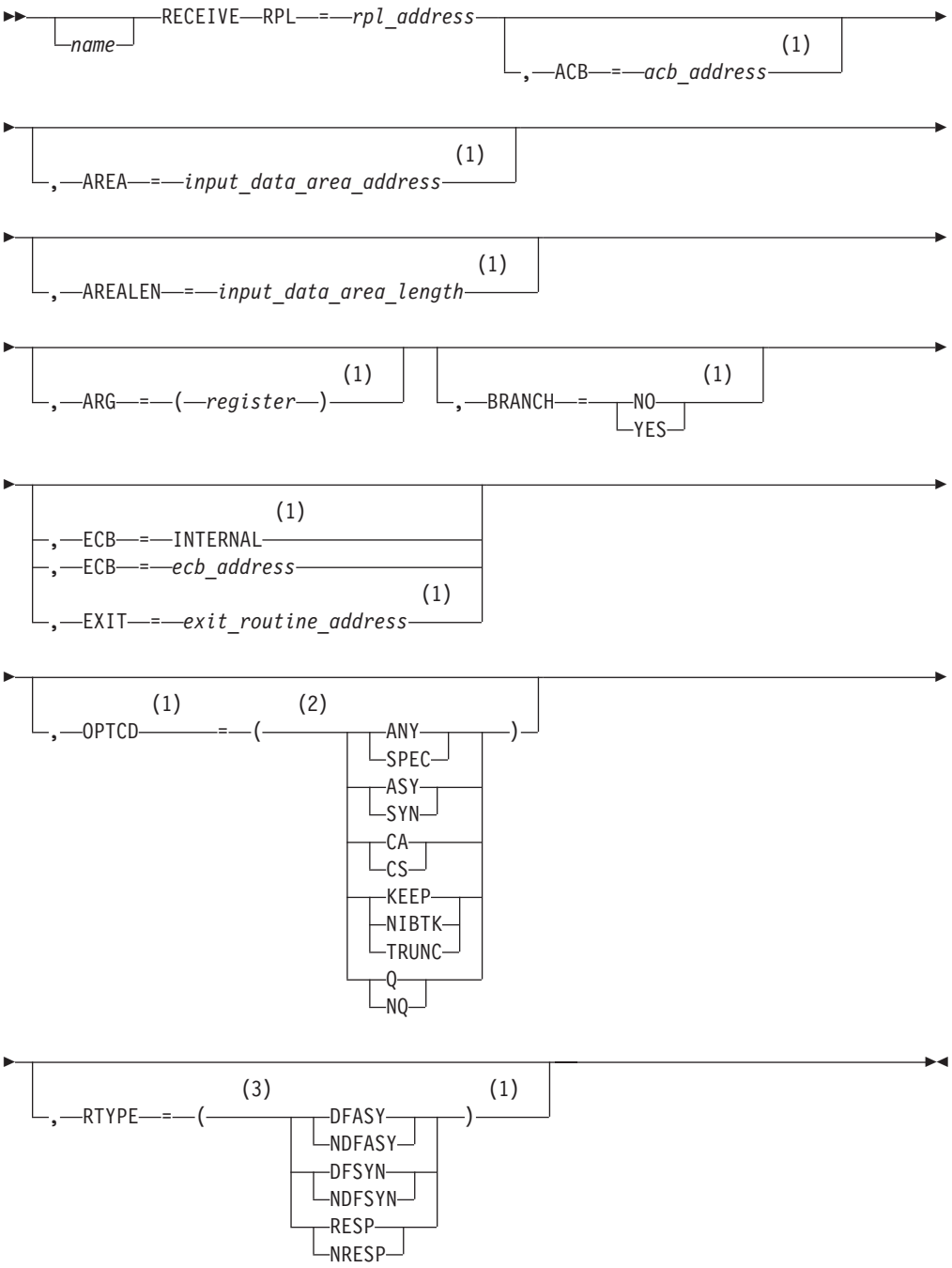
Figure 87. Major options for a RECEIVE macroinstruction

The application program designates which types of input (DFSYN, DFASY, or RESP) can cause the RECEIVE macroinstruction to be completed (any combination can be selected). Refer to "DFSYN, DFASY, and RESP types of RUs" on page 160 for details of which RUs are included in each of these types of input. Only one type of input can satisfy a particular RECEIVE macroinstruction.

Before issuing the RECEIVE macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from the RECEIVE macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

3 You can code more than one suboperand on RTYPE, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing RECEIVE is to perform.

The following RPL operands apply to the RECEIVE macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing RECEIVE.

AREA=*input_data_area_address*
The AREA field must contain the address of the area in the application program where an incoming data request is to be placed. The associated request header (RH) is not put in this area, but instead is set in various RPL fields. If a data-flow-control request or data-flow-control response is received instead of a data request or data response, the CONTROL field is posted with a value other than CONTROL=DATA, and the input data area is not used. Also, if a data response is received, the AREA field is not used. After a data request has been moved, the RPL's RECLEN field is set by VTAM with the total number of bytes of data received by VTAM. The AREA field is ignored if AREALEN=0.

AREALEN=*input_data_area_length*
The AREALEN field contains the length (in bytes) of the data area pointed to by AREA. VTAM uses this value to determine if there is too much incoming data to fit. If there is too much, the action indicated by the TRUNC-KEEP-NIBTK option code is taken. (Refer to the TRUNC-KEEP-NIBTK option code described in this section.)

AREALEN=0 with OPTCD=KEEP can be used to determine the amount of incoming data (the total length is set in RECLEN). A data area can be obtained and the RECEIVE macroinstruction reissued. AREALEN=0 with OPTCD=TRUNC can be used to eliminate unwanted data requests that are queued for the application program.

ARG=(register)
If a specific session is to be read (OPTCD=SPEC), that session must be identified by a CID. The ARG operand specifies the register containing the CID of the session. If the ARG operand is not specified, the CID in the RPLARG field is used.

Note: If your application uses the RPL DSECT, IFGRPL, you must clear the RPLNIB bit if a CID is being inserted into the RPLARG field.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than

under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) RECEIVE operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) RECEIVE operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

OPTCD=CA
OPTCD=CS
When the RECEIVE operation is completed successfully (that is, (RTNCD,FDB2)=(X'00',X'00'), normal completion, or (RTNCD,FDB2)=(X'04',X'03'), exception request received, or (RTNCD,FDB2)=(X'04',X'04'), negative response received) and the NIB used to establish the session specifies PROC=RPLC, the session is placed into continue-any mode (CA) or into continue-specific mode (CS) based on the setting of the option. This mode determines whether the next RECEIVE OPTCD=ANY can be satisfied by the next input on the session.

This option code has no effect for any other RTNCD and FDB2 settings. In particular, (RTNCD,FDB2)=(X'00',X'06'), no input available, does not cause any change to the session's CA-CS mode.

With the exception of a RECEIVE that is completed with RTYPE=(DFSYN,RESP), the switch of continue-any and continue-specific modes applies only to the type of input (specified by the RTYPE field) that actually satisfied the RECEIVE. In a RECEIVE that is completed with RTYPE=(DFSYN,RESP), the mode switch applies only to DFSYN input.

OPTCD=CONDCS
When the RECEIVE operation is completed successfully (that is, (RTNCD,FDB2)=(X'00',X'00'), normal completion, or (RTNCD,FDB2)=(X'04',X'03'), exception request received, or (RTNCD,FDB2)=(X'04',X'04'), negative response received) and the NIB used to establish the session specifies PROC=RPLC, the session is placed into continue-any mode (CA) or into continue-specific mode (CS), or conditionally into CS mode (CONDCS) if more of the input data remains based on the setting of the option. This mode determines whether the next RECEIVE OPTCD=ANY can be satisfied by the next input on the session.

For VM CONDCS, more of the input data is considered to be remaining if RECLEN > AREALEN and KEEP was indicated (see OPTCD=KEEP). If no

input data remains, the session is left in continue-any mode. To ensure that the application receives the input data in the correct order, EXIT should also be specified on the RECEIVE.

This option code has no effect for any other RTNCD and FDB2 settings. In particular, (RTNCD,FDB2)=(X'00',X'06'), no input available, does not cause any change to the session's CA-CS mode.

With the exception of a RECEIVE that is completed with RTYPE=(DFSYN,RESP), the switch of continue-any and continue-specific modes applies only to the type of input (specified by the RTYPE field) that actually satisfied the RECEIVE. In a RECEIVE that is completed with RTYPE=(DFSYN,RESP), the mode switch applies only to DFSYN input.

OPTCD=Q
OPTCD=NQ

Indicates the action to be taken if no input (of the type specified by the RTYPE operand) is available when the macroinstruction is executed. OPTCD=Q means the macroinstruction is to be completed when the appropriate input eventually arrives. OPTCD=NQ means that the macroinstruction is to be completed immediately with (RTNCD,FDB2)=(X'00',X'06') if the input is not available. No CA-CS switch is done in this case.

OPTCD=SPEC
OPTCD=ANY

Indicates whether the RECEIVE macroinstruction can be satisfied only by input from a specific session (SPEC) or whether it can be satisfied by input from any session that is in continue-any mode (ANY).

When OPTCD=SPEC is used, the session's CID must be in the RPL when the macroinstruction is executed. When OPTCD=ANY is specified, input from a session in continue-any mode can satisfy a RECEIVE issued with RTYPE=DFASY or RTYPE=RESP only if PROC=NDFASYX or PROC=NRESPX (respectively) is specified in the NIB and if there is no outstanding RECEIVE OPTCD=SPEC for this session. See the descriptions of DFASY and RESP exit routines in “Explicit RECEIVEs and EXLST exit routines” on page 178.

RECEIVE OPTCD=ANY can be issued even when no LU-LU sessions have been established with the application program if the application program has opened an ACB. The RECEIVE is queued until one or more sessions are established and input arrives on any of the sessions in continue-any mode. If an outstanding RECEIVE OPTCD=ANY has not been completed and the application program issues CLSDST and TERMSESS macroinstructions terminating its LU-LU sessions but does not close the ACB, the RECEIVE does not have to be reissued after a subsequent OPNDST or OPNSEC is issued.

At the completion of the RECEIVE macroinstruction, the ARG field contains the CID of the session whose input satisfied the RECEIVE.

OPTCD=SYN
OPTCD=ASY

If the SYN option code is set, control is returned to the application program when the RECEIVE operation has been completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. Once the RECEIVE operation has been completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the RECEIVE operation, you should not use the SYN option if suspending the RECEIVE-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

OPTCD=TRUNC
OPTCD=KEEP
Indicates whether overlength input data is to be truncated (TRUNC) or kept (KEEP), or whether the PROC=TRUNC or PROC=KEEP setting in the session's NIB is to be used to determine whether the input is to be truncated or kept.

Overlength input data is a data request unit whose length exceeds the value set in the AREALEN field of the RECEIVE macroinstruction's RPL. When overlength data is truncated, the macroinstruction is completed and the excess data is lost.

When overlength data is kept, the macroinstruction is completed normally, and RECLEN is set to indicate the total amount of data received by VTAM. One or more additional RECEIVE macroinstructions are required to obtain the excess data. After each RECEIVE, the value of RECLEN is decreased by the amount of data received. When AREALEN=0 is set and OPTCD=KEEP is specified, the entire input is kept. For an example of OPTCD=KEEP, see "Handling overlength input data" on page 182.

RTYPE
Indicates the types of input that can satisfy this RECEIVE macroinstruction. Refer to "DFSYN, DFASY, and RESP types of RUs" on page 160 for a detailed explanation of these types of input. They are summarized in Figure 39 on page 179.

The negative settings (NDFSYN, NDFASY, and NRESP) indicate that the corresponding type of input cannot satisfy the RECEIVE macroinstruction.

Examples

```
RCV1     RECEIVE RPL=RPL1,AREA=INBUF,AREALEN=128,                  C
               RTYPE=(DFSYN,DFASY,NRESP),                          C
               OPTCD=(ANY,Q,NIBTK)
```

RCV1 is completed when an incoming request (normal-flow or expedited-flow) is available from any session that is in CA mode for that RTYPE. Assuming that the NIB specifies PROC=ORDRESP, RCV1 can be also completed by a response that causes the RECEIVE to be completed with RTYPE=(DFSYN,RESP) and RESPOND=(*x*,*x*,*x*,QRESP). Other responses cannot cause RCV1 to be completed. After RCV1 is completed, the application program can examine the control field of RPL1 to determine the type of input received. If data is received (CONTROL=DATA and RTYPE=DFSYN), the data is placed in INBUF. The TRUNC-KEEP setting in the NIB used to establish the session determines what is done with any data that exceeds 128 bytes.

Completion information

A RECEIVE operation is successfully completed when the request or response has been received, the data (if any) has been placed in the input data area, and the appropriate information has been set in the RPL. If NQ is specified and no input is available, RECEIVE is completed immediately with (RTNCD,FDB2)=(X'00',X'06').

After the RECEIVE operation is completed, the following RPL fields can be set by VTAM:

- The value 35 (decimal) is set in the REQ field indicating a RECEIVE request.
- If RECEIVE was issued with OPTCD=ANY, the ARG field contains the CID of the session whose input caused the macroinstruction to be completed successfully: (RTNCD,FDB2)=(X'00',X'00'), (X'04',X'03'), or (X'04'X'04'). No CID is available if the RECEIVE OPTCD=ANY did not complete successfully. If RECEIVE was issued with OPTCD=SPEC, the ARG field still contains the CID that was placed there prior to the execution of the macroinstruction.
- The RTYPE field indicates the type of input that satisfied the RECEIVE macroinstruction. RTYPE can be set to DFSYN, DFASY, RESP, or (DFSYN, RESP). Other RPL fields can be set depending on the type of received input, as summarized in Figure 181 on page 851. When the macroinstruction is completed, the RPL's RTYPE field indicates the type actually received.
- The 3-byte RH field in the RPL (labeled RPLURH in the RPL DSECT) and the RPL fields related to RH indicators (for example, those associated with STYPE and RESPOND) are set from the input RH. Refer to "Operation for inbound RUs" on page 198 for details.
- The RECLEN field indicates the number of bytes of data (CONTROL=DATA) received by VTAM. This same value is set in RECLEN whether KEEP or TRUNC is in effect. VTAM has moved as much of this data as possible into the input data area pointed to by the AREA field. If KEEP is in effect and the value in the RECLEN field exceeds the value in the AREALEN field, excess data present can be obtained with more RECEIVE macroinstructions. The value in RECLEN decreases by an amount equal to the amount of data moved by each RECEIVE macroinstruction.
- The SEQNO field contains the sequence number of the request or response.
- The RESPOND field indicates the type of response that has been received (if RTYPE=RESP or (DFSYN,RESP)) or the type of response that the LU expects in return (if RTYPE=DFSYN or RTYPE=DFASY).

  When a response is received, the RESPOND field for each RPL used to receive the request (if more than one is used) indicates the following:

RESPOND=(*x*,*x*,*x*,QRESP) and, in the NIB, PROC=ORDRESP	All normal-flow (DFSYN) requests sent by the LU before this response have been received by the application program.
RESPOND=(*x*,*x*,*x*,NQRESP) or, in the NIB, PROC=NORDRESP	This response can be received out of order with previously sent normal-flow requests.
RESPOND=(EX,FME,RRN,*x*)	This is a negative response with response type 1 and 2 set.
RESPOND=(EX,FME,NRRN,*x*)	This is a negative response with response type 1 set.
RESPOND=(EX,NFME,RRN,*x*)	This is a negative response with response type 2 set.
RESPOND=(EX,NFME,NRRN,*x*)	Not valid.
RESPOND=(NEX,FME,RRN,*x*)	This is a positive response with response type 1 and 2 set.
RESPOND=(NEX,FME,NRRN,*x*)	This is a positive response with response type 1 set.
RESPOND=(NEX,NFME,RRN,*x*)	This is a positive response with response type 2 set.
RESPOND=(NEX,NFME,NRRN,*x*)	Not valid.

When a request is received, the RESPOND field indicates the type of response that is required (the value in the RTNCD field indicates whether the request was received by the application program successfully). For certain LU types, other rules (not detailed here) can apply.

RESPOND=(*x*,*x*,*x*,QRESP)	A DFSYN response is required. Return the appropriate response along with other normal-flow (DFSYN) requests. The QRESP indicator must be set on in the response.
RESPOND=(*x*,*x*,*x*,NQRESP)	A RESP response is required. Return the appropriate response along with other responses. NQRESP must be set on in the response.
RESPOND=(EX,FME,RRN,*x*)	If the request is processed successfully, no response is sent; if the request is not processed successfully, return a negative response with response type 1 and 2 set.
RESPOND=(EX,FME,NRRN,*x*)	If the request is processed successfully, no response is sent; if the request is not processed successfully, return a negative response with response type 1 set.
RESPOND=(EX,NFME,RRN,*x*)	If the request is processed successfully, no response is sent; if the request is not processed successfully, return a negative response with response type 2 set.
RESPOND=(EX,NFME,NRRN,*x*)	Not valid.
RESPOND=(NEX,FME,RRN,*x*)	Return a positive or negative response, as appropriate, with response type 1 and 2 set.
RESPOND=(NEX,FME,NRRN,*x*)	Return a positive or negative response, as appropriate, with response type 1 set.
RESPOND=(NEX,NFME,RRN,*x*)	Return a positive or negative response, as appropriate, with response type 2 set.
RESPOND=(NEX,NFME,NRRN,*x*)	Return no response.

For details about the RESPOND operand, refer to "What a response contains" on page 153.

- The USER field contains the value that was originally set in the USRFLD field of the NIB used to establish the session. This field is set only if the CID field is set.
- The CRYPT field indicates whether the normal-flow data request received was sent through the network in an enciphered format.
- The CODESEL field indicates whether the input is in the standard (STANDARD) or in some other code (ALT) agreed upon by each end of the session (such as EBCDIC or ASCII). It has meaning only for CONTROL=DATA; however, the value of the CODESEL indicator (which should be STANDARD) in any received data or data-flow-control request is set in the RECEIVE RPL.
- If VTAM receives either a data request or data response that indicates an FM header is present, OPTCD=FMHDR is set. This indicator is also set for all data-flow-control requests and their responses.
- The CHNGDIR field indicates whether the change-direction indicator is set:

CHNGDIR=(CMD,NREQ)	The change-direction indicator is set; the LU at the other end of the session was the sender and it has changed direction so that the application program can now transmit normal-flow requests.
CHNGDIR=(NCMD,REQ)	The meaning of this indicator is not defined by SNA. It should not be used because such use could produce unpredictable results.
CHNGDIR=(CMD,REQ)	Both indicators are set.
CHNGDIR=(NCMD,NREQ)	Neither indicator is set.

- The value of the CHNGDIR CMD indicator is set in the RECEIVE RPL for all received data and data-flow-control requests. The value of the CHNGDIR REQ indicator is set in the RECEIVE RPL for all received data and data-flow-control requests and responses.
- The BRACKET field indicates whether the current bracket is beginning, ending, or continuing. For example, for FM profiles in which BB and EB are used:

BRACKET=(BB,NEB)	The chain is the first of a new bracket.
BRACKET=(NBB,NEB)	The chain is a continuation of the current bracket. This setting is also present when brackets are not being used and for all requests that are not the first or only request of a chain.
BRACKET=(NBB,EB)	The chain is the end of the current bracket.
BRACKET=(BB,EB)	The chain itself constitutes an entire bracket.

- In FM profile 19, the CEB indicator is used instead of the EB indicator to indicate end-of-bracket. CEB can occur only on a last-in-chain request or only-in-chain request. For only-in-chain requests, it can be used in combination with the BB indicator in a similar fashion to EB shown in the preceding section. For last-in-chain requests, it is used alone because BB cannot be sent on a last-in-chain request.
- The BRACKET field has meaning only for data requests and certain normal-flow data-flow-control requests. However, the values in the field for any received data or data-flow-control request are set in the RECEIVE RPL.
- The CHAIN field indicates the request's relative position within the chain being sent to the application program (DFSYN requests only):

CHAIN=FIRST	The request is the first of a new chain.
CHAIN=MIDDLE	The request is a continuation of the current chain.
CHAIN=LAST	The request is the last of the current chain.
CHAIN=ONLY	The request itself constitutes an entire chain.

  This field is set when the RECEIVE is completed with CONTROL=SIGNAL.
- The SIGDATA field contains 4 bytes of signal information.
- The CONTROL field indicates whether the received request or response was data, or whether it was a data-flow-control request or response. In the latter case, CONTROL is set to the request code of the received data-flow-control request or response.

The following table shows the meanings of the allowed combinations of CONTROL and RTYPE values.

CONTROL=	RTYPE=	Meaning
DATA	DFSYN	A data request has been received.
DATA	RESP or (DFSYN,RESP)	The response to a data request has been received.
BID	DFSYN	A bid request has been received.
BID	RESP or (DFSYN,RESP)	The response to a Bid request has been received.
BIS	DFSYN	A Bracket Initiation Stopped request has been received.
BIS	RESP or (DFSYN,RESP)	The response to a Bracket Initiation Stopped request has been received.
CANCEL	DFSYN	A Cancel request has been received.
CANCEL	RESP or (DFSYN,RESP)	The response to a Cancel request has been received.
CHASE	DFSYN	A Chase request has been received.
CHASE	RESP or (DFSYN,RESP)	The response to a Chase request has been received.
LUS	DFSYN	An LU Status request has been received.
LUS	RESP or (DFSYN,RESP)	The response to an LU Status request has been received.
QC	DFSYN	A Quiesce Complete request has been received.
QC	RESP or (DFSYN,RESP)	The response to a Quiesce Complete request has been received.
RTR	DFSYN	A Ready to Receive request has been received.
RTR	RESP or (DFSYN,RESP)	The response to a Ready to Receive request has been received.
QEC	DFASY	A Quiesce at End-of-Chain request has been received.
RELQ	DFASY	A Release Quiesce request has been received.
RSHUTD	DFASY	A Request Shutdown request has been received.
SBI	DFASY	A Stop Bracket Initiation request has been received.
SHUTC	DFASY	A Shutdown Complete request has been received.
SHUTD	DFASY	A Shutdown request has been received.
SIGNAL	DFASY	A Signal request has been received.

- When a negative response, an exception request, or an Logical Unit Status (LUSTAT) request has been received, the SSENSEI, SSENSMI, and USENSEI fields are set indicating system-sense information, system-sense modifier, and user-sense information. More information about these fields can be found in Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651.
- Registers 0 and 15 are set as indicated in Chapter 9, “Handling errors and special conditions,” on page 277.

Example:

```
RCV1     RECEIVE RPL=RPL1,AREA=INBUF,AREALEN=128,                    C
               RTYPE=(DFSYN,DFASY,NRESP),                            C
               OPTCD=(ANY,Q,NIBTK)
```

RCV1 is completed when an incoming request (normal-flow or expedited-flow) is available from any session that is in CA mode for that RTYPE. Assuming that the NIB specifies PROC=ORDRESP, RCV1 can be also completed by a response that causes the RECEIVE to be completed with RTYPE=(DFSYN,RESP) and RESPOND=(*x*,*x*,*x*,QRESP). Other responses cannot cause RCV1 to be completed. After RCV1 is completed, the application program can examine the control field of RPL1 to determine the type of input received. If data is received (CONTROL=DATA and RTYPE=DFSYN), the data is placed in INBUF. The TRUNC-KEEP setting in the NIB used to establish the session determines what is done with any data that exceeds 128 bytes.

REQSESS—Initiate a session, application program acts as the SLU

Purpose

The REQSESS macroinstruction is used to initiate a session in which the application program acts as the SLU.

Usage

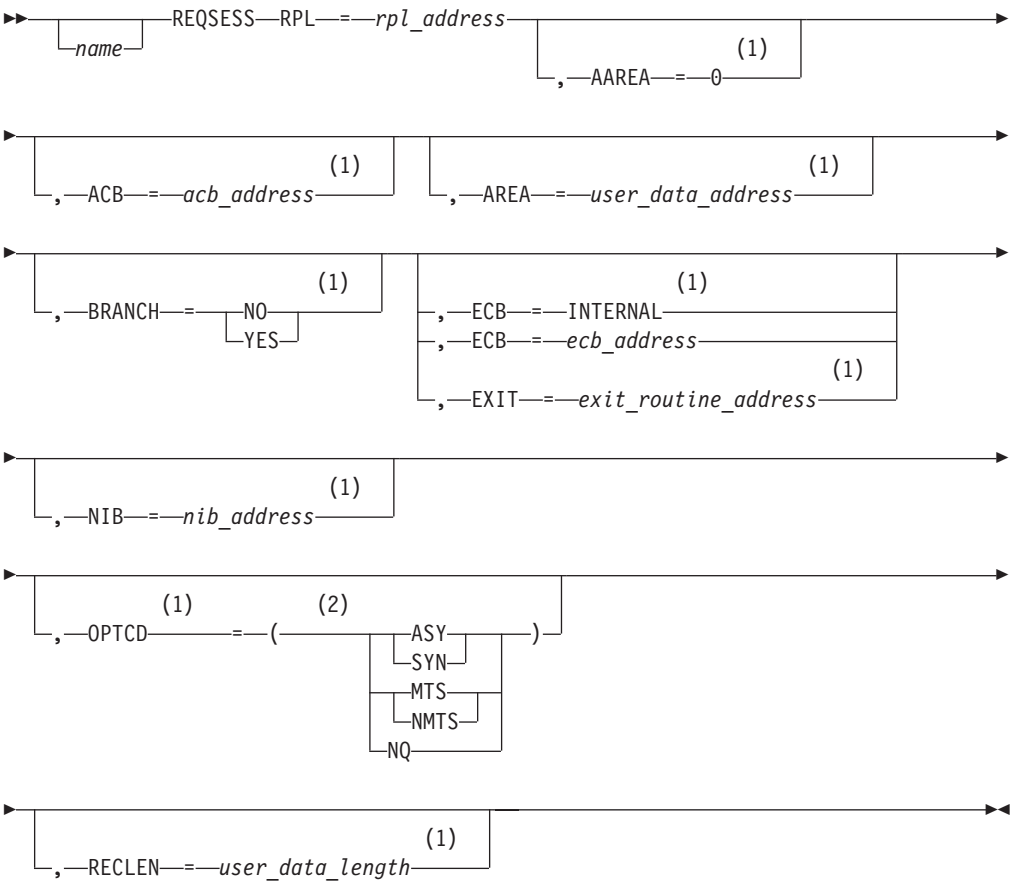
REQSESS sends an Initiate request to the SSCP which in turn sends a CINIT request to the desired PLU. If the PLU accepts the session and sends a BIND, the application program's SCIP exit routine is scheduled with the BIND. The application program can then choose to establish the session or not. Before an application program can issue the REQSESS macroinstruction, it must have issued a SETLOGON OPTCD=START, or the REQSESS fails. Also, the ACB of the application program must specify MACRF=LOGON.

Before issuing the REQSESS macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, “Summary of register usage,” on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from the REQSESS macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

If PARMS=(NQNAMES=YES) on the ACB macroinstruction, and the NIB is specified with a network identifier in the NIBNET field, the network identifier is used along with the LU name in NIBSYM to build a session initiation request.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing REQSESS is to perform.

The following RPL operands apply to the REQSESS macroinstruction:

AAREA=0
The AAREA field must be set to 0 whenever the REQSESS macroinstruction is issued. If a value other than 0 is present in this field, the REQSESS macroinstruction fails, (RTNCD,FDB2)=(X'14',X'50').

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing REQSESS.

AREA=*user_data_address*
Indicates the location of the user data that is to be sent to the PLU in the user data field of the CINIT. If this field is used, the RECLEN field must also be specified.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) REQSESS operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) REQSESS operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

NIB=*nib_address*
Indicates the NIB whose NAME field contains the symbolic name of the PLU with which the application program wants to be in session and if NQNAMES=YES, whose NIBNET field contains the network identifier in which the PLU resides. The LOGMODE field specifies the logon mode name to be used in the session being initiated. The USERFLD field can be used to specify a correlator to relate network services requests to the REQSESS. LISTEND must be set to YES. For further details, refer to "REQSESS macroinstruction" on page 95.

OPTCD=MTS
OPTCD=NMTS
If you set the MTS option code, VTAM expects to find valid MTS override data in an area pointed to by NIBMTSAR and formatted to match the ISTMTS DSECT. If you set the NMTS option code, VTAM does not expect any MTS override data.

If you do not code either OPTCD=MTS or OPTCD=NMTS on this macro, VTAM uses the value left over from the previous use of the RPL.

Note: NIBMTSAR is an alternative name for the NIBNDAR field used by the OPNDST and OPNSEC macros to point to BIND image data. Therefore, do not code both MTSAREA and BNDAREA on the same macroinstruction.

OPTCD=NQ
OPTCD=NQ is an optional parameter. If OPTCD=Q is specified (or is present in the RPL from a previous operation), an error results, (RTNCD,FDB2)=(X'14',X'50').

If the PLU with which the REQSESS requests a session is not available (for example, (1) a VTAM PLU application program has not opened its ACB, (2) has opened an ACB that specified MACRF=NLOGON, (3) is in the process of closing its ACB, (4) has issued SETLOGON OPTCD=QUIESCE, or (5) is unavailable because of an error condition), the REQSESS macroinstruction is rejected with, for example, (RTNCD,FDB2)=(X'10',X'02'). If the PLU application program has opened an ACB that specifies MACRF=LOGON but has not issued a SETLOGON OPTCD=START, the session-initiation request is accepted and the REQSESS macroinstruction is completed successfully. The resulting CINIT is queued at the PLU application program until it does a SETLOGON OPTCD=START, OPNDST OPTCD=ACCEPT, or CLSDST OPTCD=RELEASE.

OPTCD=SYN
OPTCD=ASY
When the SYN option code is set, control is returned to the application program when the REQSESS operation has completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. After the REQSESS operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the REQSESS operation, you should not use the SYN option if suspending the REQSESS-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

RECLEN=*user_data_length*
Indicates the number of bytes of user data (located at the AREA address) to be sent to the receiving PLU. The value in RECLEN must be 255 (decimal) or less. If the RECLEN field is set to 0, the AREA field is ignored.

Examples

```
CALLPRI  REQSESS RPL=RPLA1,NIB=NIBPA1,OPTCD=(ASY,NQ),                C
               EXIT=RQEXRTN,AREA=LOGMSG,                             C
               RECLEN=L'LOGMSG,AAREA=0
         .
         .
         .
RPLA1    RPL   ACB=ACB1,AM=VTAM
NIBPA1   NIB   NAME=GRACIE,LISTEND=YES,                               C
               LOGMODE=MODE1
LOGMSG   DC    C'LOGON REQUEST FROM USER 09'
```

CALLPRI requests a session between the application program associated with ACB1, which acts as the secondary end of the session, and application program GRACIE, which acts as the primary end of the session. When the REQSESS macroinstruction is completed, RQEXRTN is scheduled.

Completion information

A REQSESS operation is successfully completed when the SSCP responds to the Initiate request. This can be before or after BIND has been received for the requested session.

After the REQSESS macroinstruction is completed, the following RPL fields are set:

- The value 41 (decimal) is set in the REQ field, indicating a REQSESS request.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.
- If the macroinstruction returns an error code, the SSENSEI, SSENSMI, and USENSEI fields can be set indicating system-sense information, system-sense modifier, and user-sense information. More information about these fields is in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

RESETSR—Cancel RECEIVE operations and switch a session's CA-CS mode

Purpose

The RESETSR macroinstruction is used to change the continue-any or continue-specific modes of a specified session and to cancel certain RECEIVE OPTCD=SPEC macroinstructions that are outstanding for the session.

Usage

Figure 88 on page 482 summarizes the functions of RESETSR and their associated operands. For detailed information about the RUs contained in the input types, refer to "DFSYN, DFASY, and RESP types of RUs" on page 160.

Before issuing the RESETSR macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from the RESETSR macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

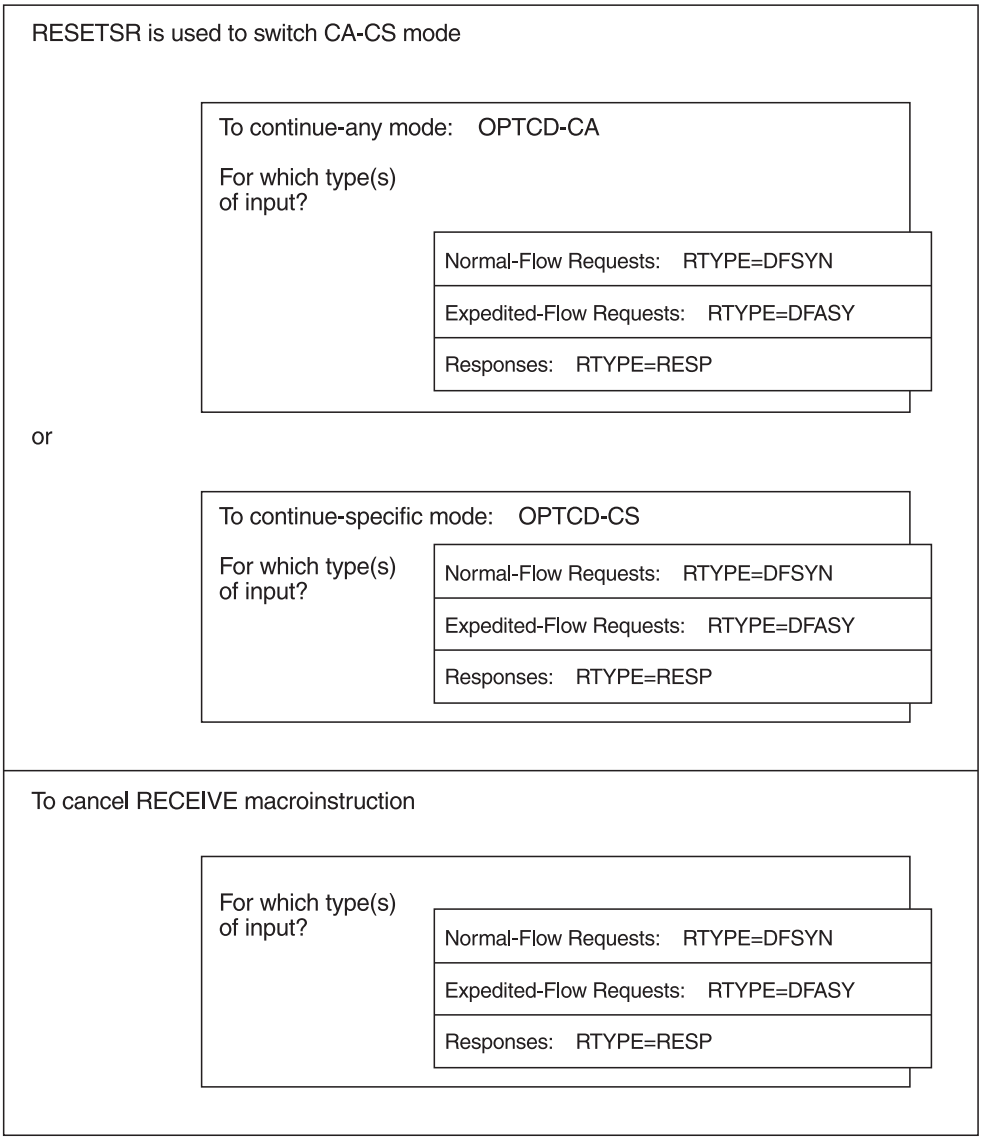


Figure 88. Major RESETSR options

Changing CA-CS mode

RESETSR changes a session's continue-any (CA) or continue-specific (CS) mode in the same manner as do SEND and RECEIVE macroinstructions.

When the CA-CS option code is set to CA, RESETSR places the session into continue-any mode if it is not already in that mode. Continue-any mode means that RECEIVE macroinstructions issued in the any-mode (OPTCD=ANY) as well as in the specific-mode (OPTCD=SPEC) can be satisfied by input from the session.

When the CA-CS option code is set to CS, RESETSR places the session into continue-specific mode if it is not already in that mode. Continue-specific mode means that only RECEIVE macroinstructions issued in the specific-mode can be satisfied by input from the session.

A session's CA-CS mode does not apply to all input from the session, but applies individually for the three types of input from the session—DFSYN, DFASY, and RESP. Refer to “Normal-flow and expedited-flow requests and responses” on page 158 for information about these types. The application program selects the type or types of input by setting the RPL's RTYPE field.

For example, suppose that RESETSR is issued with the CA-CS option set to CS (change to CS mode), and the RTYPE field set to DFASY. When the RESETSR macroinstruction is completed, the session is placed in continue-specific mode for DFASY requests. This would mean that DFASY requests sent on the session could not satisfy a RECEIVE issued in the any-mode; they could satisfy only a RECEIVE macroinstruction issued in the specific-mode.

If a RESETSR, issued to change a session's CA-CS mode, completes successfully (that is, actually changes the mode) and a pending SEND or RECEIVE for that session also specifies OPTCD=CA or OPTCD=CS. the SEND or RECEIVE might not complete successfully. The mode specified in the RESETSR can conflict with the OPTCD=CA or CS operand that is specified in the pending SEND or RECEIVE.

Note: If program performance is a critical factor, it might be more efficient to change the CA-CS mode by using the OPTCD=CA or OPTCD=CS operand on the SEND macroinstruction used for the last response or request sent.

Canceling receive requests

The RTYPE field of the RESETSR's RPL indicates, for the designated session, the type or types of RECEIVE requests that are canceled. For every RTYPE specified in the RESETSR macroinstruction, VTAM sets the corresponding RTYPE operand to its negative value (NDFSYN, NDFASY, NRESP) in each pending RECEIVE OPTCD=SPEC for the session. A RECEIVE is canceled if the combination of input types specified in its RPL is included in those specified in the RESETSR macroinstruction.

For example, suppose that these three specific RECEIVE macroinstructions are pending for a session:

```
RCV1     RECEIVE RPL=RPL1,RTYPE=(DFSYN,NDFASY,NRESP),OPTCD=SPEC
RCV2     RECEIVE RPL=RPL2,RTYPE=(DFSYN,DFASY,NRESP),OPTCD=SPEC
RCV3     RECEIVE RPL=RPL3,RTYPE=(DFSYN,DFASY,RESP),OPTCD=SPEC
```

The following RESETSR macroinstruction would change all DFSYN values to NDFSYN and all DFASY values to NDFASY:

```
RST      RESETSR RPL=RPL4,RTYPE=(DFSYN,DFASY),OPTCD=CA
```

Because the three RECEIVE macroinstructions would, in effect, now be set as follows, RCV1 and RCV2 would be canceled (all three RTYPE operands are negative), but RCV3 would not be canceled:

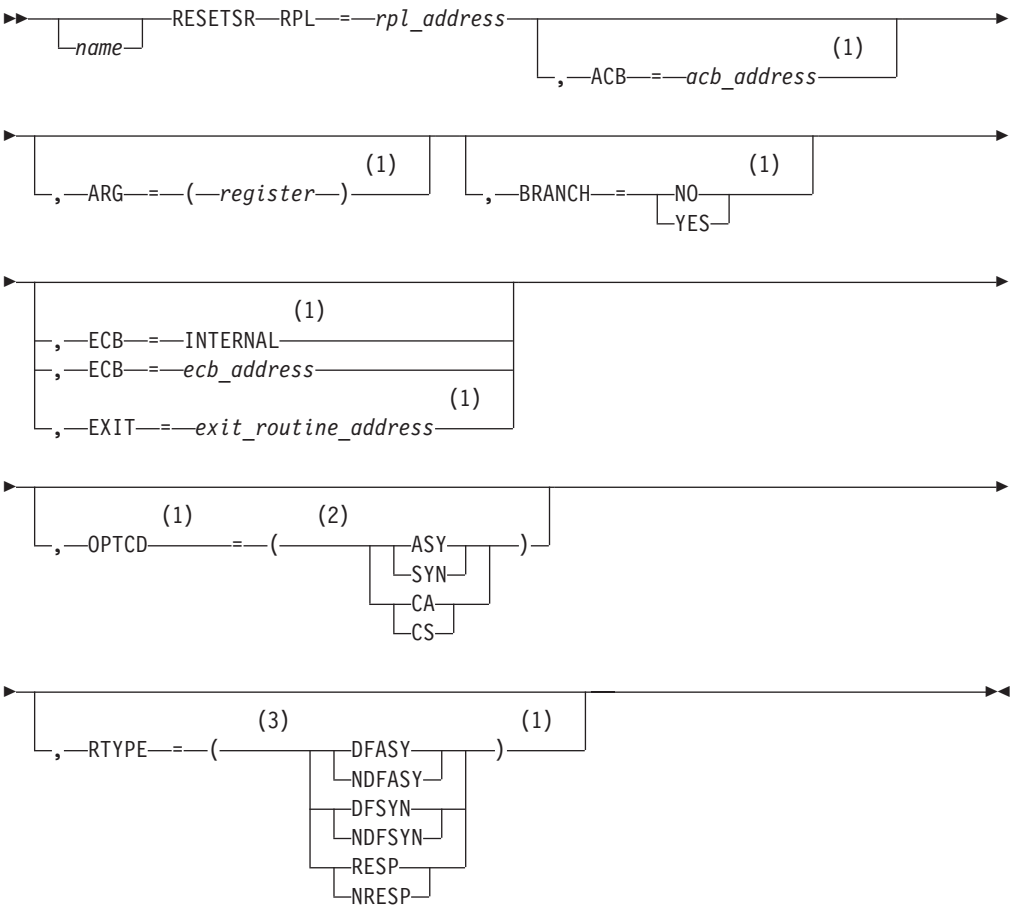
```
RCV1     RECEIVE RPL=RPL1,RTYPE=(NDFSYN,NDFASY,NRESP),OPTCD=SPEC
RCV2     RECEIVE RPL=RPL2,RTYPE=(NDFSYN,NDFASY,NRESP),OPTCD=SPEC
RCV3     RECEIVE RPL=RPL3,RTYPE=(NDFSYN,NDFASY,RESP),OPTCD=SPEC
```

When a RECEIVE is canceled, its RPL is posted complete with (RTNCD,FDB2)=(X'0C',X'0A'). The OPTCD=CA or CS setting of each canceled RPL is ignored.

Because a CA-CS mode is also specified on RESETSR, either explicitly on the RESETSR macroinstruction, or indirectly in the RPL used by the RESETSR, it can

change the CA-CS mode of the session as described in the previous section.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

3 You can code more than one suboperand on RTYPE, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing RESETSR is to perform.

The following RPL operands apply to the RESETSR macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing RESETSR.

ARG=(register)
The RESETSR macroinstruction is always directed to a specific session. The

ARG operand specifies the register containing the CID of that session. If the ARG operand is not specified, the CID in the RPLARG field is used.

Note: If your application uses the RPL DSECT, IFGRPL, you must clear the RPLNIB bit if a CID is being inserted into the RPLARG field.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB will be posted when an asynchronous (OPTCD=ASY) RESETSR operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) RESETSR operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

OPTCD=CA
OPTCD=CS
This option code determines whether the session is placed in continue-any (CA) or continue-specific (CS) mode upon successful completion of the RESETSR, (RTNCD,FDB2)=(X'00',X'00').

The new CA-CS mode applies to the type of input specified in the RTYPE field.

OPTCD=SYN
OPTCD=ASY
If the SYN option code is set, control is returned to the application program when the RESETSR operation has completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. After the RESETSR operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the RESETSR operation, you should not use the SYN option if suspending the RESETSR-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

RTYPE
The RTYPE operand indicates the type of input to be affected by the resetting of the session's continue-any or continue-specific mode and which outstanding RECEIVE macroinstructions are canceled.

RTYPE=DFASY
RTYPE=NDFASY
The session's CA-CS mode applies to expedited-flow data-flow-control requests; NDFASY means that the session's CA-CS mode for expedited-flow data-flow-control requests is not affected.

RTYPE=DFSYN
RTYPE=NDFSYN
The session's CA-CS mode applies to normal-flow requests or to DFSYN responses; NDFSYN means that the session's CA-CS mode for normal-flow requests and DFSYN responses is not affected.

RTYPE=RESP
RTYPE=NRESP
The session's CA-CS mode applies to response units (other than DFSYN responses); NRESP means that the session's CA-CS mode for responses (other than DFSYN responses) is not affected.

The RTYPE operand also designates the type of pending RECEIVEs to be canceled. A RECEIVE request is canceled, however, only if all the input types specified for the RECEIVE request's RTYPE field are also included among those specified on the RESETSR request's RTYPE field. When the RECEIVE request is canceled, its RPL is posted complete with (RTNCD,FDB2)=(X'0C',X'0A').

The following are for outstanding RECEIVE macroinstructions:

RTYPE=DFASY
RTYPE=NDFASY
Any pending RECEIVE macroinstructions that would receive expedited-flow data-flow-control requests are canceled; NDFASY means that RECEIVE requests for this type of input are not canceled.

RTYPE=DFSYN
RTYPE=NDFSYN
Any pending RECEIVE macroinstructions that would receive normal-flow requests and DFSYN responses are canceled; NDFSYN means that RECEIVE requests for this type of input are not canceled.

RTYPE=RESP
RTYPE=NRESP
Any pending RECEIVE macroinstructions that would receive responses (other than DFSYN responses) are canceled; NRESP means that RECEIVE requests for responses (other than DFSYN responses) are not canceled.

Examples

```
RST1      RESETSR RPL=RPL1,OPTCD=CA,                                    C
                RTYPE=(DFSYN,NDFASY,NRESP)
```

RST1 cancels pending RECEIVE OPTCD=SPEC,RTYPE=DFSYN macroinstructions for the session identified in RPL1's ARG field. RST1 also switches the session's CA-CS mode for DFSYN input to continue-any (CA) mode. That is, a RECEIVE OPTCD=ANY macroinstruction (RTYPE=DFSYN) can obtain normal-flow input from the session. The session's CA-CS mode for DFASY and RESP input is not affected; RECEIVE macroinstructions for these request types are also not affected.

Completion information

A RESETSR operation is successfully completed when the appropriate macroinstructions are canceled and the session's CA-CS mode is set.

After the RESETSR operation completes, the following RPL fields are set:

- The value 36 (decimal) is set in the REQ field, indicating a RESETSR request.
- The USER field contains the value that is set in the USERFLD field of the NIB when the session is established.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

RPL—Create a request parameter list

Purpose

Every request that an application program makes for session establishment or communication must refer to a request parameter list (RPL). See Figure 7 on page 20 for the list of macroinstruction categories.

Usage

The application program uses the RPL to describe most of the requests that it makes to VTAM. The application program can, for example, issue a RECEIVE and indicate an RPL. The RPL shows VTAM which session to obtain input from, where to place the input data, how to notify the application program once the operation is completed, and other options to be followed during processing of the request. If the RPL contains a request code in its REQ field, an EXECRPL macroinstruction can be used in place of the RPL-based macroinstruction indicated in REQ.

An application program can create many RPLs; a separate RPL can, in fact, be created for every RPL-based request in the application program. At the other extreme, one RPL can serve for all RPL-based requests in the program (assuming that all the requests are synchronous—that is, issued with OPTCD=SYN set). This multiple use is possible because each RPL-based request can itself modify fields of the RPL to which it points. You can think of the RPL as a list form of all RPL-based macroinstructions.

If the same RPL is used for multiple requests, it is good programming practice to reset the RPL control block fields after the request has completed. All ACB, NIB, and RPL fields that are not used by a particular macroinstruction should be set to 0 unless otherwise indicated.

The RPL macroinstruction builds an RPL during assembly. The RPL is built on a fullword boundary. Also, the GENCB macroinstruction can generate an RPL during

program execution. Requests for RPL modification can be made as part of an RPL-based request or by the MODCB macroinstruction. Either way involves naming an RPL field and specifying its new value. Also, the IFGRPL DSECT can alter RPL field values. Be aware that every operand of the RPL macroinstruction represents a field in the RPL it generates. Subsequent requests to modify any RPL field use the keyword of the operand corresponding to the field being modified.

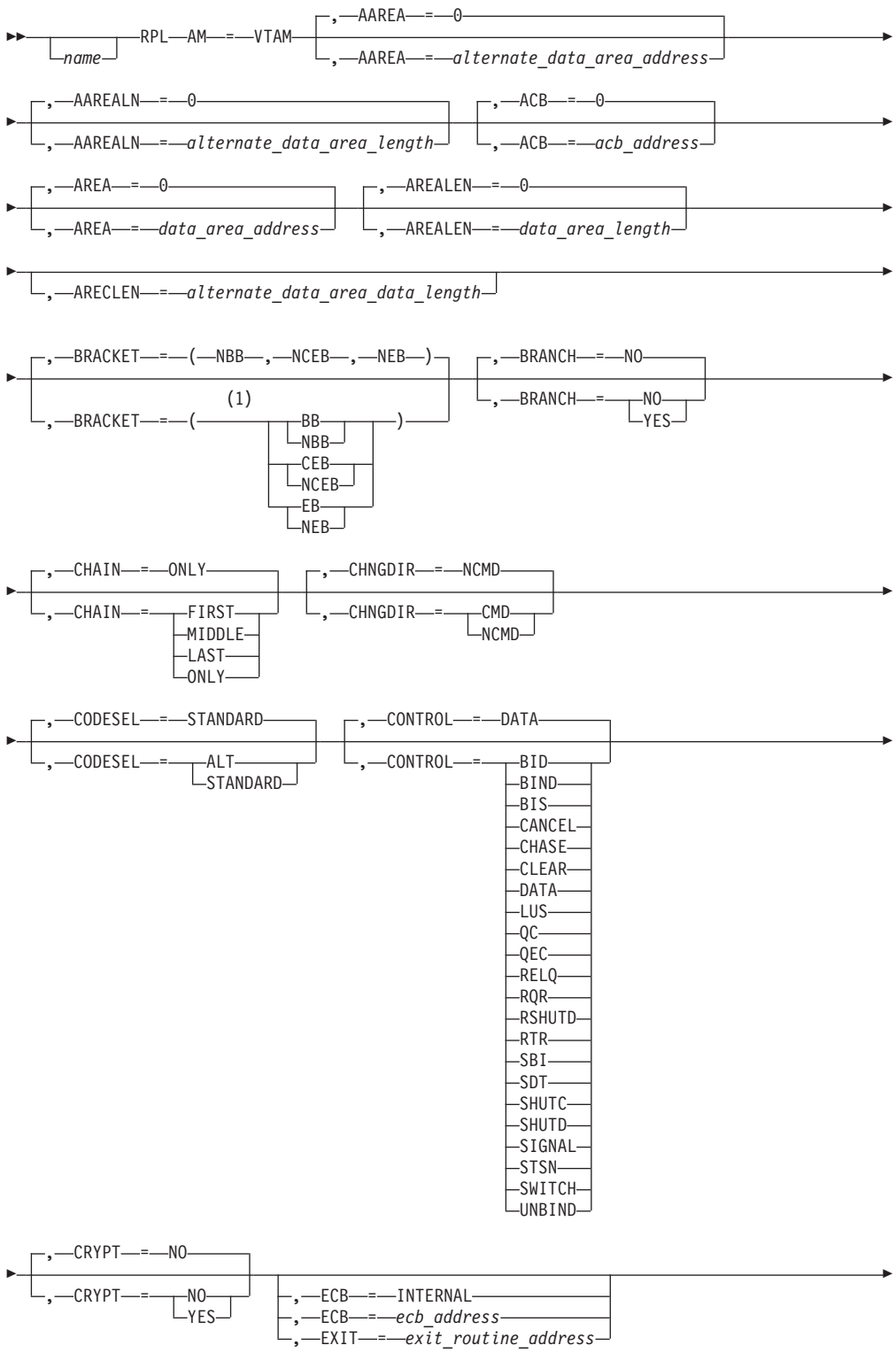
VTAM sets default values for most of the RPL fields when the RPL is initially assembled or generated. These values are underlined in the operand descriptions. After an RPL-based macroinstruction uses an RPL, VTAM might modify some of the RPL's fields. These fields are listed at the end of each macroinstruction description in this chapter and are summarized in Figure 89 on page 512.

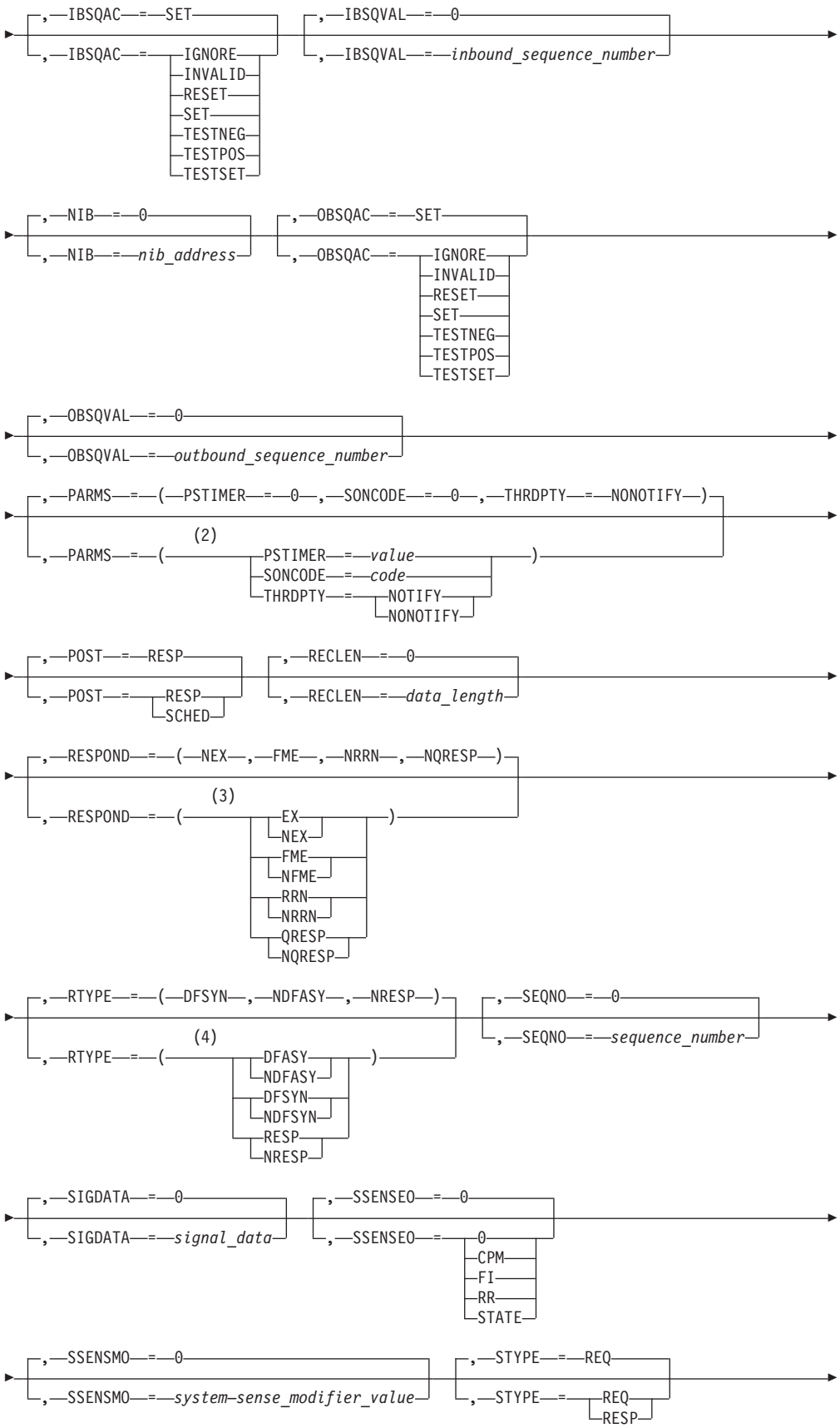
Although all of the RPL operands are optional (with the exception of AM=VTAM) and can be specified with any of the RPL-based macroinstructions, each of the RPL-based macroinstructions requires that certain RPL fields be set when the macroinstruction is executed. These fields are identified in Figure 89 on page 512 at the end of this macroinstruction description.

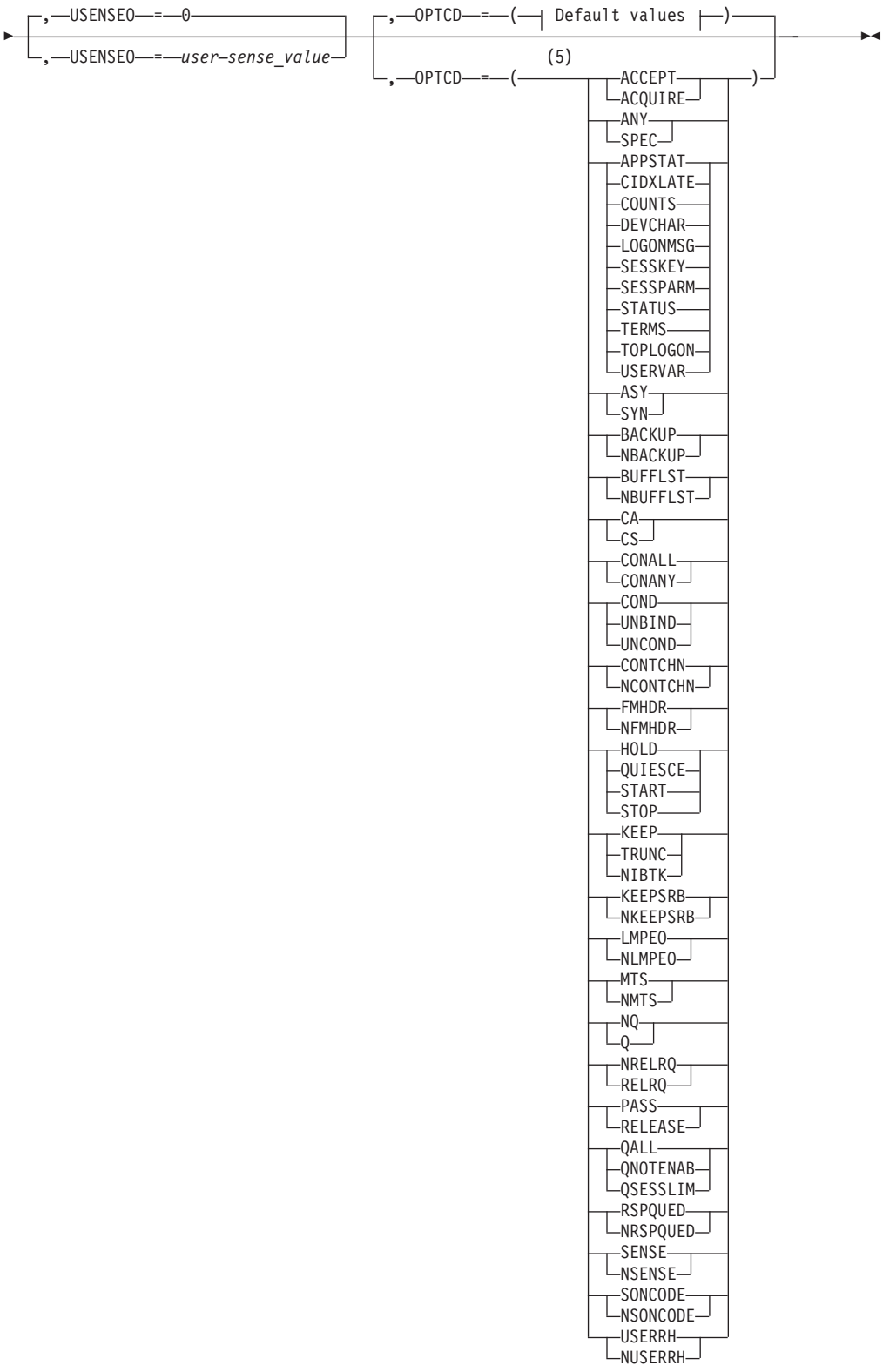
Note: For detailed descriptions about the use of the RPL operands by each RPL-based macroinstruction, see the associated macroinstruction description in this chapter.

The expansion of the RPL macroinstruction is identical for 24- and 31-bit addressing mode application programs.

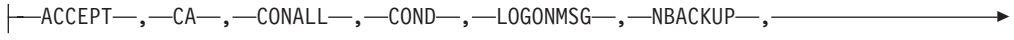
Syntax


name
RPL
AM=VTAM
,AAREA=0
,AAREA=alternate_data_area_address
,AAREALN=0
,AAREALN=alternate_data_area_length
,ACB=0
,ACB=acb_address
,AREA=0
,AREA=data_area_address
,AREALEN=0
,AREALEN=data_area_length
,ARECLEN=alternate_data_area_data_length
,BRACKET=(NBB,NCEB,NEB)
,BRACKET=(
(1)
BB
NBB
CEB
NCEB
EB
NEB
)
,BRANCH=NO
,BRANCH=
NO
YES
,CHAIN=ONLY
,CHAIN=
FIRST
MIDDLE
LAST
ONLY
,CHNGDIR=NCMD
,CHNGDIR=
CMD
NCMD
,CODESEL=STANDARD
,CODESEL=
ALT
STANDARD
,CONTROL=DATA
,CONTROL=
BID
BIND
BIS
CANCEL
CHASE
CLEAR
DATA
LUS
QC
QEC
RELQ
RQR
RSHUTD
RTR
SBI
SDT
SHUTC
SHUTD
SIGNAL
STSN
SWITCH
UNBIND
,CRYPT=NO
,CRYPT=
NO
YES
,ECB=INTERNAL
,ECB=ecb_address
,EXIT=exit_routine_address


,IBSQAC=SET
,IBSQAC=IGNORE
INVALID
RESET
SET
TESTNEG
TESTPOS
TESTSET
,IBSQVAL=0
,IBSQVAL=inbound_sequence_number
,NIB=0
,NIB=nib_address
,OBSQAC=SET
,OBSQAC=IGNORE
INVALID
RESET
SET
TESTNEG
TESTPOS
TESTSET
,OBSQVAL=0
,OBSQVAL=outbound_sequence_number
,PARMS=(PSTIMER=0,SONCODE=0,THRDPTY=NONOTIFY)
,PARMS=(
(2)
PSTIMER=value
SONCODE=code
THRDPTY=NOTIFY
NONOTIFY
)
,POST=RESP
,POST=RESP
SCHED
,RECLEN=0
,RECLEN=data_length
,RESPOND=(NEX,FME,NRRN,NQRESP)
,RESPOND=(
(3)
EX
NEX
FME
NFME
RRN
NRRN
QRESP
NQRESP
)
,RTYPE=(DFSYN,NDFASY,NRESP)
,RTYPE=(
(4)
DFASY
NDFASY
DFSYN
NDFSYN
RESP
NRESP
)
,SEQNO=0
,SEQNO=sequence_number
,SIGDATA=0
,SIGDATA=signal_data
,SSENSEO=0
,SSENSEO=0
CPM
FI
RR
STATE
,SSENSMO=0
,SSENSMO=system-sense_modifier_value
,STYPE=REQ
,STYPE=REQ
RESP



Default values for RPL's OPTCD operand:



```
►─NRSPQUED──,──NSENSE──,──NSONCODE──,──NUSERRH──,──Q──,──QALL──,──────────────┤
```

Notes:

1 You can code more than one suboperand on BRACKET, but code no more than one from each group.

2 You can code more than one suboperand on PARMS, but code no more than one from each group.

3 You can code more than one suboperand on RESPOND, but code no more than one from each group.

4 You can code more than one suboperand on RTYPE, but code no more than one from each group.

5 You can code more than one suboperand on OPTCD, but code no more than one from each group.

Input parameters

Format: Expressions involving registers cannot be used with the RPL macroinstruction.

ACB=*acb_address*
Associates the request that uses this RPL with an ACB. If you omit this operand, the ACB field is set to 0.

AM=VTAM
Indicates that a VTAM RPL is built. This operand is required.

AAREA=*alternate_data_area_address*
When used by a CLSDST OPTCD=PASS macroinstruction, AAREA indicates the location of an 8-byte area containing the symbolic name of the target PLU. When used by an INTRPRET macroinstruction, AAREA indicates a work area where VTAM places the interpreted data sequence. When used by a REQSESS macroinstruction, AAREA must be set to 0. When used by an OPNDST macroinstruction, AAREA indicates a work area where VTAM places a negotiable BIND response if the NIB specified PROC=NEGBIND.

When used by OPNDST OPTCD=RESTORE, AAREA points to the area allocated by the application to hold the recovery data that is used during session recovery.

If you omit this operand, the AAREA field is set to 0.

For XRF, the AAREA field in the RPL is initialized by the application program to provide the address of the input area where the SWITCH response information should be placed.

AAREALN=*alternate_data_area_length*
Indicates the length (in bytes) of the data area identified by the AAREA operand. When AAREA is used as an input area for an INTRPRET or OPNDST (for a negotiable BIND response) macroinstruction, VTAM uses this length to determine whether the data to be placed there is too long to fit.

When used by OPNDST OPTCD=RESTORE, the area obtained must be large enough to hold all the recovery data.

If you omit this operand, the AAREALN field is set to 0.

For XRF, the AAREALN field in the RPL is initialized by the application program to contain the length of the input area pointed to by AAREA. The AAREA field is ignored if AAREALN=0 (no switch response information is returned).

AREA=*data_area_address*
When used by a SIMLOGON, REQSESS, or a CLSDST OPTCD=PASS macroinstruction, AREA indicates the address of an area containing user data that is to be sent to the PLU (through the SSCP) in the user data field of the CINIT request.

When used by a RECEIVE macroinstruction, AREA indicates the address of the area into which data is to be read. When used by a SEND OPTCD=NBUFFLST macroinstruction, AREA indicates the address of the area from which data is to be written. When used by a SEND OPTCD=BUFFLST macroinstruction, AREA indicates the address of a buffer list which further indicates the data to be sent.

When used by an INQUIRE macroinstruction, AREA indicates where the data obtained by INQUIRE is to be placed.

When used by an INTRPRET macroinstruction, AREA indicates the address of an area containing data to be interpreted by VTAM.

When used by an RCVCMD macroinstruction, AREA contains the address of an area into which a header and a VTAM operator message is placed.

When used by a SENDCMD macroinstruction, AREA contains the address of an area containing a header and a VTAM operator command.

If you omit this operand, the AREA field is set to 0.

AREALEN=*data_area_length*
Indicates the length (in bytes) of the data area identified by the AREA operand. The AREALEN operand is meaningful only for RECEIVE, RCVCMD, and INQUIRE macroinstructions; VTAM uses this length to determine whether the data it is placing in the area is too long to fit. AREALEN=0 means that no input data area is available. If you omit this operand, the AREALEN field is set to 0.

ARECLEN=*data_length*
Indicates the length (in bytes) of the data area identified by the AAREA parameter, provided by a CLSDST OPTCD=PASS macroinstruction.

BRACKET
This operand is used when a normal-flow request is sent by SEND on a session.

BRACKET=(BB)
The begin-bracket indicator is set in the request.

BRACKET=(NBB)
The begin-bracket indicator is not set in the request.

BRACKET=(EB)
The end-bracket indicator is set in the request.

BRACKET=(NEB)
The end-bracket indicator is not set in the request.

BRACKET=(CEB)
The conditional-end-bracket indicator is set in the request.

BRACKET=(NCEB)
The conditional-end-bracket indicator is not set in the request. For details, see "Bracket protocols" on page 213.

When the session is the SSCP-LU session for a CNM application program, BRACKET=(NBB,NEB) must be specified, explicitly or by default.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

CHAIN
This field is set when a request is sent by SEND on a session. It denotes the request's relative position within the chain currently being sent.

CHAIN=FIRST
The request is first within the current chain.

CHAIN=MIDDLE
The request is in the middle of the current chain.

CHAIN=LAST
The request is last within the current chain.

CHAIN=ONLY
The request is the only request of the chain.

When the session is the SSCP-LU session for a CNM application program, CHAIN=ONLY must be specified, explicitly or by default.

CHNGDIR
This field is set when a normal-flow request is sent by SEND on a session.

CHNGDIR=CMD
A change-direction indicator is set in the request. For details, see "Half-duplex protocols" on page 211.

CHNGDIR=NCMD
When the session is the SSCP-LU session for a CNM application program, CHNGDIR=(NCMD) must be specified, explicitly or by default.

CODESEL
Indicates which data code is used in data requests being sent by SEND.

CODESEL=ALT
The alternate data code is used.

CODESEL=STANDARD
The standard data code is used.

When the session is the SSCP-LU session for a CNM application program, CODESEL=STANDARD must be specified explicitly, or by default.

CONTROL

Indicates whether data, data-flow-control, or session-control requests and responses are to be sent on a session. Data and data-flow-control requests (BID, BIS, CANCEL, CHASE, LUSTAT, QC, RTR, QEC, RELQ, RSHUTD, SBI, SHUTC, SHUTD, SIGNAL) are sent with the SEND macroinstruction. The session-control requests (CLEAR, RQR, SDT, STSN) are issued by the SESSIONC macroinstruction. Session-control responses (BIND, SDT, and STSN) are also sent using the SESSIONC macroinstruction. CONTROL=UNBIND is specified in a read-only RPL when the SCIP exit routine is scheduled with CONTROL=UNBIND. See Chapter 6, “Communicating with logical units,” on page 151, and the SEND and SESSIONC macroinstructions for an explanation of the requests designated by CONTROL.

CONTROL=DATA is the default.

When the session is the SSCP-LU session for a CNM application program, CONTROL=DATA must be specified, explicitly or by default.

CONTROL=SWITCH causes the backup XRF session to become the primary XRF session. The former primary XRF session, if it is still active, is terminated with an UNBIND(CLEANUP). This command can be issued only on a backup XRF session. If issued on a primary XRF session, it is rejected.

CRYPT

Indicates whether data is to be enciphered before it is sent by SEND on a session.

CRYPT=YES

Data is to be enciphered.

CRYPT=NO

Data is not to be enciphered.

When the session is an SSCP-LU session for a CNM application program, CRYPT=NO must be specified or defaulted.

For details about the use of this operand, see "Redbooks".

ECB

ECB=*event_control_block_address*

Indicates the location of an event control block (ECB) to be posted by VTAM when the request associated with this RPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary.

The ECB field and the EXIT field share the same RPL field. If asynchronous handling of the request has been specified (ASY option code in the RPL), the ECB-EXIT field is used in this manner:

- If you specify ECB=*address*, VTAM uses the field as the address of an external ECB; you check and clear this ECB yourself (for example, with CHECK for the RPL).
- If you specify EXIT=*address*, VTAM uses the field as the address of the RPL exit routine, and schedules the routine as indicated under the EXIT operand description.

ECB=INTERNAL

If you specify ECB=INTERNAL, VTAM uses the ECB-EXIT field as an internal ECB; you must issue CHECK for the RPL to check and clear this ECB.

If synchronous handling has been specified (SYN option code in the RPL), VTAM flags the RPL to be processed as if ECB=INTERNAL were specified. VTAM uses the ECB-EXIT field as an ECB which is cleared and checked automatically.

VTAM clears internal ECBs when:

- It begins processing any RPL-based macroinstruction
- The RPL is checked.

However, VTAM clears external ECBs only when the RPL is checked. (RPL checking is done at request completion by VTAM for synchronous request handling, and is done by the user issuing CHECK for asynchronous request handling.) Users of external ECBs must, therefore, be sure that the external ECB is cleared (with CHECK for the RPL or with assembler instructions) before the next RPL-based macroinstruction is issued.

For further information about asynchronous processing, refer to Chapter 9, "Handling errors and special conditions," on page 277.

EXIT=*rpl_exit_routine_address*
Indicates the address of a routine to be scheduled when the request represented by this RPL is completed.

If the SYN option code has been specified, the exit routine is not used; should you specify an address anyway, the address is overwritten before the synchronous request completes. (VTAM uses the ECB-EXIT field as an internal ECB in this situation—see the ECB operand discussion on page ECB.) The RPL exit routine is scheduled only if asynchronous handling of the request has been specified.

If the EXIT operand is specified, the ECB operand must not be specified. (The EXIT field and the ECB field occupy the same field in the RPL.)

For further information about asynchronous processing, refer to Chapter 9, "Handling errors and special conditions," on page 277.

IBSQAC=SET
IBSQAC=TESTSET
IBSQAC=INVALID
IBSQAC=IGNORE
IBSQAC=TESTPOS
IBSQAC=TESTNEG
IBSQAC=RESET
OBSQAC=SET
OBSQAC=TESTSET
OBSQAC=INVALID
OBSQAC=IGNORE
OBSQAC=TESTPOS
OBSQAC=TESTNEG
OBSQAC=RESET
These fields are used by a SESSIONC macroinstruction to designate which type of Set and Test Sequence Numbers (STSN) request or response is being sent on a session. The setting of the IBSQAC field relates to the inbound sequence number. The setting of the OBSQAC field relates to the outbound sequence number. SET is the default.

IBSQVAL=*inbound_sequence_number*
OBSQVAL=*outbound_sequence_number*
When SESSIONC is used to send certain STSN requests or responses, the

inbound and outbound sequence numbers can be specified by these fields. Specify any decimal integer 0 - 65535 inclusive, or any hexadecimal value that does not exceed X'FFFF'.

If either of these operands is omitted, the associated field is set to 0.

NIB=*nib_address*

Identifies the NIB whose NAME or CID field indicates the resource that is to be the object of an RPL-based macroinstruction. It is used with the CLSDST, INQUIRE, INTRPRET, OPNDST, OPNSEC, REQSESS, SESSIONC, SIMLOGON, and TERMSESS macroinstructions.

The CID and the NIB address occupy the same physical field (RPLARG) in the RPL control block. VTAM can distinguish between an NIB address and a CID only through a particular bit (RPLNIB). For this reason, this book calls the field the NIB field when an NIB address is being inserted into it and the ARG field when a CID is being inserted into it. When NIB=address appears on a SIMLOGON macroinstruction, for example, the bit is set to indicate that the field contains an NIB address. When ARG=(register) is coded on a SEND macroinstruction, for example, the bit is cleared to indicate that the field contains a CID. (Register notation must be used with ARG, because CIDs are not available until program execution.)

When dealing with the NIB and ARG operands, remember if only one physical field is involved, always use the NIB keyword to insert an NIB address and always use the ARG keyword to insert a CID. This rule also applies to the GENCB and MODCB macroinstructions.

Note: If your application uses the RPL DSECT, IFGRPL, you must set the RPLNIB bit if an NIB address is being inserted into the RPLARG field, and clear RPLNIB if a CID is being inserted into RPLARG.

If you omit this operand, the NIB field is set to 0.

OPTCD=option code or (option code, . . .)

Indicates options that are to affect the requests made using this RPL.

Code as indicated in the assembler format table. If only one option code is specified, the parentheses can be omitted.

```
RPL   ACB=ACB1,OPTCD=(SPEC,SYN,CS),AM=VTAM
RPL   ACB=ACB1,OPTCD=SPEC,AM=VTAM
```

Note: The MODCB macroinstruction can be used to change some option codes set in the RPL after it has been built.

OPTCD=ACCEPT
OPTCD=ACQUIRE

Indicates whether OPNDST is being issued to accept a session-establishment request or to acquire a session by issuing a session-initiation request.

For XRF requests, the NIB BNDAREA field must point to a BIND specifying a control vector or vectors included with the XRF-session-activation control vector (including correlation ID) initialized appropriately. OPTCD=BACKUP must be specified on the OPNDST OPTCD=ACQUIRE if the session establishment request is for a backup XRF session. The request fails if the secondary logical unit cannot support XRF. OPTCD=BACKUP or OPTCD=NBACKUP does not apply for

OPNDST OPTCD=ACCEPT and is ignored. However, if a SIMLOGON OPTCD=BACKUP is issued, it drives a LOGON exit routine that contains OPNDST OPTCD=ACCEPT.

OPTCD=CA
OPTCD=CS
The CA (continue-any) and CS (continue-specific) option codes determine which type of RECEIVE is required to obtain input from the session. This operand is used with the RTYPE operand for SEND, RECEIVE, and RESETSR. It is also used with OPNDST and OPNSEC. For further information, refer to "Continue-any mode versus continue-specific mode" on page 176.

CA places the session in a mode for a particular type of input wherein that input is subject to a RECEIVE OPTCD=ANY as well as a RECEIVE OPTCD=SPEC macroinstruction. This mode is called *continue-any mode*.

CS places the session into a mode for a particular type of input wherein only RECEIVEs for that type of input that are directed specifically to the session can be used to obtain input from it. This mode is called *continue-specific mode*.

OPTCD=CONALL
OPTCD=CONANY
Used with SIMLOGON and with OPNDST OPTCD=ACQUIRE to control the number of sessions established by these macroinstructions when an NIB is used. When CONANY is set, a session is initiated for the first available logical unit in the NIB list. Control is passed to the application program's LOGON exit routine, if one exists, when the resulting CINIT has been generated. When CONALL is set, a session is initiated for each available logical unit in the NIB list.

OPTCD=COND
OPTCD=UNCOND
OPTCD=UNBIND
Indicates the action to be taken when a TERMSESS macroinstruction is issued. If OPTCD=COND, the application program acting as the primary end of the session determines if and when to terminate the session. If OPTCD=UNCOND, VTAM terminates the session. If OPTCD=UNBIND, an UNBIND request is sent from the SLU to the PLU to terminate the session; control returns to the application program when the response to the UNBIND request is received.

OPTCD=APPSTAT
OPTCD=CIDXLATE
OPTCD=COUNTS
OPTCD=DEVCHAR
OPTCD=LOGONMSG
OPTCD=SESSKEY
OPTCD=STATUS
OPTCD=TERMS
OPTCD=TOPLOGON
OPTCD=USERVAR
Indicates the action that VTAM is to take when an INQUIRE macroinstruction is issued.

OPTCD=MTS
OPTCD=NMTS
The MTS option code only applies to the REQSESS and CLSDST OPTCD=PASS macroinstructions. If you code OPTCD=MTS, VTAM expects valid MTS

override data in an area pointed to by NIBMTSAR and formatted to match the ISTMTS DSECT. The NMTS option code tells VTAM not to expect any MTS override data.

OPTCD=NBACKUP
OPTCD=BACKUP
This parameter applies only when trying to initiate an XRF backup session. NBACKUP means that the session establishment request is for a primary XRF session or a non-XRF session. BACKUP indicates that the OPNDST request is to initiate a backup XRF session.

A backup XRF session must not be requested by the application until the OPNDST for the primary XRF session has been posted complete.

OPTCD=NBUFFLST
OPTCD=BUFFLST
Indicates, when a SEND macroinstruction is issued, whether FM data is sent by VTAM from a number of discontiguous buffers. See "The buffer-list (BUFFLST) option" on page 191 for more information.

OPTCD=NCONTCHN
OPTCD=CONTCHN
Indicates, when a SEND macroinstruction is issued with OPTCD=LMPEO, whether VTAM continues to send a chain upon receipt of a negative response.

OPTCD=NFMHDR
OPTCD=FMHDR
Indicates to VTAM how the format bit in the request header (RH) is to be set. This option applies only to SEND for data requests and data responses, and should be used to notify the logical unit that the request or response contains or does not contain (FMHDR and NFMHDR, respectively) a function management header. If FMHDR is set, the format bit is set on in the request header and is delivered to the receiver.

When the session is the SSCP-LU session for a CNM application program, FMHDR must be specified.

Note: VTAM does not prevent setting FMHDR for sending responses on LU-LU sessions; however, because this is not an SNA protocol, it should not be used. Because the format indicator is on in all data-flow-control requests and responses, the application program must ensure that the desired value is in the RPL used for SEND. (It might have been set by a previously received data-flow-control request or response.)

OPTCD=NIBTK
OPTCD=TRUNC
OPTCD=KEEP
Indicates the action to be taken when a RECEIVE macroinstruction is completed with input that is too large to fit in the input data area. TRUNC causes the excess data to be discarded. KEEP causes the excess data to be saved for subsequent RECEIVE macroinstructions. NIBTK causes the PROC=TRUNC or PROC=KEEP setting in the NIB to be used either to truncate or keep the data. RCVCMD must specify TRUNC.

OPTCD=NKEEPSRB
OPTCD=KEEPSRB
Indicates whether VTAM should return to the application under the same SRB in which VTAM was invoked. This parameter is meaningful only for synchronous SRB (*non* cross-memory) invocations. Asynchronous SRB and

synchronous cross-memory SRB invocations return under the same SRB. Preserving the SRB provides the following environmental advantages to the invoker:

- The FRR stack is maintained if KEEPFRR=YES is specified on the ACB.
- The linkage stack is maintained, thus preserving any existing linkage stack entries. Refer to the z/OS MVS Programming: Extended Addressability Guide for additional details regarding linkage stack considerations.

Note: To provide this function, VTAM utilizes SUSPEND and RESUME. Suspending the SRB (as opposed to exiting and returning under a different SRB) allows the environment to be preserved. However, SUSPEND and RESUME may impact performance. Take this into account when making use of this parameter for performance sensitive API invocations (such as SEND or RECEIVE).

OPTCD=NLMPEO
OPTCD=LMPEO
Indicates, when a SEND macroinstruction is issued, whether the large message performance enhancement outbound option is to be used. The effect of this option is to allow VTAM to split the FM data being sent into a chain of RUs. See "Large message performance enhancement outbound (LMPEO) option" on page 183 for more information.

OPTCD=NRSPQUED
OPTCD=RSPQUED
Indicates, when a SEND macroinstruction is issued, whether VTAM searches for any queued responses. When a SEND OPTCD=RSPQUED macroinstruction is posted complete, the RPL flag RPLRSPNM is set if there are any responses on the normal flow inbound response queue and the RPL flag RPLRSPQR is set if there are any responses on the normal flow inbound data queue. When the SEND is posted complete, the application program must test these RPL flags explicitly to see whether there are any queued responses.

OPTCD=NSENSE
OPTCD=SENSE
When a CLSDST macroinstruction is issued with OPTCD=RELEASE to reject a CINIT request, OPTCD=NSENSE or SENSE indicates whether sense values were specified with the SSENSEO, SSENSMO, and SSENSEO operands. If OPTCD=NSENSE is specified, VTAM rejects the CINIT with a sense value of X'08010000'. If OPTCD=SENSE is specified, VTAM rejects the CINIT with the application-specified sense values in the SSENSEO, SSENSMO, and SSENSEO fields of the RPL.

Note: Only a nonzero sense code is allowed for OPTCD=SENSE. If you specify OPTCD=SENSE, and a sense code of X'00000000', VTAM rejects the CLSDST with RTNCD/FDB2=X'14',X'50'. (This return code indicates that the RPL field is not valid).

OPTCD=NSONCODE
OPTCD=SONCODE
In the following cases, indicates whether an UNBIND type code is specified in the RPL with the PARMS=(SONCODE=*code*) operand:

- When a TERMSESS macroinstruction is issued with OPTCD=UNBIND
- When a CLSDST macroinstruction is issued with OPTCD=RELEASE.

If OPTCD=NSONCODE, VTAM uses a SON code of hex 01. If OPTCD=SONCODE, VTAM uses the SON code specified in the RPL with the PARMS=(SONCODE=*code*) operand.

OPTCD=NUSERRH
OPTCD=USERRH
Indicates, when a SEND macroinstruction is issued, whether the application program specifies the RH when sending FM data or data-flow-control requests and responses. See "The user RH (USERRH) option" on page 196 for more information.

Note: The RH field can be set up by using the RPLURH DSECT label.

OPTCD=Q
OPTCD=NQ
Indicates the action VTAM is to take when the application program issues RECEIVE, RCVCMD, SIMLOGON, REQSESS, or OPNDST OPTCD=ACCEPT and the operation cannot be completed immediately.

OPTCD=QALL
OPTCD=QSESSLIM
OPTCD=QNOTENAB
Indicates the action to be taken when a SIMLOGON OPTCD=Q is issued and the logical unit that is the object of this simulated logon request is at its session limit or not enabled for sessions. This option determines whether the SIMLOGON is queued for both or just one of these conditions.

OPTCD=QUIESCE
OPTCD=STOP
OPTCD=START
OPTCD=HOLD
Indicates the action VTAM takes when a SETLOGON macroinstruction is issued. QUIESCE causes VTAM to indicate that the application program is inhibited for any new sessions. The STOP version of SETLOGON causes any application program issuing INQUIRE OPTCD=APPSTAT to be told that new sessions should not be initiated with the application program that issued SETLOGON STOP. The START version of SETLOGON causes any application program issuing INQUIRE OPTCD=APPSTAT to be told that your application program is active. START can also be used in conjunction with SETLOGON=HOLD to pace session requests. HOLD causes CINIT and BIND requests to be queued, and the LOGON and SCIP exits not to be driven for session requests until a subsequent SETLOGON OPTCD=START is issued. SETLOGON OPTCD=STOP does not stop the scheduling of the LOGON or SCIP exit routines. STOP can be used only for private application program protocols and is not enforced by VTAM.

OPTCD=RELEASE
OPTCD=PASS
Indicates whether a session-initiation request is generated when a CLSDST macroinstruction is issued to terminate a session.

OPTCD=RELRQ
OPTCD=NRELRQ
Indicates the action to be taken when a SIMLOGON macroinstruction is issued and the logical unit that is the object of this simulated logon request has an established session or pending active session with another PLU, and is at its session limit. The effect of this option is to determine whether the application program that is in session with the logical unit is to be notified of your request.

Note the difference in spelling between the RELRQ-NRELRQ RPL option, and the related exit routine. The latter is coded in the EXLST macroinstruction as RELREQ.

OPTCD=SPEC
OPTCD=ANY

When the RPL is used by an OPNDST OPTCD=ACCEPT macroinstruction, this option code indicates whether any session or a specific session can be accepted.

When the RPL is used by a RECEIVE macroinstruction, this option code indicates whether input from any session or a specific session can complete the RECEIVE.

OPTCD=SYN
OPTCD=ASY

Indicates whether VTAM should synchronously or asynchronously handle any request made using this RPL. For further information, refer to:

- "How a synchronous operation works" on page 40
- "How an asynchronous operation works" on page 41
- "Advantages and disadvantages of different forms of operation" on page 44
- "Cautions, restrictions and techniques" on page 235
- "Synchronous versus asynchronous operations" on page 169.

SYN

Synchronous handling means that when a request is made, control is not returned to the application program until the requested operation has completed (successfully or otherwise). The application program should not use the CHECK macroinstruction for synchronous requests: VTAM automatically performs this checking (which includes clearing the internal ECB; the ECB-EXIT field in the RPL is used as an internal ECB) as discussed in the RPL ECB and EXIT operand descriptions. When control is returned to the application program, registers 0 and 15 contain completion codes. See Chapter 9, "Handling errors and special conditions," on page 277, for further information.

Some macroinstructions can take a relatively long time to complete. Because the SRB or task issuing the macroinstruction with the SYN option code is suspended until processing completes, use the ASY option code if the SRB or task cannot be allowed to be suspended for that long. Also use the ASY option when the macroinstruction is issued within an exit identified in an ACB exit list.

ASY

Asynchronous handling means that after VTAM schedules the requested operation, control is immediately passed back to the application program. When the event has completed, VTAM does one of the following:

- If an ECB address is specified for the RPL, VTAM posts a completion indicator in the event control block. The application program must issue a CHECK or a system WAIT macroinstruction to determine whether the ECB has been posted. If a system WAIT or similar technique is used, the application program must still issue a CHECK on the RPL to mark the RPL inactive so that it can be reused and to cause entry to the SYNAD or LERAD exit routine if the requested operation ends with a logic or other error.
- If the EXIT operand is in effect for the RPL, VTAM schedules the exit routine indicated by this operand. This exit routine should issue the

CHECK macroinstruction so that the RPL can be reused, and also to cause automatic entry into a LERAD or SYNAD exit routine if the requested operation ends with a logic or other error. CHECK should be issued in the exit routine even if the application program has no LERAD or SYNAD routine, because CHECK returns a code indicating whether an error occurred. See Chapter 9, "Handling errors and special conditions," on page 277, for further information.

Note: After an asynchronous request is accepted and before it is completed, do not modify the RPL used by the request. A modification during this interval could cause VTAM to be unable to complete the request in a normal manner, which in turn causes VTAM to terminate the application program.

ASY is recommended for CONTROL=SWITCH.

PARMS=(SONCODE=code)
The application program can set the UNBIND SON code by specifying OPTCD=SONCODE and PARMS=(SONCODE=code), where *code* is the 1-byte UNBIND type code to be used by VTAM on an UNBIND RU. See the description of the UNBIND RU in *SNA Formats* for definitions of those SON codes. VTAM does not validate the code specified in this parameter.

If PARMS=(SONCODE=X'FE') is specified, system and user sense codes are set with the existing SSENSEO, SSENSMO, and USENSEO RPL fields.

PARMS=(THRDPTY=NONOTIFY)
PARMS=(THRDPTY=NOTIFY)
Indicates for CLSDST OPTCD=PASS whether the application program receives notification when the new session is established between the target PLU and the SLU.

POST
This field is used with the SEND macroinstruction to determine when the SEND of a request is to be posted as being complete.

POST=RESP
Post the SEND as being complete when a response is received for the request.

POST=SCHED
Post the SEND as being complete as soon as the RPL and buffer area are available for reuse.

RECLEN=*data_length*
When used by a REQSESS, SIMLOGON, INTRPRET, or CLSDST OPTCD=PASS macroinstruction, RECLEN indicates the length (in bytes) of the user data or sequence contained in the area indicated by the AREA operand.

When used by a SEND OPTCD=NBUFFLST or SENDCMD macroinstruction, RECLEN indicates the length (in bytes) of the data that begins at the address indicated by AREA. RECLEN tells VTAM how much data to transfer. When used by a SEND OPTCD=BUFFLST macroinstruction, RECLEN indicates the length of the buffer list.

If you omit this operand, the RECLEN field is set to 0.

RESPOND
When a response is sent by SEND or SESSIONC, the RESPOND field indicates

the characteristics of the response. When a request is sent by SEND or SESSIONC, the RESPOND field indicates characteristics of the expected response to that request.

For details about the RESPOND field, refer to "What a response contains" on page 153.

When the session is the SSCP-LU session for a CNM application program, FME and NRRN are required.

RTYPE
When a RECEIVE macroinstruction is issued, the RTYPE field designates the type or types of input eligible to satisfy the macroinstruction (only one type can actually satisfy the RECEIVE). When a SEND or RESETSR macroinstruction is issued, the RTYPE field indicates the type or types of input for which the session's CA-CS mode is to be switched.

DFSYN, NDFASY, and NRESP are the defaults.

For further information about these input types, see "DFSYN, DFASY, and RESP types of RUs" on page 160.

SEQNO=*sequence_number*
Indicates the 2-byte sequence number of a response or of an expedited-flow DFC request.

If an application program responds to a request, it must set the SEQNO field for the SEND STYPE=RESP or SESSIONC STYPE=RESP with the appropriate sequence number to identify the request it is responding to.

The SEQNO field can be set to any value by the application program for the sending of an expedited data-flow-control request. VTAM uses the contents of this field for the TH sequence number of the request. Specify any decimal integer 0 - 65535 inclusive. If you omit this operand, the SEQNO field is set to 0.

For certain non-SNA 3270 logical units, the sequence number field wraps to 0 after it reaches 255. The high-order byte of the field is always set to 0. The application program should always set both bytes in the SEQNO field for all sessions, including sessions with those non-SNA 3270 logical units.

SIGDATA=*signal_data*
When the SEND macroinstruction is used to send a SIGNAL request to a logical unit, this field contains the signal data to be sent.

Specify a decimal, hexadecimal, or character constant of 1-4 bytes. If fewer than 4 bytes are specified, the value is padded to 4 bytes as if the constant were an assembler language DC statement with a length attribute of 4.

SSENSEO
This field is set by VTAM for a Logical Unit Status (LUSTAT) request and informs the logical unit of the type of error that caused the exception condition. These error types are described in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651. SSENSEO=0 is the default.

This field can also be used to provide application-specified sense values for negative responses to CINIT or for UNBIND.

SSENSMO=*system-sense_modifier_value*
This field is set by VTAM for a Logical Unit Status (LUSTAT) request. The value set in this field is used in conjunction with the SSENSEO setting to describe the specific type of error that caused the exception condition. The

meanings assigned to the SSENSMO values are described in detail in *SNA Formats* If this operand is omitted, the SSENSMO field is set to 0.

This field can also be used to provide application-specified sense values for negative responses to CINIT or for UNBIND. Refer to the TERMSESS macroinstructions in this chapter.

Specify any decimal integer 0–255 inclusive, or specify a 1-byte hexadecimal constant.

STYPE=REQ
STYPE=RESPONSE
This field designates the type of output to be sent on a session by SEND or SESSIONC. The application program uses STYPE=REQ to send a request. STYPE=RESP is used when a response is to be sent. Only REQ applies to CONTROL=SWITCH.

USENSEO=*user-sense_value*
This field is set by VTAM for a Logical Unit Status (LUSTAT) request. In most instances the user-sense field is user-defined and can be used to inform the logical unit that an exception condition is being indicated for an application-program-related error that is not an SNA-defined error, or it can be used to further modify the SNA-defined system-sense and system-sense modifier values. See Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651, for more information. If this operand is omitted, the USENSEO field is set to 0.

This field can also be used to provide application specified sense values for negative responses to CINIT or for UNBIND. Refer to the TERMSESS macroinstructions in this chapter.

Specify any decimal integer 0 - 65535 inclusive, or specify a 2-byte hexadecimal or character constant.

Examples

```
RPL1     RPL    ACB=ACB1,NIB=NIB1,AM=VTAM,                          C
                OPTCD=(SPEC,ASY),                                   C
                EXIT=EXITPGM
```

RPL1 can be used by an OPNDST macroinstruction to establish a session between the application program (represented by ACB1) issuing OPNDST and the logical unit represented in NIB1. When the operation is complete, the application program is interrupted, and the routine at EXITPGM is invoked.

```
RPL2     RPL    ACB=ACB1,AM=VTAM,AREA=SOURCE,POST=RESP,             C
                RECLEN=132,ECB=ECBWORD,OPTCD=ASY
```

RPL2 can be used by a SEND macroinstruction to write a data request (132 bytes from SOURCE) on a session with a logical unit. When the request is accepted, control is returned. When the request is completed, ECBWORD is posted. (The CHECK macroinstruction used to check the SEND operation points to RPL2.)

RPL fields set by VTAM

All of the RPL operands described in the preceding section can be supplied by the application program and cause the RPL fields to be set when the RPL macroinstruction is assembled or when the RPL-based macroinstruction is executed. Some of the fields described in the preceding section that are initially set by the application program can be (for certain macroinstructions) reset by VTAM before the macroinstruction is completed. Additional RPL fields cannot be set by

the application program but can be examined by it during program execution. VTAM uses both types of fields to return information to the application program upon completion of RPL-based macroinstruction processing. See Figure 89 on page 512 and Figure 90 on page 513 for a description of these fields.

In some cases, fields set by VTAM upon the completion of one macroinstruction cause erroneous results if the application program reuses the same RPL for another macroinstruction without again initializing the field. (Only the SSENSEI, SSENSMI, USENSEI, SSENSEO, SSENSMO, USENSEO, FDBK, FDB2, and RTNCD fields are cleared by VTAM, and no fields are reset to their original values by VTAM.)

For example, before a RECEIVE is issued, the RTYPE field is set by the application program to indicate the types of input (DFSYN, DFASY, RESP) that are eligible to satisfy the RECEIVE. The application program might indicate all three. When the RECEIVE is completed, VTAM uses the same field to indicate the type of input that actually satisfied the RECEIVE; if a RESP response was received, for instance, VTAM would reset the RTYPE field to RTYPE=(NDFSYN, NDFASY, RESP). Should the application program issue another RECEIVE with the same RPL and fail to reset the RTYPE field to its intended setting, the second RECEIVE could receive only responses.

VTAM can alter the following RPL fields prior to posting an RPL-based macroinstruction complete. These RPL fields are listed in alphabetical order. Additional information about the particular operation to which the field modifications apply is found in the following tables:

- Figure 89 on page 512 and Figure 90 on page 513
- Figure 178 on page 848, Figure 179 on page 849, and Figure 180 on page 850
- Figure 181 on page 851 and Figure 182 on page 852

See individual macroinstruction descriptions in this chapter for details. Figure 181 on page 851 and Figure 182 on page 852 show which fields are set up by VTAM in the read-only RPL supplied in various exits. Further information can be found under the exit routine descriptions given in Chapter 7, “Using exit routines,” on page 219. The read-only RPL information is similar to that which would have been placed into the RPL of a RECEIVE that had just received the RU that caused the exit routine to be scheduled.

Field name
Contents

AREA When an OPNDST or OPNSEC macroinstruction completes, AREA is set to the address of an NIB or a list of NIBs. (The NIB field is replaced by the CID). If the RPL for the OPNDST or OPNSEC is to be reused for subsequent RPL-based operations, the AREA field must be reset to indicate the data area to be used for the operation.

ARECLEN
The number of bytes of data returned by OPNDST with a NIB specifying PROC=(NEGBIND) or by INTRPRET.

ARG The communication identifier (CID) for the session is provided by the OPNDST or OPNSEC macroinstruction that establishes the session. The CID, a 32-bit value, is generated by VTAM when the session is established and is used by the application program to indicate the identity of the session for subsequent requests. It is also returned by VTAM when a RECEIVE OPTCD=ANY completes successfully. The ARG field is also used to return the CID of the session determined by INQUIRE OPTCD=TOPLOGON.

BRACKET
When a request is received, the BRACKET field indicates the status of bracket indicators.

CHAIN
When a RECEIVE RTYPE=DFSYN macroinstruction has received a request, the CHAIN field indicates the request's relative position within the chain.

CHNGDIR
When a request or response is received, the CHNGDIR field indicates the status of a change-direction indicator (CMD).

CODESEL
When a RECEIVE macroinstruction is completed, the CODESEL field indicates whether the data received is in standard or alternate code.

CONTROL
After a RECEIVE macroinstruction has completed, CONTROL is set to indicate which data or data-flow-control RU was received. The CONTROL field is also set after a SEND POST=RESP or SESSIONC STYPE=REQ completes. If the logical unit obeys SNA protocols, the CONTROL field is the same as in the original SEND or SESSIONC.

CRYPT
When a RECEIVE macroinstruction for a data request is completed, this indicates whether the RH enciphered data indicator was present for the request.

ECB/EXIT
This field has been used as an internal ECB if ECB=INTERNAL was specified, or if OPTCD=SYN was specified.

FDBK Status information for INQUIRE OPTCD=APPSTAT. This field is cleared by VTAM when the processing of the macroinstruction begins.

FDB2 A specific error return code returned by all RPL-based macroinstructions. This is one of the feedback fields described in Chapter 9, “Handling errors and special conditions,” on page 277, and in Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651. A DSECT containing labeled EQU instructions for each FDB2 return code is described in Appendix E, “Control block formats and DSECTs,” on page 739 (ISTUSFBC). These DSECT labels can be used instead of the numerical values that are cited for FDB2 throughout this book.

IBSQAC
When a SESSIONC STYPE=REQ,CONTROL=STSN is completed, the IBSQAC field contains the logical unit's response regarding the inbound sequence number for this session.

IBSQVAL
When a SESSIONC STYPE=REQ,CONTROL=STSN is completed and IBSQAC is set to TESTPOS or TESTNEG, the IBSQVAL field contains the logical unit's version of the inbound sequence number for this session.

OBSQAC
When a SESSIONC STYPE=REQ,CONTROL=STSN is completed, the OBSQAC field contains the logical unit's response regarding the outbound sequence number for this session.

OBSQVAL
When a SESSIONC STYPE=REQ,CONTROL=STSN is completed and OBSQAC is set TESTPOS or TESTNEG, the OBSQVAL field contains the

logical unit's version of the outbound sequence number for the session. When SEND is used with OPTCD=LMPEO, the OBSQVAL field is set by VTAM with the sequence number of the first RU generated.

OPTCD
When a RECEIVE macroinstruction for data is completed and a function management header is present, the OPTCD field is set to indicate FMHDR. The OPTCD field is also set when data-flow-control requests or their responses are received. Therefore, the field corresponds to the format indicator in the request header of the received request or response.

RECLEN
After an INQUIRE macroinstruction has completed, RECLEN contains the length of the requested information (such as the session parameters). After a SETLOGON OPTCD=QUIESCE macroinstruction has completed, RECLEN contains the number of CINIT requests queued for the application program. After a RCVCMD macroinstruction has completed, RECLEN contains the amount of input data. After a RECEIVE macroinstruction has completed, RECLEN contains the total length of data in VTAM's buffers (prior to the discarding of the data if TRUNC is in effect) plus the length of data already moved by that RECEIVE macroinstruction. This value might be greater than AREALEN, indicating the presence of excess data (the value in RECLEN represents the total length of excess data plus the data in AREA).

REQ A value returned by all RPL-based macroinstructions (except EXECRPL and CHECK) that identifies the type of macroinstruction. This field shows which type of macroinstruction last used the RPL. These are the values that are set:

Value Macroinstruction

21 (X'15')
SETLOGON

22 (X'16')
SIMLOGON

23 (X'17')
OPNDST

25 (X'19')
CHANGE

26 (X'1A')
INQUIRE

27 (X'1B')
INTRPRET

31 (X'1F')
CLSDST

34 (X'22')
SEND

35 (X'23')
RECEIVE

36 (X'24')
RESETSR

37 (X'25')
SESSIONC

39 (X'27')
SENDCMD

40 (X'28')
RCVCMD

41 (X'29')
REQSESS

42 (X'2A')
OPNSEC

44 (X'2C')
TERMSESS

RESPOND
When a SESSIONC STYPE=REQ, a SEND POST=RESP, or a RECEIVE macroinstruction has received a response, the RESPOND field indicates the characteristics of the response. When a RECEIVE macroinstruction has received a request, the RESPOND field indicates the characteristics of the expected response.

RPLURH
[4] When a RECEIVE macroinstruction is completed, the 3-byte RPLURH field is set with the contents of the RH used for the input operation.

RTNCD
A general return code returned by all of the RPL-based macroinstructions. This is one of the feedback fields described in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651, and in Chapter 9, "Handling errors and special conditions," on page 277.

RTYPE
When a RECEIVE macroinstruction is completed, the RTYPE field indicates the type of input that caused the completion.

SEQNO
When a RECEIVE macroinstruction has received a request or a response, the SEQNO field contains the sequence number of that RU. When a SEND or SESSIONC macroinstruction is used to send a request (STYPE=REQ), the SEQNO field contains the VTAM-supplied sequence number of the request. When using SEND OPTCD=LMPEO, SEQNO is set by VTAM with the sequence number of the last RU sent.

SIGDATA
When a Signal request is received, the SIGDATA field contains the signal information sent on the session.

SSENSEI
When a SEND POST=RESP, RECEIVE, or SESSIONC STYPE=REQ macroinstruction receives a negative response, or when a RECEIVE receives an exception request or an LUSTAT data-flow-control request, the SSENSEI, SSENSMI, and USENSEI fields are set with sense information. Similarly, under some circumstances, the INQUIRE, INTRPRET, CLSDST, OPNDST, OPNSEC, REQSESS, SIMLOGON, SETLOGON, and TERMSESS macroinstructions can be posted complete with sense information. For

4. This is a label in the ISTRH DSECT (shown in Appendix E, "Control block formats and DSECTs," on page 739), rather than a field name. No RPL operand exists for this field.

further details about these fields, refer to Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651. VTAM clears these fields when the processing of the macroinstruction begins.

SSENSEO
This field is always set to 0 when an RPL-based macroinstruction is completed.

SSENSMI
When a SEND POST=RESP, RECEIVE, or SESSIONC STYPE=REQ macroinstruction receives a negative response, or when a RECEIVE receives an exception request or an LUSTAT data-flow-control request, the SSENSEI, SSENSMI, and USENSEI fields are set with sense information. Similarly, under some circumstances, the INQUIRE, INTRPRET, CLSDST, OPNDST, OPNSEC, REQSESS, SIMLOGON, SETLOGON, and TERMSESS macroinstructions can be posted complete with sense information. For further details about these fields, refer to Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651. VTAM clears these fields when the processing of the macroinstruction begins.

SSENSMO
This field is always set to 0 when an RPL-based macroinstruction is completed.

USENSEI
When a SEND POST=RESP, RECEIVE, or SESSIONC STYPE=REQ macroinstruction receives a negative response, or when a RECEIVE receives an exception request or an LUSTAT data-flow-control request, the SSENSEI, SSENSMI, and USENSEI fields are set with sense information. Similarly, under some circumstances, the INQUIRE, INTRPRET, CLSDST, OPNDST, OPNSEC, REQSESS, SIMLOGON, SETLOGON, and TERMSESS macroinstructions can be posted complete with sense information. For further details about these fields, refer to Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651. These fields are cleared by VTAM when the processing of the macroinstruction begins.

USENSEO
This field is always set to 0 when an RPL-based macroinstruction is completed.

USER Upon the completion of a SEND, RECEIVE, RESETSR, or SESSIONC macroinstruction, this field contains whatever value you previously placed in the USERFLD field of the NIB used to establish the session. See the description of the USERFLD operand of the NIB macroinstruction for more information.

RPL fields and RPL-based macroinstructions

Figure 89 on page 512 and Figure 90 on page 513 show all of the VTAM macroinstructions that allow RPL modifications to be indicated when the macroinstruction is coded. It also shows all of the RPL fields, including the option codes of the OPTCD field, that might be modified by the application program or by VTAM. The symbols in the table indicate, for a given macroinstruction, the RPL fields that are set by VTAM or by the application program. The programmer coding the macroinstruction should be aware of each field's effect by checking the description of that macroinstruction in this chapter. The absence of an A or V means that the contents of that field are ignored by VTAM when that macroinstruction is issued.

Note: When issuing an RPL-based macroinstruction, the value remaining in an RPL field after its last use is reused unless the application program explicitly alters the field. Default values for the operands of RPL-based macroinstructions do not exist. The programmer must ensure that the current contents of each RPL field applicable to this macroinstruction are properly set.

Figure 178 on page 848, Figure 179 on page 849, and Figure 180 on page 850 provide detailed information about the SEND, SESSIONC, and RECEIVE macroinstructions for various inputs, outputs, and modes of operation. This figure lists information set in the read-only RPL supplied in exit routines. For further information about exit routines, refer to Chapter 7, "Using exit routines," on page 219.

RPL - Modifying Macro-instructions → / Applicable RPL Fields:			CLSDST				OPNDST														
	CHANGE	(PASS)	(RELEASE)	EXECRPL	INQUIRE	INTRPRET	(ACQUIRE)	(ACCEPT)	(RESTORE)	OPNSEC	RCVCMD	RECEIVE	REQSESS	RESETSR	SEND	SENDCMD	SESSIONC	SETLOGON	SIMLOGON	TERMSESS
ACB	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
ARG/NIB (when ARG specified)		A	A	AV	AV	A	V[3]	V[3]	V[3]	V[3]		AV		AV	AV		AV			A
ARG/NIB (when NIB specified)	A	A	A	A	A	A	A[3]	A[3]	A[3]	A[3]			A				A		A	A
AREA		A		AV	A	A	V	V		V	A	A	A		A	A			A	
AREALEN				A	A						A	A								
RECLEN		A		AV	V	A					V	V	A		A	A		V	A	
AAREA		A		A		A	A	A	A				A				A			
AAREALN				AV		A	A	A	A		V						A			
ARECLEN				V		V	V	V	V											
BRANCH	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
EXIT/ECB (when ECB specified)	For all macros: A																			
EXIT/ECB (when EXIT specified)	For all macros: If OPTCD=ASY,A; OPTCD=SYN,AV																			
EXIT/ECB (when internal ECB is used)	For all macros: V																			
REQ	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
RTNCD[1]	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
FDB2[1]	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
FDBK[1]				V	V															
USER				V								V		V	V		V			
SEQNO				AV								V			AV		AV			
POST				A											A					
RESPOND				AV								V			AV		AV			
CONTROL				AV								V			AV		AV			
CHAIN				AV								V			A					
CHNGDIR				AV								V			A		V			
BRACKET				AV								V			A					
RTYPE				AV								AV		A	A					
STYPE				A											A		A			
SSENSEO[2]			A	A											A		A			A
SSENSMO[2]			A	A											A		A			A
USENSEO[2]		V		A											A		A			A
SSENSEI[1]		V		V	V	V	V	V		V		V	V		V		V	V	V	V
SSENSMI[1]		V		V	V	V	V	V		V		V	V		V		V	V	V	V
USENSEI[1]				V	V	V	V	V		V		V	V		V		V	V	V	V
CRYPT				AV								V			A					
RPLURH[4]				AV								V			A					
IBSQAC				AV													AV			
OBSQAC				AV													AV			
IBSQVAL				AV													AV			
OBSQVAL				AV													AV			
SIGDATA				AV								V			AV[5]					
CODESEL				AV								V			A					

Figure 89. RPL fields applicable to the macroinstructions that can modify RPLs (Part 1 of 2)

OPTCD: Applicable RPL Fields:	CHANGE	(PASS)	(RELEASE) CLSDST	EXECRPL	INQUIRE	INTRPRET	(ACQUIRE) OPNDST	(ACCEPT)	(RESTORE)	OPNSEC	RCVCMD	RECEIVE	REQSESS	RESETSR	SEND	SENDCMD	SESSIONC	SETLOGON	SIMLOGON	TERMSESS
ENDAFFIN-ENDAFFNF	A			A																
TRUNC-KEEP-NIBTK				A							A	A								
FMHDR-NFMHDR				AV								V			A					
CONANY-CONALL				A			A												A	
ACQUIRE-ACCEPT-RESTORE				A			A	A	A											
SPEC-ANY				A				A				A								
QUIESCE-START-STOP-HOLD-PERSIST-NPERSIST-GNAMEADD-GNAMEDEL				A														A		
PASS-RELEASE-TERMQ		A	A	A																
LOGONMSG-DEVCHAR-COUNTS-TERMS-APPSTAT-CIDXLATE-TOPLOGON-SESSPARM-SESSKEY-PERSESS-USERVAR-SESSNAME-NQN-STATUS				A	A															
SYN-ASY	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
CS-CA				A			A	A		A		A		A	A					
COND-UNCOND-UNBIND-TERMQ				A															A	A
RELRQ-NRELRQ				A															A	
Q-NQ				A				A			A	A	A							
LMPED-NLMPEO				A											A					
CONTCHN-NCONTCHN				A											A					
BUFFLST-NBUFFLST				A											A					
USERRH-NUSERRH				A											A					
SENSE-NSENSE			A	A																
SONCODE-NSONCODE			A	A																A
RSPQUED-NRSPQUED				A											A					
QALL-QSESSLIM-QNOTENAB				A															A	
MTS-NMTS		A		A									A							
BACKUP-NBACKUP				A			A	A											A	
KEEPSRB-NKEEPSRB	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
PARMS																				
THRDPTY=NOTIFY		A																		
SONCODE=code			A																	A
PSTIMER=value																		A		
FORCETKO																		A		

RPL - Modifying Macro-instructions

[1] VTAM sets this field to 0 when the macroinstruction is accepted; a non zero value may be set by VTAM when the macroinstruction is posted complete.

[2] VTAM sets this field to 0 when the macroinstruction is posted complete.

[3] The NIB-ARG is specified as a NIB field by the application program, but VTAM may change it into an ARG field before returning control to the application program.

[4] RPLURH is a label in the ISTRH DSECT (Appendix E), rather than a field name. There is no operand for this field.

[5] This field is set by VTAM when SEND OPT=LIMPEO is posted complete.

The presence of a symbol means that the RPL field or option code is set by the macroinstruction in one of three ways;

A The field or option code is set by the application program to supply VTAM information about the request.

V The field is set by VTAM when the request has been completely processed.

AV The field is set by the application program and then reset by VTAM. Users intending to reissue requests that use these fields should reinitialize them first.

Figure 90. RPL fields applicable to the macroinstructions that can modify RPLs (Part 2 of 2)

SEND—Send output on a session

Purpose

The SEND macroinstruction transmits a request or a response for data or data-flow-control on a session with a logical unit. SEND can also be used by a CNM application program to send a request or a response to the SSCP on the SSCP-LU session. The major options for a SEND macroinstruction are illustrated in Figure 91 on page 517.

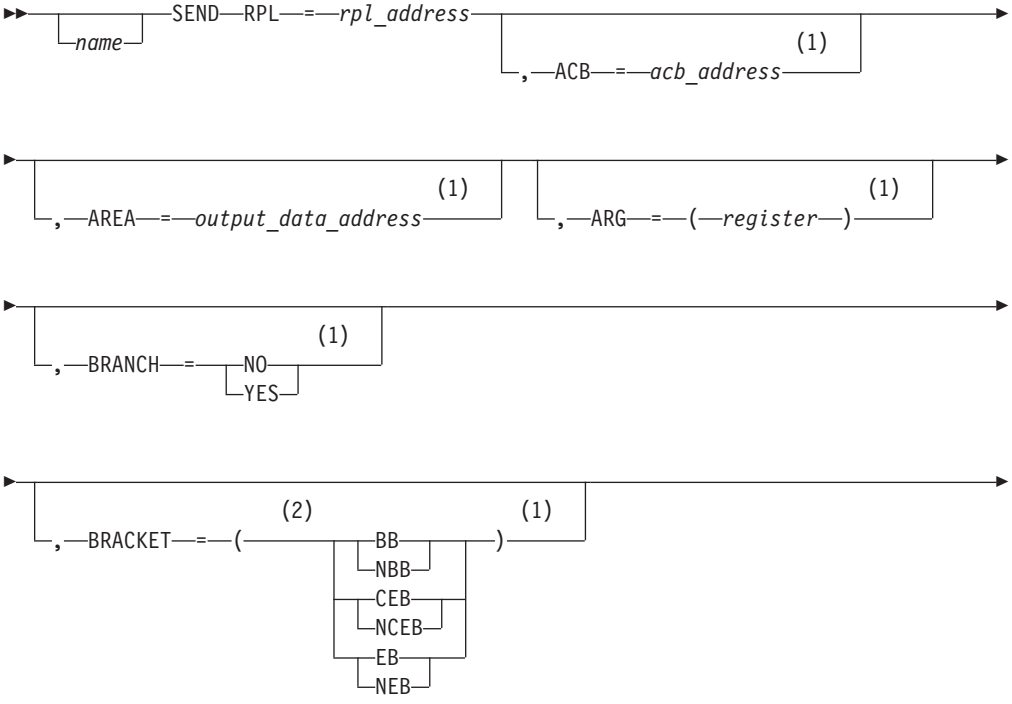
Usage

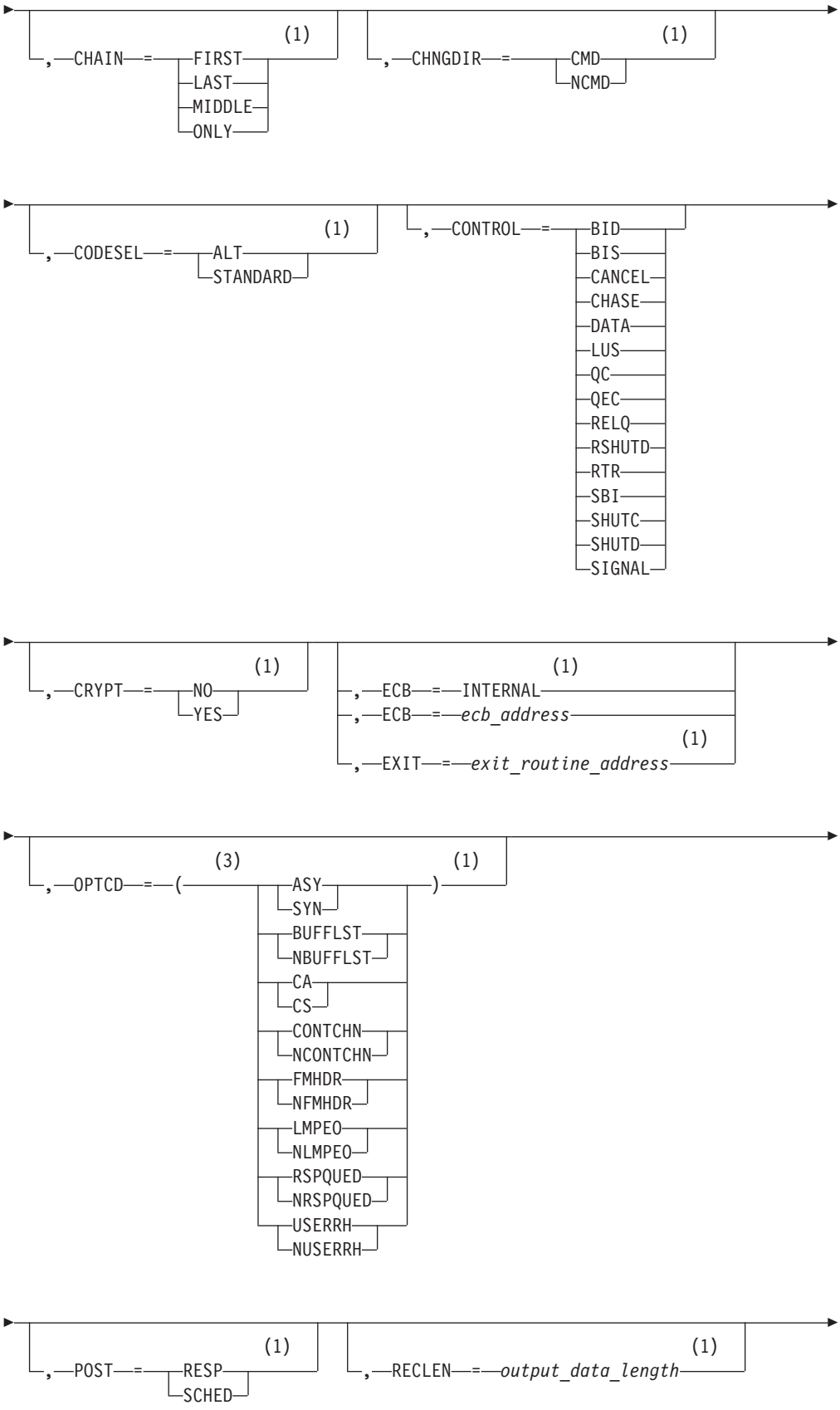
Before issuing the SEND macroinstruction the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

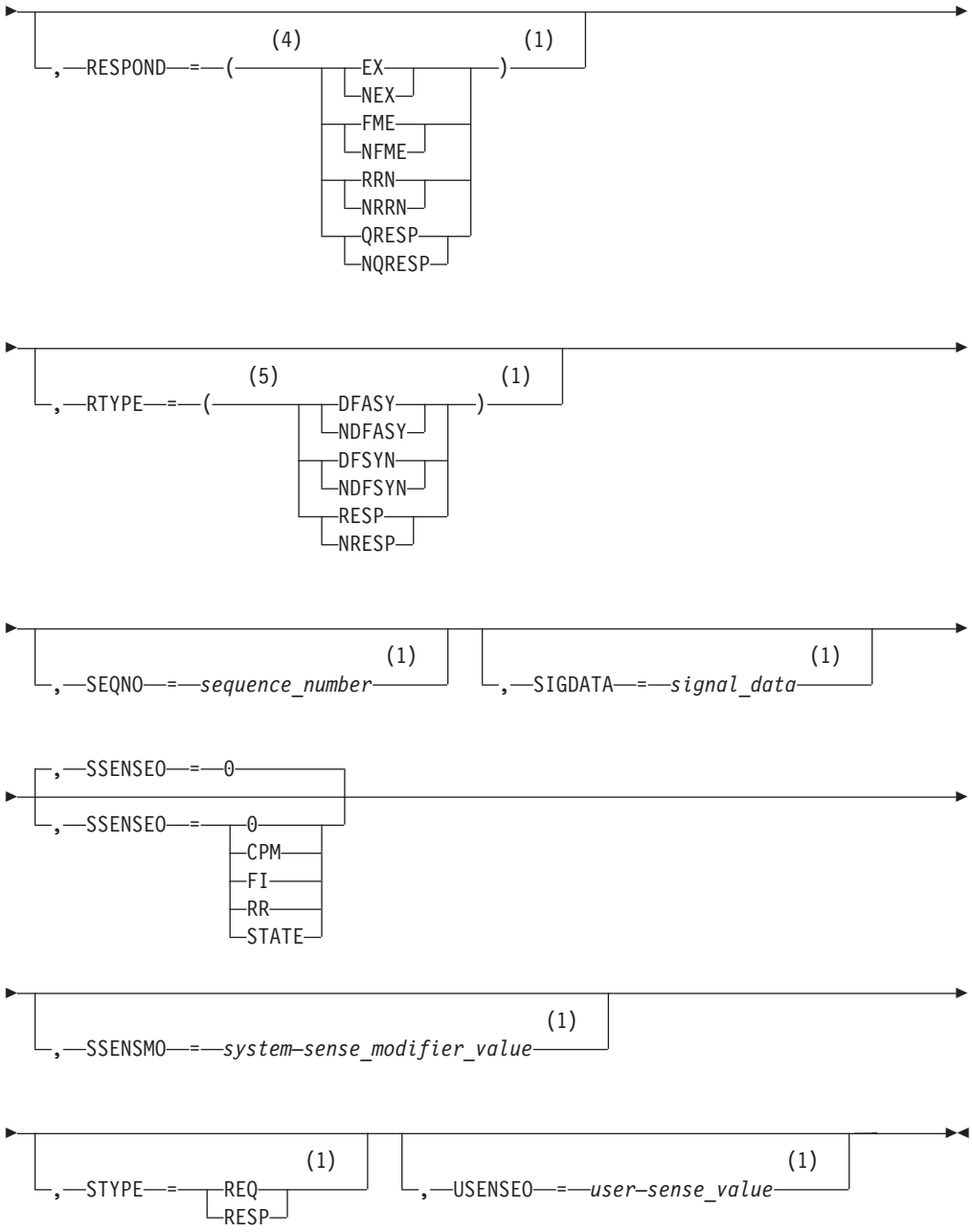
For further information about the SEND macroinstruction, refer to "Using VTAM to communicate with LUs" on page 168.

VTAM receives control from the SEND macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

Syntax




,CHAIN=(1) FIRST LAST MIDDLE ONLY
,CHNGDIR=(1) CMD NCMD
,CODESEL=(1) ALT STANDARD
,CONTROL= BID BIS CANCEL CHASE DATA LUS QC QEC RELQ RSHUTD RTR SBI SHUTC SHUTD SIGNAL
,CRYPT=(1) NO YES
,ECB=(1) INTERNAL
,ECB=ecb_address
,EXIT=(1) exit_routine_address
,OPTCD=(3) ASY SYN BUFFLST NBUFFLST CA CS CONTCHN NCONTCHN FMHDR NFMHDR LMPEO NLMPEO RSPQUED NRSPQUED USERRH NUSERRH (1)
,POST=(1) RESP SCHED
,RECLEN=(1) output_data_length



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on BRACKET, but code no more than one from each group.

3 You can code more than one suboperand on OPTCD, but code no more than one from each group.

4 You can code more than one suboperand on RESPOND, but code no more than one from each group.

5 You can code more than one suboperand on RTYPE, but code no more than

one from each group.

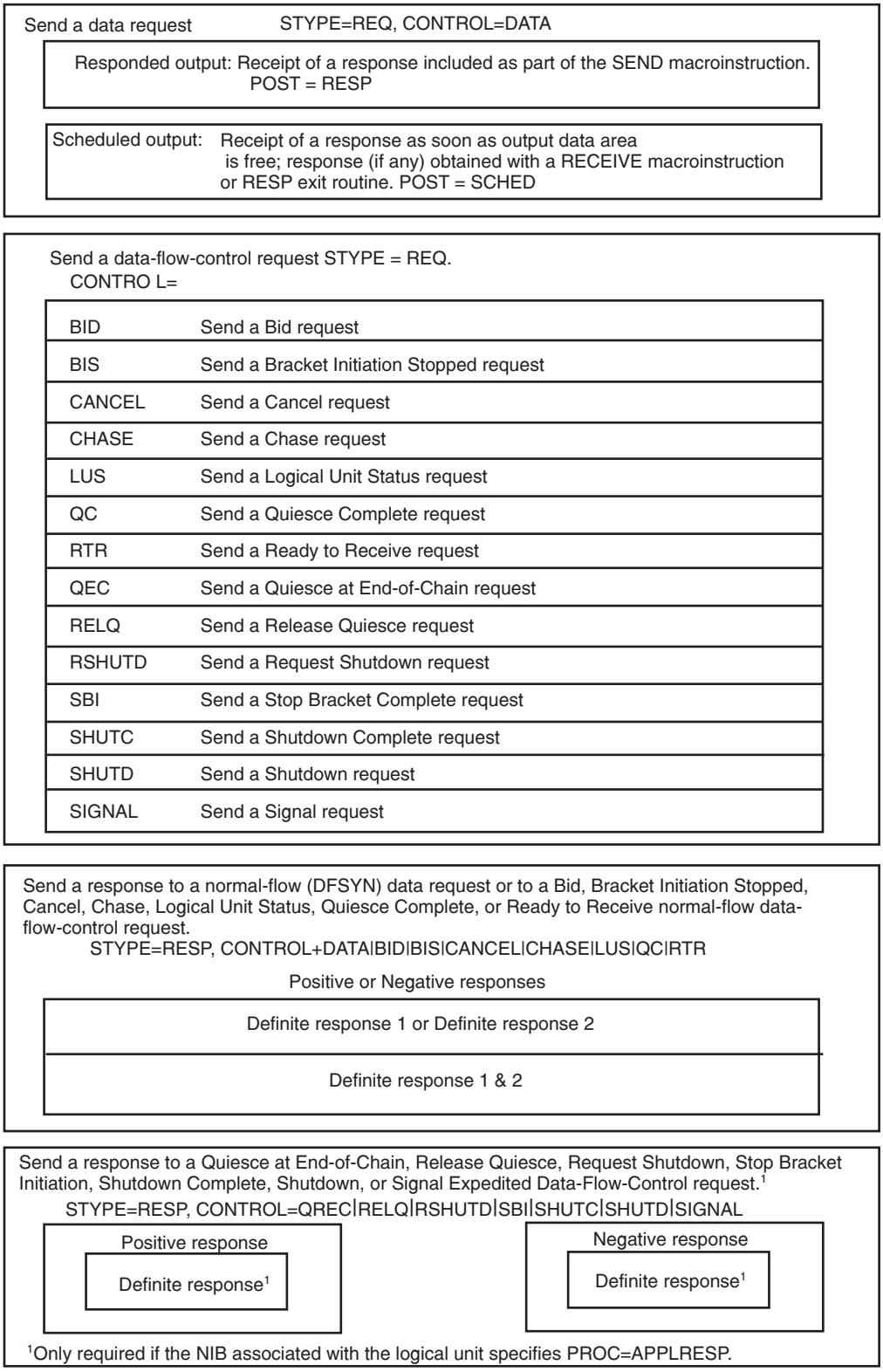


Figure 91. Major SEND options

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing SEND is to perform.

The following RPL operands apply to the SEND macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing SEND.

AREA=*output_data_address*
If OPTCD=NBUFFLST, CONTROL=DATA, and STYPE=REQ, AREA specifies the address of the data to be sent. If OPTCD=BUFFLST, CONTROL=DATA, and STYPE=REQ, AREA specifies the address of a buffer list which in turn specifies the data to be sent. This storage can be reused as soon as VTAM has transferred the data to its own buffers (see the following POST=SCHED discussion).

ARG=(register)
The SEND macroinstruction is always directed to one session. The ARG operand specifies the register containing the CID of the session being transmitted to. If the ARG operand is not specified, the CID that is already in the RPLARG field is used.

Note: If your application uses the RPL DSECT, IFGRPL, you must clear the RPLNIB bit if a CID is being inserted into the RPLARG field. For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH
If the macroinstruction is issued in an application program that is running in supervisor state under a TCB, set BRANCH to YES to specify that you should use authorized path processing.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

BRACKET
This field indicates whether a chain is the beginning, middle, end, or the only chain in a bracket. This operand is meaningful only when bracket protocols are being used on the session. Refer to "Bracket protocols" on page 213.

- BRACKET=(BB,NEB) This chain is the beginning of a bracket.
- BRACKET=(NBB,NEB) This chain is the middle of a bracket.
- BRACKET=(NBB,EB) This chain is the end of a bracket.
- BRACKET=(BB,EB) This chain is the only chain of a bracket.

For all requests in a chain except the first and last requests, (NBB,NEB) is set.

BRACKET=CEB is used to set the conditional end bracket indicator. This can be used only for the last or for the only request in a chain.

CHAIN
Indicates the position of the request within the chain of requests currently being transmitted. For additional information, see “Chaining” on page 201.

CHAIN=FIRST
The request is first within the current chain.

CHAIN=MIDDLE
The request is in the middle of the current chain.

CHAIN=LAST
The request is last within the current chain.

CHAIN=ONLY
The request is the only request of the chain. Also, see “Large message performance enhancement outbound (LMPEO) option” on page 183 to understand how VTAM splits a message sent with SEND OPTCD=LMPEO into chained request units.

CHNGDIR
Indicates whether the change-direction indicator is set on in the request sent. Refer to “Half-duplex protocols” on page 211 for an explanation of the change-direction indicator.

CHNGDIR=(CMD)
Turn on the change-direction indicator in the request to be sent (valid only for a last-of-chain DFSYN request). This corresponds to the SNA-defined change-direction indicator.

CHNGDIR=(NCMD)
Turn off the change-direction indicator in the request to be sent. This is the setting for all requests of the chain except the last of chain.

CODESEL
Indicates whether the request being sent on the session is encoded in the standard code (STANDARD) or in some other code (ALT). The application program and the logical unit must have previously agreed what type of code is recognized as the standard code (such as EBCDIC) and what code is recognized as the alternate code (such as ASCII). This operand is meaningful only for CONTROL=DATA,STYPE=REQ.

CODESEL=STANDARD
Indicates that the request being sent on the session is encoded in the standard code (STANDARD).

CODESEL=ALT
Indicates that the request being sent on the session is encoded in some other code (ALT).

CRYPT
This operand applies only if CONTROL=DATA and STYPE=REQ.

CRYPT=YES
Indicates that the data at the location specified by AREA is to be enciphered before it is sent on the session.

CRYPT=NO
Indicates that the data at the location specified by AREA is not to be enciphered before it is sent on the session.

See "Redbooks" for details and a warning about the use of this operand.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) SEND operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) SEND operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

OPTCD=BUFFLST
OPTCD=NBUFFLST
Specifies whether the buffer list option is used. OPTCD=BUFFLST allows FM data to be sent from a number of discontiguous buffers. The RPL AREA field points to a buffer list which is a contiguous set of 16-byte control blocks, called buffer list entries. The RPL RECLEN field must specify a buffer list length that is a nonzero multiple of 16 bytes. For a detailed description of the use of the BUFFLST option, refer to "The buffer-list (BUFFLST) option" on page 191.

OPTCD=CA
OPTCD=CS
When the SEND operation is completed successfully, (RTNCD,FDB2)=(X'00',X'00') or (X'04',X'04'), the session is placed in continue-any mode (OPTCD=CA) or continue-specific mode (OPTCD=CS) for the types of input specified by the RTYPE field. More than one type of input can be specified. VTAM switches the modes for all specified types of input. No switching occurs if RTYPE=(NDFSYN,NDFASY,NRESP) is in effect for the SEND. The SEND is posted complete when the last request unit associated with the SEND is handled by VTAM; for OPTCD=LMPEO and OPTCD=BUFFLST, more than one request unit can be sent by SEND.

OPTCD=CONTCHN
OPTCD=NCONTCHN
Specifies the action VTAM should take on receipt of a negative response to a chain being sent with OPTCD=LMPEO. VTAM either continues to send the chain (OPTCD=CONTCHN) or aborts the chain, (OPTCD=NCONTCHN) possibly by sending a zero-length LIC request unit. (Refer to "Exception conditions" on page 190 for details.)

OPTCD=FMHDR
OPTCD=NFMHDR
When OPTCD=FMHDR is used, the receiver is notified that the data contains a function management header.

OPTCD=LMPEO
OPTCD=NLMPEO
Specifies whether the large message performance enhancement outbound option is to be used. OPTCD=LMPEO allows VTAM to reformat FM data into one or more request units to form a chain or partial chain of RUs. (A detailed description of LMPEO is contained in "Large message performance

enhancement outbound (LMPEO) option" on page 183.) When OPTCD=(LMPEO,BUFFLST) is specified, the FM data to be sent is pointed to indirectly by the RPL AREA field which points to a buffer list. If OPTCD=(LMPEO,NBUFFLST) is specified, the RPL AREA field points directly to the FM data to be sent. (See "The buffer-list (BUFFLST) option" on page 191 for a detailed description of buffer list operation.) POST=SCHED or POST=RESP can be used with OPTCD=LMPEO. If a negative response is received before a SEND is posted complete, the POST=SCHED or POST=RESP and OPTCD=CONTCHN or OPTCD=NCONTCHN operands determine what action is to be taken. For a detailed description, refer to "Exception conditions" on page 190.

OPTCD=RSPQUED
OPTCD=NRSPQUED
Specifies whether VTAM should search for any queued responses. When the SEND is posted complete, if OPTCD=RSPQUED is specified, the RPL flag RPLRSPNM is set if there are any responses on the normal-flow inbound response queue. The RPL flag RPLRSPQR is set if there are any responses on the normal-flow inbound data queue.

The application program can test these RPL flags when the SEND is posted complete to see if there are any queued responses.

OPTCD=SYN
OPTCD=ASY
If the SYN option code is set, control is returned to the application program when the SEND operation has completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. After the SEND operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. The SEND is posted complete when the last request unit associated with the SEND is handled by VTAM; for OPTCD=LMPEO and OPTCD=BUFFLST, more than one request unit can be sent by SEND. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

It might take VTAM a relatively long time to complete the SEND operation. Because the SEND-issuing task or SRB issuing the macroinstruction with the SYN option code is suspended until processing completes, use the ASY option code if the SEND-issuing task or SRB cannot be allowed to be suspended for that long.

OPTCD=USERRH
OPTCD=NUSERRH
Specifies whether the user RH option is used. If OPTCD=USERRH is specified, VTAM includes a 3-byte user RH header from the RPL (RPLURH) or from a buffer list entry for each RU sent. FM data and data-flow-control requests and responses can be sent using OPTCD=USERRH. The RPL CONTROL operand must be set to the type of request or response unit being sent. The RUCAT field in the user RH must be set to FMD or DFC. If OPTCD=(USERRH,BUFFLST), the initial RH is specified in the RH field of certain buffer list entries. For a detailed description of the user RH option, refer to "The user RH (USERRH) option" on page 196.

Note: The RH field in the RPL can be set up using the RPLURH DSECT label.

POST
Defines at what point in the output operation the SEND macroinstruction is to be completed. The OPTCD=SYN or OPTCD=ASY, ECB, and EXIT operands govern the action to be taken when the macroinstruction completes. Figure 92 on page 522

on page 522 summarizes the use of the POST operand, including defaults assumed. See also "Scheduled versus responded output operations" on page 172.

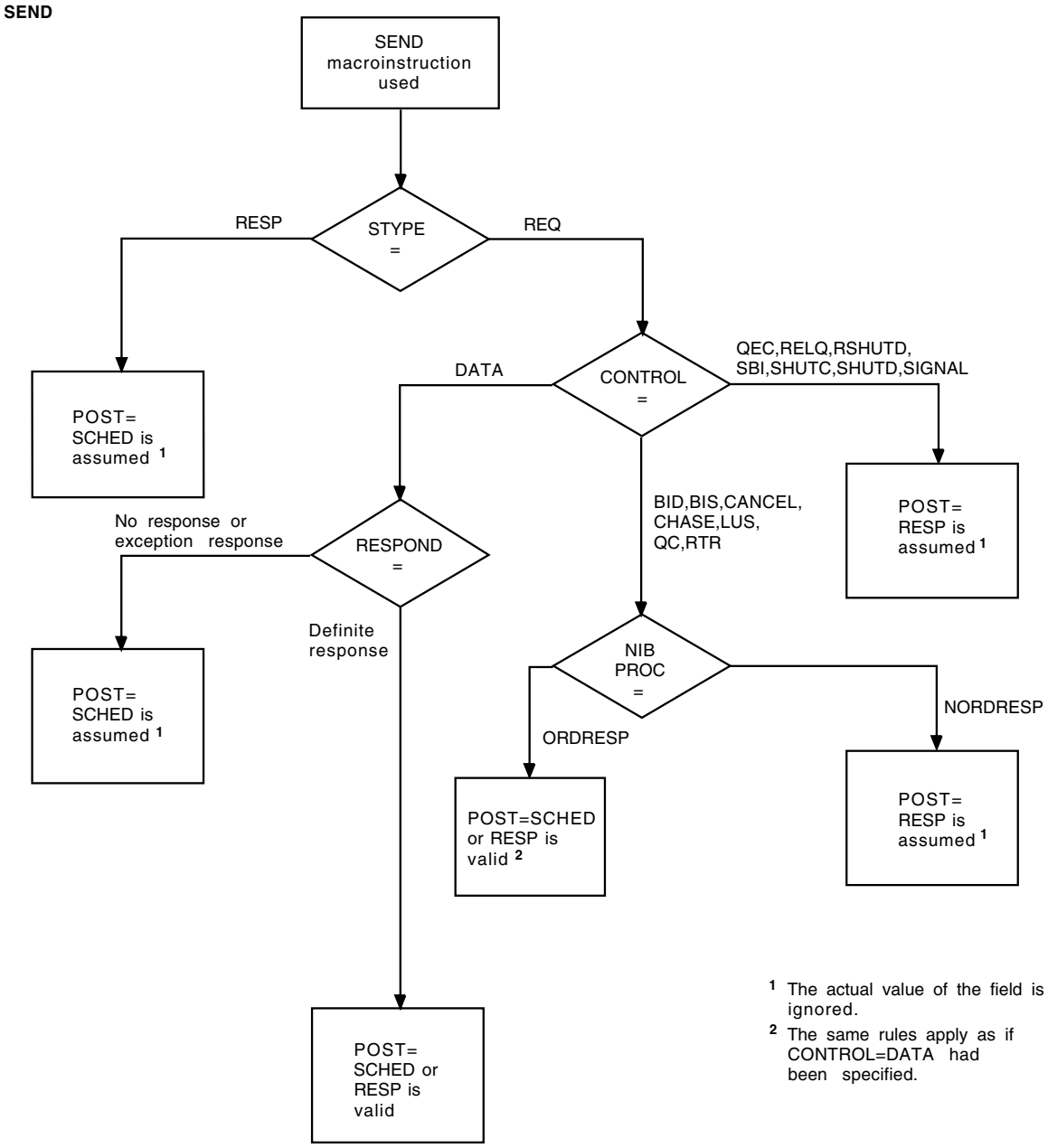


Figure 92. How the POST operand in the SEND macroinstruction is used

For OPTCD=LMPEO, the POST operand applies to the last generated request unit for the SEND. For OPTCD=BUFFLST, this operand applies to the last request unit associated with the buffer list.

POST=RESP

Indicates that the macroinstruction generally completes only when VTAM returns a response for the request on the session. (The macroinstruction is posted complete under certain exception conditions, such as the processing of a Clear, or the receipt of a negative response to an earlier request in the chain, or session termination. A RECEIVE does not obtain the response; the response

information posts in the SEND RPL. The RESPLIM field of the NIB, used to establish the session, limits the number of SEND POST=RESP macroinstructions for normal-flow requests that can be outstanding at one time on the session. If the limit is exceeded, VTAM posts the SEND complete with (RTNCD,FDB2)=(X'14',X'44'). When you specify asynchronous handling (ASY option code in effect) you must use a CHECK macroinstruction before you can reuse the output data area (pointed to by the AREA field).

POST=SCHED
Indicates that the macroinstruction completes as soon as the output data area (pointed to by the AREA field) and the RPL are available for reuse. This occurs prior to the actual transmission of the data. A RECEIVE macroinstruction (or an RESP exit routine) is required to obtain the response. Only one SEND POST=SCHED macroinstruction can be outstanding for a given session at one time. A second SEND POST=SCHED macroinstruction issued before the first has been completed is rejected by VTAM with (RTNCD,FDB2)= (X'14',X'3C'). The limit of one SEND POST=SCHED macroinstruction outstanding at a time for a session does not apply to the sending of responses.

RECLEN=*output_data_length*
If OPTCD=NBUFFLST, CONTROL=DATA, and STYPE=REQ, RECLEN specifies the number of bytes of data to be sent from the field pointed to by AREA. The maximum number that can be set is decimal 65532, unless OPTCD=LMPEO is specified. If OPTCD=LMPEO and the maximum RU size specified in BIND is not 0 (that is, VTAM splits the message if required), then a larger amount of data can be sent. If the RECLEN field is set to 0, the AREA field is not examined.

If OPTCD=BUFFLST, CONTROL=DATA, and STYPE=REQ, RECLEN specifies the length of the buffer list whose address is in AREA; this must be a nonzero multiple of 16.

RECLEN must be set to 0 for sending exception requests. Refer to "Exception conditions" on page 190.

RESPOND
Details about the RESPOND operand are given in "What a response contains" on page 153. When a request is sent (STYPE=REQ), this field indicates the desired response:

RESPOND=(*x,x,x*,QRESP) and the NIB used to establish the session specified PROC=ORDRESP
The response is sent in order with respect to normal-flow (DFSYN) requests.

RESPOND=(*x,x,x*,NQRESP) or the NIB used to establish the session specified PROC=NORDRESP
The response is sent in order with respect to other responses and not necessarily in order with respect to normal-flow (DFSYN) requests.

RESPOND=(EX,FME,RRN,*x*)
Only negative responses type 1 and 2 are expected (see the following note).

RESPOND=(EX,FME,NRRN,*x*)
Only a negative response type 1 is expected.

RESPOND=(EX,NFME,RRN,*x*)
Only a negative response type 2 is expected.

RESPOND=(EX,NFME,NRRN,*x*)
Invalid. Should not be specified. VTAM assumes (NEX,NFME,NRRN,*x*).

RESPOND=(NEX,FME,RRN,*x*)
Definite response type 1 or 2 is expected (see the following note).

RESPOND=(NEX,FME,NRRN,*x*)
Definite response type 1 is expected.

RESPOND=(NEX,NFME,RRN,*x*)
Definite response type 2 is expected.

RESPOND=(NEX,NFME,NRRN,*x*)
No response expected.

If a normal-flow data-flow-control request is sent, RESPOND=(NEX,FME,NRRN,NQRESP) is assumed unless NIB PROC=ORDRESP is specified in the NIB used to establish the session. In that case, the value in RESPOND is used.

Note: Although VTAM and certain logical units permit response types 1 and 2 to be sent separately, sending multiple responses per chain is not allowed by SNA protocols and should not be used.

If response types 1 and 2 are returned and POST=RESP for the SEND RPL, the first response completes the SEND operation. If the two responses are returned together, both are received as one response; that is, the second response is also reflected in the completed RPL. If the second response does not accompany the first, however, the second response must be received by a separate RECEIVE macroinstruction or by an RESP exit routine.

When a response is sent (STYPE=RESP), this field indicates the response type:

RESPOND=(*x,x,x*,QRESP or NQRESP)
The queued response indicator must be set to the same value as received in the request.

RESPOND=(EX,FME,RRN,*x*)
This is a negative response type 1 or 2.

RESPOND=(EX,FME,NRRN,*x*)
This is a negative response type 1.

RESPOND=(EX,NFME,RRN,*x*)
This is a negative response type 2.

RESPOND=(EX,NFME,NRRN,*x*)
Invalid. Rejected with (RTNCD,FDB2)=(X'14',X'3B').

RESPOND=(NEX,FME,RRN,*x*)
This is a positive response type 1 or 2.

RESPOND=(NEX,FME,NRRN,*x*)
This is a positive response type 1.

RESPOND=(NEX,NFME,RRN,*x*)
This is a positive response type 2.

RESPOND=(NEX,NFME,NRRN,*x*)
Invalid. Rejected with (RTNCD,FDB2)=(X'14',X'3B').

RTYPE
When a SEND is issued, the RTYPE field indicates the type or types of input

for which the session's CA-CS mode is to be switched. For a description of input types, see “DFSYN, DFASY, and RESP types of RUs” on page 160.

SEQNO=*sequence_number*
Specifies the sequence number sent with a request or response. The number specified by this operand is used when sending a normal-flow response or when sending an expedited DFC request or response. The number is not used when sending a normal-flow request; VTAM generates the sequence number in that case.

Usually the sequence number received for a request (and placed into the SEQNO field of the RPL used to receive the request) is the sequence number that should be sent with any response to that request. The correct sequence number to use is dictated by SNA protocols.

SIGDATA=*signal_data*
Contains the signal data that is to be sent when CONTROL=SIGNAL, STYPE=REQ is specified. The signal data can be a decimal, hexadecimal, or character constant of 1–4 bytes or a register (the value in the register is used). If fewer than 4 bytes are specified, the value is padded to 4 bytes as if the constant were an assembler language DC statement with a length attribute of 4.

SSENSEO
This field is set by VTAM for a Logical Unit Status (LUSTAT) request and informs the logical unit of the type of error that caused the exception condition. These error types are described in Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651. If you omit this operand, the SSENSEO field defaults to 0.

This field can also provide application-specified sense values for negative responses to CINIT or for UNBIND. Refer to the sections on the CLSDST or TERMSESS macroinstructions in this chapter for additional information.

SSENSMO=*system-sense_modifier_value*
The value set in this field is used in conjunction with the SSENSEO setting to describe the specific type of error that caused the exception condition. The meanings assigned to the SSENSMO values are described in detail in *SNA Formats* If this operand is omitted, the SSENSMO field is set to 0.

This field can also be used to provide application-specified sense values for negative responses to CINIT or for UNBIND. Refer to the TERMSESS macroinstructions in this chapter.

Specify any decimal integer 0–255 inclusive, or specify a 1-byte hexadecimal constant.

STYPE
Designates whether a request (STYPE=REQ) or a response (STYPE=RESP) is to be sent. The CONTROL field governs the request code if a data-flow-control request or response is sent. Some of these requests are described in more detail in Chapter 6, “Communicating with logical units,” on page 151.

STYPE=REQ
Must be specified if OPTCD=LMPEO or OPTCD=BUFFLST.

If OPTCD=USERRH is specified, the RPL flag set by the STYPE operand is ignored. However, the user RH must specify that the RU is a request unit for OPTCD=LMPEO or OPTCD=BUFFLST.

STYPE=RESP
Sends a response indicating whether a normal or expedited flow has been processed successfully. The response can be either positive or negative.

STYPE=REQ,CONTROL
The following operands show the types of control data requests:

CONTROL=BID
Sends a Bid request. The receiver interprets this as a request on the part of the sender for permission to begin a new bracket on the session.

CONTROL=BIS
Sends a Bracket Initiation Stopped request to indicate that the sender stops initiating any new brackets on the session.

CONTROL=CANCEL
Sends a Cancel request. The receiver might interpret this request as an indication that the receiver should discard the chain that it is receiving on the session. A Cancel request is sent instead of a request indicating CHAIN=LAST when a chain is canceled.

CONTROL=CHASE
Sends a Chase request. When the sender receives a response to this request, it can be certain that no normal-flow requests were previously sent on the session (before CHASE) for which the session partner later returns a response.

CONTROL=DATA
Sends a Data request. This must be specified if OPTCD=LMPEO or OPTCD=BUFFLST is coded.

CONTROL=LUS
Sends a Logical Unit Status (LUSTAT) request. An LUSTAT request can convey the same type of information as does a negative response. It can also convey information about the availability of the sender or its components on this session. An LUSTAT is sent when the sender wants to indicate an exception or a change-of-status condition, but cannot do so with a negative response (for example, the session-partner is sending requests and asking for no responses). The SSENSEO, SSENSMO, and USENSEO fields are used for LUSTAT indicators.

CONTROL=QC
Sends a Quiesce Complete request. This informs the receiver that the sender no longer transmits normal-flow requests on the session. Once this request is sent, no other normal-flow requests can be transmitted to the session partner until the session partner returns a Release Quiesce (RELQ) request.

CONTROL=QEC
Sends a Quiesce at End-of-Chain request. This informs the receiver that when it is through transmitting the current chain, it is to stop transmitting further normal-flow requests on the session and return a Quiesce Complete (QC) request.

CONTROL=RELQ
Sends a Release Quiesce request. This informs the receiver that it can begin transmitting normal-flow requests on the session.

CONTROL=RSHUTD
Sends a Request Shutdown request to the primary end of the session. This requests the primary logical unit to terminate the session (for example, issue a CLSDST macroinstruction).

CONTROL=RTR
Sends a Ready to Receive request. This indicates that the sender is now willing to receive a new bracket on the session.

CONTROL=SBI
Sends a Stop Bracket Initiation request to request that the receiver not begin any new brackets on the session.

CONTROL=SHUTC
Sends a Shutdown Complete request to the primary end of the session. This acknowledges the receipt of a SHUTD request and indicates the secondary logical unit is ready to accept session termination.

CONTROL=SHUTD
Sends a Shutdown request to the secondary end of the session. The secondary logical unit interprets this as an indication that the primary logical unit is about to terminate the session. When a secondary logical unit is ready to accept session termination, it returns a Shutdown (SHUTC) request.

CONTROL=SIGNAL
Sends a Signal request containing the 4 bytes of signal data in the SIGDATA field of the RPL.

STYPE=RESP,CONTROL
Sends a response indicating whether a previously received normal-flow request has been processed successfully. A positive or negative response of any type (definite response 1, 2, or both) can be sent; however, only certain combinations are valid within SNA protocols. The CONTROL field operands, DATA, BID, BIS, CANCEL, CHASE, LUS, QC, and RTR, are explained in the preceding section.

STYPE=RESP,CONTROL
Sends a response indicating whether a previously received expedited-flow data-flow-control request has been processed successfully. Only a positive or negative definite response 1 is allowed. The CONTROL field operands, QEC, RELQ, RSHUTD, SBI, SHUTC, SHUTD, and SIGNAL, are explained in the preceding section.

Note: A response to an expedited-flow data-flow-control request on a session can be sent only if the NIB used to establish the session specified PROC=APPLRESP; otherwise, VTAM sends the response on behalf of the application program. Sending or receiving a Clear request eliminates the requirement to send a response to a previously received expedited-flow data-flow-control request.

USENSEO=*user-sense_value*
This field is set by VTAM for a Logical Unit Status (LUSTAT) request. In most instances, the user-sense field is user-defined and can be used to inform the logical unit that an exception condition is being indicated for an application-program-related error that is not an SNA-defined error, or it can be used to further modify the SNA-defined system-sense and system-sense modifier values. See Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651, for more information. If this operand is omitted, the USENSEO field is set to 0.

This field can also be used to provide application-specified sense values for negative responses to CINIT or for UNBIND. Refer to the TERMSESS macroinstructions in this chapter.

Specify any decimal integer 0–65535 inclusive or specify a 2-byte hexadecimal or character constant.

Examples

```
SEND1    SEND   RPL=RPL1,STYPE=REQ,CONTROL=DATA,                      C
                AREA=OUTBUF,RECLEN=60,CHAIN=ONLY,                     C
                RESPOND=(EX,FME,NRRN,NQRESP),POST=SCHED
```

SEND1 sends a 60-byte data request on the session identified in RPL1's ARG field. SEND1 is completed as soon as VTAM has scheduled the output operation and OUTBUF and RPL1 are available for reuse. The RESPOND field indicates that only negative response type 1 should be returned; that is, if the request is processed normally, no response is returned. A RECEIVE RTYPE=RESP macroinstruction (or RESP exit routine) is required to obtain the negative response, if one is returned.

Completion information

The actual or implied setting of the POST field governs what constitutes the "completion" of the SEND operation. Refer to the description of the POST operand in the preceding section for details.

After the SEND operation is completed, the following RPL fields are set:

- The value 34 (decimal) is set in the REQ field, indicating a SEND request.
- The sequence number is placed in the SEQNO field except when OPTCD=LMPEO. For OPTCD=LMPEO, SEQNO is set by VTAM with the sequence number of the last RU generated from the message, and OBSQVAL is set by VTAM with the sequence number of the first RU generated from the message.
- The USER field contains the value that was set in the USERFLD field of the NIB when the session was established.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

The following fields can also be set when a response has been received (POST=RESP):

- The RESPOND field indicates the type of response that has been returned. This field is set in exactly the same manner as indicated in the preceding discussion for sending a response.
- The control field is set to the request code value received in the response. However, this should be the same value as was in the original SEND if the logical unit obeys SNA protocols.
- If a negative response has been returned, the SSENSEI, SSENSMI, and USENSEI fields are set indicating system-sense information, system-sense modifier, and user-sense information. For additional information about these fields, refer to Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

SENDCMD—Send a VTAM operator command to VTAM

Purpose

It is possible for an application program to send VTAM operator commands to VTAM and to reply to messages sent to the application program by VTAM. The SENDCMD macroinstruction permits an authorized application program to send the following commands: VARY, DISPLAY, MODIFY, and REPLY.

Note: The HALT and START commands cannot be issued using the SENDCMD macroinstruction. They can be issued only by the VTAM operator at the system console.

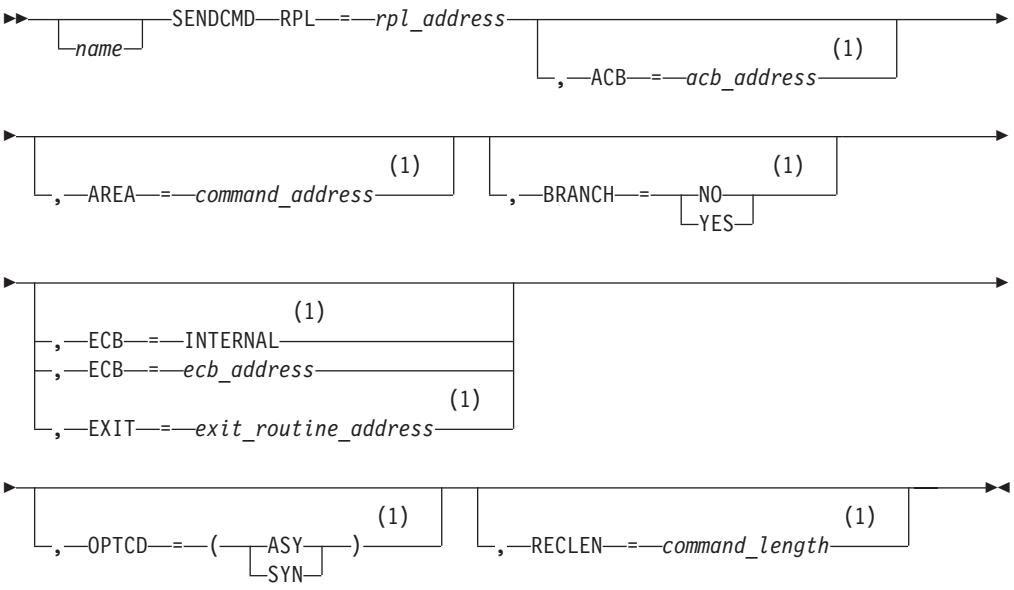
Usage

Before issuing the SENDCMD macroinstruction the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

For information on writing an application program that can issue VTAM operator commands and receive VTAM messages, see Appendix L, "Program operator coding requirements," on page 875.

VTAM receives control from the SENDCMD macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing SENDCMD is to perform.

The following RPL operands apply to the SENDCMD macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing SENDCMD.

AREA=*command_address*
Indicates the address of the header and command to be sent to VTAM.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) SENDCMD operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) SENDCMD operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

OPTCD=SYN
OPTCD=ASY
If the SYN option code is set, control is returned to the application program when the SENDCMD operation has been completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. After the SENDCMD operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the SENDCMD operation, the SYN option should not be used if suspending the SENDCMD-issuing task or SRB for this time is undesirable. Instead, the ASY option code should be used.

RECLEN=*command_length*
Indicates the number of bytes to be sent to VTAM. The number specified must be equal to the length of the header and command pointed to by the AREA field. If the RECLEN field is set to 0, the AREA field is not examined. The maximum length is 130.

Examples

```
SENDCMD1 SENDCMD RPL=RPL1,AREA=VARYCMD1,RECLEN=CMDLEN
         .
         .
         .
VARYCMD1 DC    00          HEADER
         DC    03          STATUS FIELD
         DC    0001        IDENTIFICATION NUMBER
         DC    C'VARY NET,ACT,ID=LU1' COMMAND
CMDEND   EQU   *
CMDLEN   EQU   CMDEND-VARYCMD1
```

SENDCMD1 sends a message to VTAM, consisting of the area between VARYCMD1 and CMDEND, instructing it to activate logical unit LU1. The status field indicates that a reply is returned to the application program. See “Data exchanged between a program operator and VTAM” on page 883 for information pertaining to the header and status field settings.

Completion information

A SENDCMD operation is successfully completed when the command is passed to VTAM for subsequent processing.

After the SENDCMD operation is completed, the following RPL fields are set:

- The value 39 (decimal) is set in the REQ field, indicating a SENDCMD request.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

SESSIONC—Send a session-control request or response

Purpose

SESSIONC sends a Start Data Traffic (SDT), Clear, or Set and Test Sequence Numbers (STSN) request on a session from an application program acting as a PLU. SESSIONC is also used to send a Request Recovery (RQR) request as well as responses to rejected BIND requests, SDT requests, and STSN requests on a session from an application program acting as an SLU.

Usage

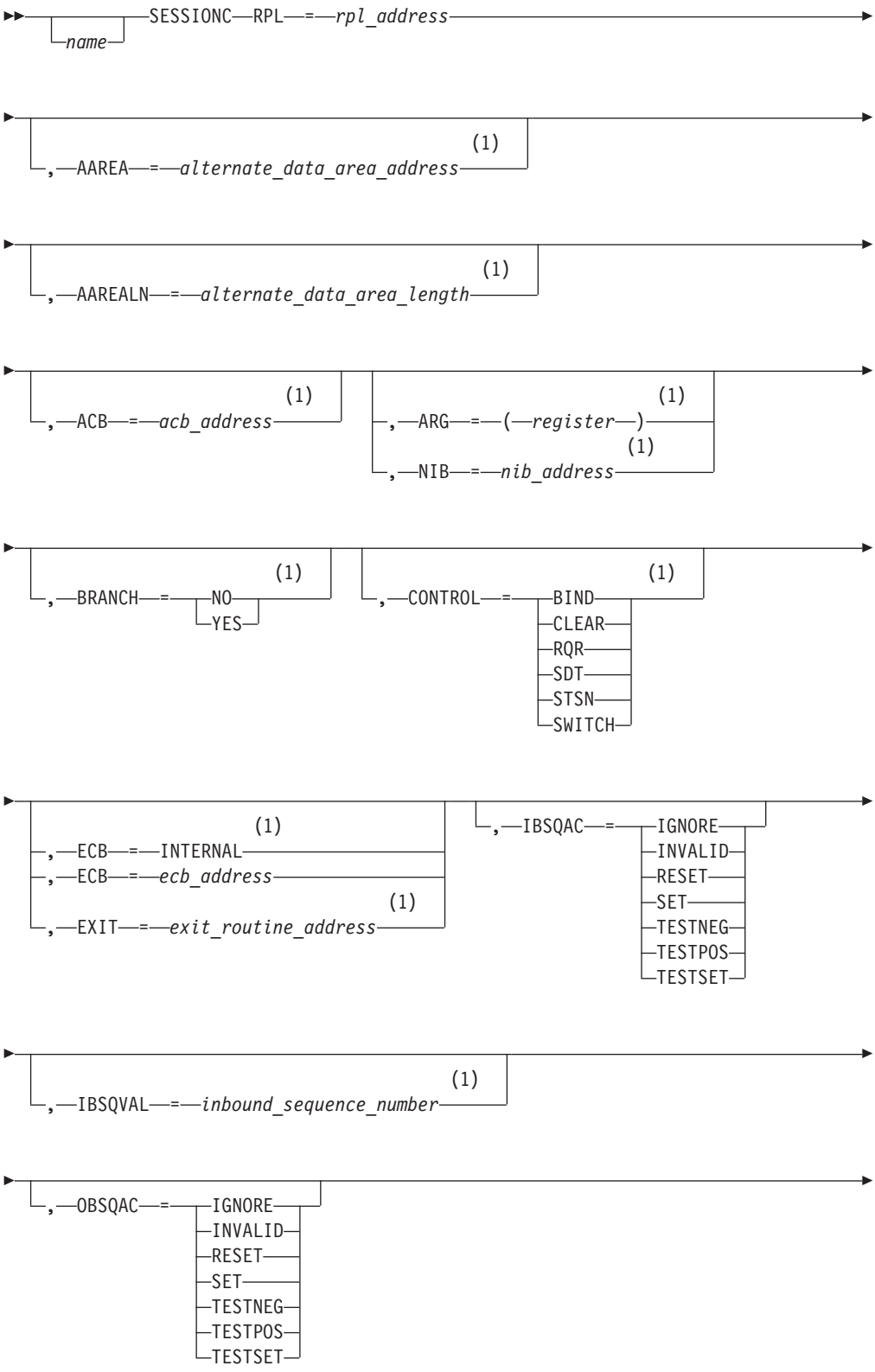
For XRF, the SESSIONC macroinstruction initiates the switch from backup session status to primary session status. SESSIONC uses a CONTROL operand to specify this "switch."

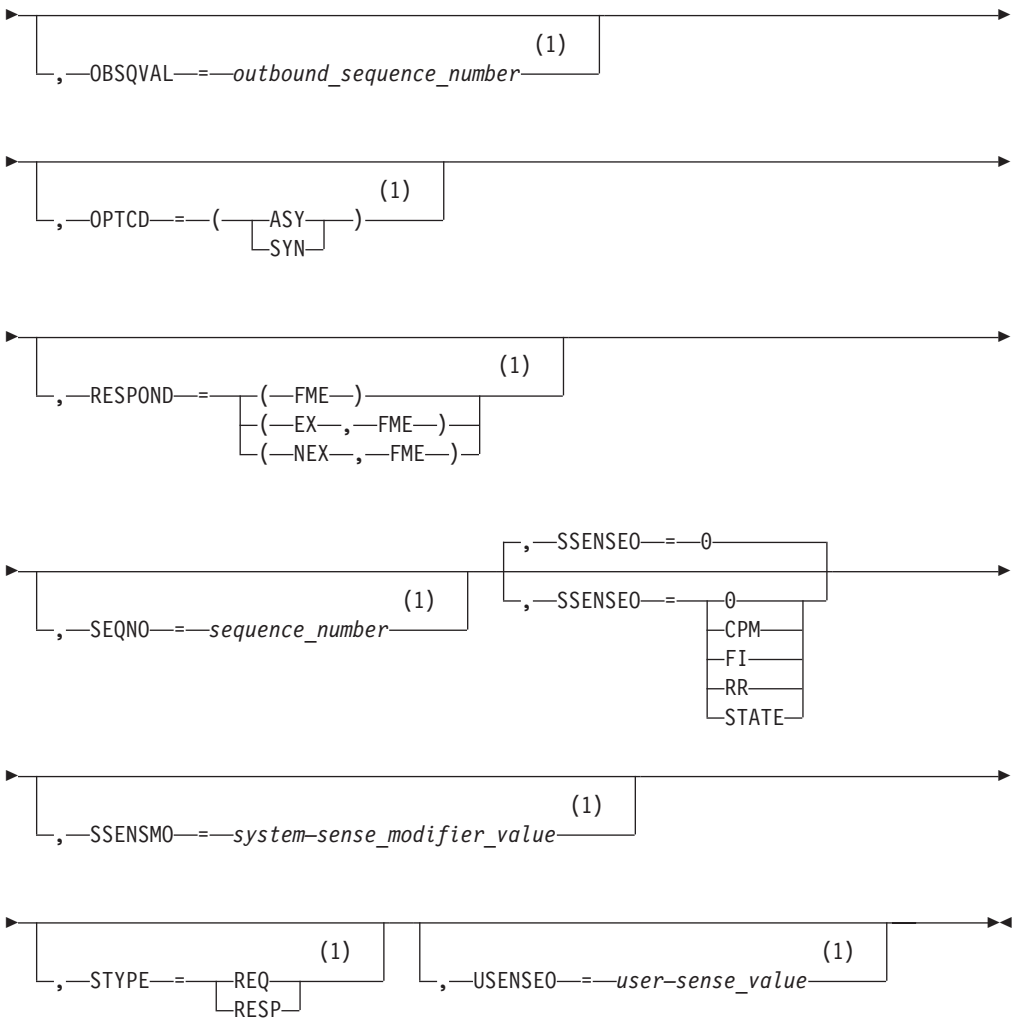
Material from "Send a Start Data-Traffic request to the SLU" on page 540 through "Send a Switch request initiating the switch from backup to primary session status" on page 541 shows the SESSIONC options. The transmission services profile that is in use determines the use of these requests. See Appendix F, "Specifying a session parameter," on page 793, for a description of the profiles and the requests that can be used. Examples of the use of session-control requests and responses sent by SESSIONC are given in Appendix D, "Request and response exchanges for typical communication operations," on page 693, and in "Controlling flow" on page 164.

Before issuing the SESSIONC macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

SESSIONC -RSP(BIND): If PARMS=(NQNAMES=YES) on the ACB macroinstruction, and if the NIB is specified with a network identifier in the NIBNET field, the network identifier is used along with the LU name in NIBSYM to determine the target of the UNBIND or BIND response.

Syntax


name
SESSIONC
RPL
=
rpl_address
,
AAREA
=
alternate_data_area_address
(1)
,
AAREALN
=
alternate_data_area_length
(1)
,
ACB
=
acb_address
(1)
,
ARG
=
(
register
)
(1)
,
NIB
=
nib_address
(1)
,
BRANCH
=
NO
YES
(1)
,
CONTROL
=
BIND
CLEAR
RQR
SDT
STSN
SWITCH
(1)
,
ECB
=
INTERNAL
,
ECB
=
ecb_address
,
EXIT
=
exit_routine_address
(1)
,
IBSQAC
=
IGNORE
INVALID
RESET
SET
TESTNEG
TESTPOS
TESTSET
,
IBSQVAL
=
inbound_sequence_number
(1)
,
OBSQAC
=
IGNORE
INVALID
RESET
SET
TESTNEG
TESTPOS
TESTSET



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing SESSIONC performs.

The following RPL operands apply to the SESSIONC macroinstruction:

AAREA=*alternate_data_area_address*
For XRF, the AAREA field in the RPL is initialized by the application program to provide the address of the input area where the SWITCH response information should be placed.

AAREALN=*alternate_data_area_length*
For XRF, the AAREALN field in the RPL is initialized by the application program to contain the length of the input area pointed to by AAREA. The AAREA field is ignored if AAREALN=0 (no switch response information is returned).

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing SESSIONC. If not specified, the ACB address already in RPLDACB is used.

ARG=(register)
The SESSIONC macroinstruction is always directed to one specific session. The ARG operand specifies the register containing the CID of the session. If the ARG operand is not specified, the CID already in the RPLARG field is used. The RPL form of SESSIONC can be used for rejecting a BIND by specifying RPLARG equal to the CID, as provided in the SCIP exit when the BIND request was received.

Note: If your application uses the RPL DSECT, IFGRPL, you must clear the RPLNIB bit if a CID is being inserted into the RPLARG field. For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH
If the macroinstruction is issued in an application program that is running in supervisor state under a TCB, set BRANCH to YES to specify that you should use authorized path processing.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

CONTROL

CONTROL=BIND
Sends a BIND request-rejected response to a primary logical unit to indicate a rejection of the BIND request.

CONTROL=CLEAR
Sends a Clear request on the session. All SEND, RECEIVE, RESETSR, and SESSIONC requests in progress for the session are completed normally or with (RTNCD,FDB2)=(0C,0C). If the transmission services profile supports SDT, all subsequent SEND, RECEIVE, and RESETSR requests are rejected with (RTNCD,FDB2)=(14,41) until SDT is sent and a positive SDT response is received. Before SESSIONC is completed, VTAM sets the inbound and outbound sequence numbers to 0.

CONTROL=RQR
Sends a Request Recovery request to the primary logical unit. This operand asks the primary logical unit to take recovery action for this session (for example, issue Clear, STSN, and SDT).

CONTROL=SDT
Sends a Start Data Traffic request or a response to an SDT on the session. When SDT=SYSTEM is coded as part of the NIB used to establish the session, VTAM automatically sends a Start Data Traffic request or an SDT response as part of the session-establishment process. If SDT=APPL is coded instead, it is the application program's responsibility to send the request or response when data traffic is to begin.

CONTROL=STSN
Sends a Set and Test Sequence Numbers request (if STYPE=REQ) to the logical unit acting as the secondary end of the session or returns information (if STYPE=RESP) to the primary logical unit in response to an STSN request.

CONTROL=SWITCH
Causes the backup XRF session to become the primary XRF session.

The former primary XRF session, if still "active," is terminated with an UNBIND (CLEANUP). This command can be issued only on a backup XRF session. If issued on a primary XRF session, it is rejected.

After the positive response to the SWITCH is received by VTAM and the operation is posted complete, the session recovery can proceed.

The SWITCH response contains the status of the session at the time the SWITCH is completed. The application program uses this information (found in control vector hex 29) to recover the session. See Table 110 on page 747 for more information.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) SESSIONC operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) SESSIONC operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

IBSQAC and OBSQAC
The IBSQAC (inbound sequence number action code) and the OBSQAC (outbound sequence number action code) fields designate the type of STSN request to be sent on the session. The application program can set either or both of these fields. The effect of setting one is identical to the effect of setting the other, except that one applies to incoming requests and the other to outgoing requests. Table 93 on page 542 summarizes the STSN request types (SET, TESTSET, INVALID and IGNORE) and the responses they can elicit from the logical unit at the other end of the session.

IBSQAC
OBSQAC
The IBSQAC (inbound sequence number result code) and the OBSQAC (outbound sequence number result code) fields designate the type of STSN response to be sent on the session. The result codes are set in response to the action codes set in a previously received STSN request as shown in Table 93 on page 542. The STSN request types are: TESTPOS, TESTNEG, INVALID, and RESET.

IBSQVAL=*inbound_sequence_number*
Indicates a value that is one less than the new value that VTAM is to begin assigning to inbound requests. This value must be a decimal integer 0—65535 inclusive. This field can be sent on either an STSN request or a STSN response, as shown in Table 93 on page 542.

Note: *Inbound* refers to requests that are transmitted from the terminal to the application.

NIB=nib address
When an application program wants to return a request-rejected response to a BIND request, the NIB address indicates the NIB whose NAME field identifies a pending active session to be terminated, and if using network-qualified names, whose NIBNET field contains the network identifier of the logical unit. Alternatively, the ARG operand can be used to identify the session.

For a detailed description of the use of the SESSIONC macroinstruction to send a BIND request rejected response, refer to "SESSIONC macroinstruction with CONTROL=BIND" on page 98.

Note: If your application uses the RPL DSECT, IFGRPL, you must set the RPLNIB bit if an NIB address is being inserted into the RPLARG field.

OBSQVAL=*outbound_sequence_number*
Indicates a value that is one less than the new value that VTAM is to begin assigning to outbound (from the application program) requests. This value must be between a decimal integer 0–65535 inclusive. This field can be sent on either a STSN request or a STSN response as shown in Table 93 on page 542.

Note: *Outbound* refers to requests that are transmitted from the application program to a device.

OPTCD=SYN
OPTCD=ASY
If the SYN option code is set, control is returned to the application program when the SESSIONC operation has completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. After the SESSIONC operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field.

Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the SESSIONC operation, you should not use the SYN option if suspending the SESSIONC issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

For XRF, ASY is recommended for CONTROL=SWITCH.

RESPOND
Indicates whether a negative or positive response is to be sent for a session-control request. See "How requests and responses are exchanged" on page 157 for possible responses that can be sent.

SEQNO=*sequence_number*
Indicates the sequence number of a response. The application program using SESSIONC to respond to any session-control request must set this field with

the sequence number of that request. The sequence number was made available in the SCIP exit routine when the session-control request was received.

SSENSEO
This field is set by VTAM for a Logical Unit Status (LUSTAT) request and informs the logical unit of the type of error that caused the exception condition. These error types are described in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651. SSENSEO=0 is the default.

This field can also provide application-specified sense values for negative responses to CINIT or for UNBIND. Refer to the sections on the CLSDST or TERMSESS macroinstructions in this chapter for additional information.

SSENSMO=*system-sense_modifier_value*
The value set in this field is used in conjunction with the SSENSEO setting to describe the specific type of error that caused the exception condition. The meanings assigned to the SSENSMO values are described in detail in *SNA Formats* If this operand is omitted, the SSENSMO field is set to 0.

This field can also be used to provide application-specified sense values for negative responses to CINIT or for UNBIND. Refer to the TERMSESS macroinstructions in this chapter.

Specify any decimal integer 0–255 inclusive, or specify a 1-byte hexadecimal constant.

STYPE
This field designates the type of output to be sent on the session.

STYPE=REQ
The application program uses this to send a request.

STYPE=RESP
Is used when a response is to be sent.

For XRF, only REQ applies to CONTROL=SWITCH.

USENSEO=*user-sense_value*
Indicates sense information related to a negative response to an SDT, STSN request, or a BIND request rejected response. Refer to the RPL macroinstruction description of these operands for format information. Additionally, register notation can be used for SSENSMO and USENSEO; the low-order 1 or 2 bytes, respectively, are used from the register.

Examples

```
SESSC1   SESSIONC RPL=RPL1,
CONTROL=STSN,OBSQAC=TESTSET,                C
              OBSQVAL=(3),IBSQAC=IGNORE
```

SESSC1 sends an STSN request to a logical unit on a session and sets the VTAM-supplied outbound (from the application program) sequence number to the value contained in the low-order 2 bytes of register 3. The logical unit, noting that the type of STSN request is TESTSET, can indicate TESTPOS, TESTNEG, INVALID, or RESET with its response. The response information is available in RPL1 when SESSC1 is completed. If OBSQAC is found by the application program to be set to TESTPOS or TESTNEG, the OBSQVAL field contains the logical unit's version of the outbound sequence number. No action is taken by either end on the inbound (to the PLU) sequence number.

Completion information

If STYPE=REQ, the SESSIONC operation is successfully completed when the request has been sent and a response has been returned and posted in the RPL (similar to SEND POST=RESP). If STYPE=RESP, the SESSIONC operation is successfully completed when the response is moved to VTAM buffers and the RPL is available for reuse (similar to SEND POST=SCHED).

After the SESSIONC operation is completed, the following RPL fields are set:

- The value 37 (decimal) is set in the REQ field, indicating a SESSIONC operation.
- The value originally set in the USERFLD field of the NIB used to establish the session is set in the USER field of the RPL. The USER field is not set when SESSIONC is used to send a request rejected response to BIND.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

If a CONTROL value of other than BIND, RQR, SDT, CLEAR, STSN, or SWITCH is used, then return code (RTNCD,FDB2) "CONTROL not valid" is issued.

If the SESSIONC is issued for a session that has been terminated, a return code (RTNCD,FDB2)=(0C,0B) can be posted.

For unauthorized application programs for which CONTROL=SWITCH is specified, if AAREALN is not zero and the area pointed to by AAREA and delimited by AAREALN is not entirely within user storage, then return code (RTNCD,FDB2)= (14,1E) is posted.

If CONTROL=SWITCH is specified and AAREALN specifies a length greater than zero but less than the length of control vector hex 29, then the returned data is truncated and return code (RTNCD,FDB2)=(00,05) is posted.

If SESSIONC is used to send a session-control request, when the macroinstruction is posted complete, the following RPL fields are set:

- The SEQNO field is set to the sequence number assigned to the session-control request.
- The CONTROL field is set to the value received in the response; however, this should be the same request code value as was in the original SESSIONC.
- Additionally, if SESSIONC was originally used to send a STSN request, either the IBSQAC or OBSQAC fields (or both) are usually set to TESTPOS, TESTNEG, INVALID, or RESET depending on the action codes initially set in these fields when SESSIONC was issued. Table 93 on page 542 lists the result codes that can be returned for each action code initially set. Either the IBSQVAL or OBSQVAL fields (or both) contain a sequence number when the corresponding IBSQAC or OBSQAC result code is set to TESTPOS or TESTNEG.
- If the macroinstruction returns an error code, the SSENSEI, SSENSMI, and USENSEI field can be set indicating system-sense information, system-sense modifier, and user-sense information. See Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651 for more information about these fields.

Send a Start Data-Traffic request to the SLU

About this task

```
CONTROL=SDT,STYPE=REQ
```

The application program sends an SDT request to a secondary logical unit to allow the flow of data or data-flow-control requests or responses to begin (or resume) on the session.

Send a Clear request to the SLU

About this task

```
CONTROL=CLEAR,STYPE=REQ
```

The application program sends a Clear request to a secondary logical unit. This resets sequence numbers, prevents the flow of data or data-flow-control requests or responses (if required by the transmission services profile), and discards any such requests or responses in the network for this session. All pending SEND and RECEIVE SPEC macroinstructions for the session are canceled with (RTNCD,FDB2)= (0C,0C).

Send a Request Recovery request to the PLU

About this task

```
CONTROL=RQR,STYPE=REQ
```

Request the primary logical unit to take a recovery action appropriate for this session (such as issuing Clear and STSN).

Send a Set and Test Sequence Number request to the SLU

About this task

```
CONTROL=STSN,STYPE=REQ
```

When the SESSIONC macroinstruction is issued, the settings of the IBSQAC and OBSQAC fields determine what happens to the associated sequence numbers.

Setting of IBSQAC or OBSQAC field on request	Action on associated sequence number at application program
SET	Set the sequence number for the session specified value and send it to the logical unit.
TESTSET	Set the sequence number for the session to the specified value, send it to the logical unit, and obtain a sequence number from the logical unit for the session.
INVALID	Do not change the current sequence number. Obtain the sequence number from the logical unit for the session.
IGNORE	Do not change the current sequence number.

Send the PLU a response to a Set and Test Sequence Number (STSN) request

About this task

```
CONTROL=STSN,STYPE=RESP,RESPOND=(EX or NEX,FME)
```

The settings of the IBSQAC/OBSQAC fields received by the logical unit cause it to take the following action on the sequence numbers for the session and respond appropriately:

Setting of IBSQAC or OBSQAC on request	Action on associated sequence number at secondary logical unit	Possible responses to request
SET	Set the sequence number to the specified value for the session.	TESTPOS
TESTSET	Compare the received value to the current sequence number and respond accordingly. Set the sequence number to the specified value for the session.	TESTPOS TESTNEG INVALID RESET
INVALID	Return sequence number information to the sender of STSN.	TESTNEG INVALID RESET
IGNORE	Do not change sequence numbers. Ignore this part of the request.	TESTPOS

Send the PLU a Request-Rejected response to a BIND request

About this task

```
CONTROL=BIND,STYPE=RESP
```

Indicates that the BIND request is unacceptable to the application program (for example, the session parameters are not valid or the application program does not request a session with the PLU specified in the BIND).

Send the PLU a response to an SDT request

About this task

```
CONTROL=SDT,STYPE=RESP,RESPOND=(EX or NEX,FME)
```

Indicates that the application program accepts or rejects an SDT. If a positive response is sent, requests and responses for data and data-flow-control requests can now be sent on the session. If a negative response is sent, sense information can be included to indicate the reason for the rejection.

Send a Switch request initiating the switch from backup to primary session status

About this task

```
CONTROL=SWITCH,STYPE=REQ
```

The application program sends a Switch request that causes the backup XRF session to become the primary XRF session. The former primary XRF session, if still “active”, is terminated with an UNBIND(CLEANUP).

Either end of a session has the ability to check its inbound and outbound sequence numbers and, if necessary, to suspend traffic flow and ask that correct numbers be reestablished. When the primary logical unit requests to reestablish correct sequence numbers, it issues a Clear (if allowed in the TS profile for the session) followed by an STSN request. If the logical unit acting as the secondary end of the session requests to reestablish correct sequence numbers, it issues an RQR request. The PLU application program then issues the Clear and STSN requests.

The PLU application program can send four STSN options (CONTROL=STSN,STYPE=REQ) to the logical unit: SET, TESTSET, INVALID, and IGNORE. The logical unit acting as the secondary end of the session responds with an STSN response (CONTROL=STSN, STYPE=RESP) as shown in Table 93. A SESSIONC macroinstruction can be used to send STSN requests that apply to either the inbound or the outbound sequence numbers, or that apply to both independently.

Table 93. Types of STSN requests and their possible responses

PLU STSN request	**Meaning**	**SLU STSN response(s)**	**Meaning**
Set (01) (set)	CPMGR sequence number at PLU and SLU is to be set to value in IBSQVAL or OBSQVAL field.	TESTPOS (01) (ignore)	Secondary logical unit accepts value. No value is returned.
TESTSET (11) (set and test)	CPMGR sequence number at PLU and SLU is to be set to value in IBSQVAL or OBSQVAL field. The SLU NAU sequence number is to be tested against the value in the IBSQVAL or OBSQVAL field; the SLU NAU sequence number is returned in the response.	TESTPOS (01)	Secondary logical unit agrees with value. Value (=SLU NAU sequence number) is returned in IBSQVAL or OBSQVAL field.
		TESTNEG (11)	Secondary logical unit disagrees with value. SLU NAU sequence number is returned in IBSQVAL or OBSQVAL field.
		INVALID (10)	Secondary logical unit does not maintain or has lost its NAU sequence number. No value returned.
		RESET (00)	Meaning depends on LU type.
INVALID (10) (sense)	Primary logical unit is not setting or testing the sequence number for the associated flow. The CPMGR sequence number is not changed. The SLU sequence number is to be returned in the response.	TESTNEG (11)	Secondary logical unit returns NAU sequence number in IBSQVAL or OBSQVAL field.
		INVALID (10)	Secondary logical unit does not maintain or has lost its NAU sequence number. No value returned.
		RESET (00)	Meaning depends on LU type.
IGNORE (00)	Primary logical unit is not setting or testing the sequence number for the associated flow. The CPMGR sequence number is not changed.	TESTPOS (01)	Secondary logical unit ignores this part of the STSN request. No value is returned.

Table 93. Types of STSN requests and their possible responses (continued)

PLU STSN request	Meaning	SLU STSN response(s)	Meaning

Notes:

1. The numbers in parentheses are the corresponding bit settings for the action and result codes in the STSN RU. The names in parentheses are the associated SNA names for the action and result codes. Refer to *SNA Formats* for the format of the STSN RU. If there is no name in parentheses, SNA does not currently have an abbreviated name for the result code.
2. Responses to PLU requests are returned in the RPL fields (IBSQAC and OBSQAC) originally used to contain the request.
3. The term CPMGR sequence number, as used in this table, refers to the sequence number assigned by VTAM when a normal-flow request is sent or received. This is a number returned to the application program in the SEQNO field of the SEND or RECEIVE RPL. The term NAU sequence number refers to a related sequence number kept by the logical unit. For example, for a NAU outbound sequence number, the logical unit might keep a log on a disk file of the CPMGR outbound sequence number assigned to the last request that the logical unit sent for which it received a positive response. Similarly, the NAU inbound sequence number might be the inbound CPMGR sequence number of the last normal-flow request received by the logical unit for which it sent a positive response. The NAU sequence numbers are of interest during recovery operations, because they can be used to let each logical unit in the session understand which RUs have been successfully processed by its session partner.

   The inbound and outbound sequence numbers are handled independently by SESSIONC. Either one or both can be set by a single STSN request.

When an STSN, SDT, or Clear request is sent to the logical unit, a response type 1 is returned as part of the SESSIONC operation. That is, the request is sent as though POST=RESP and RESPOND=(NEX,FME,NRRN) had been specified on a SEND macroinstruction. If a negative response is returned, the SSENSEI, SSENSMI, and USENSEI fields are set as they would be for SEND POST=RESP.

SETLOGON—Modify an application program's capability to establish sessions

Purpose

The SETLOGON macroinstruction indicates:

- Whether VTAM is to schedule an application program's LOGON exit routine when a CINIT request is received
- Whether an application program is enabled to act as an SLU
- The application's use of persistent sessions or generic resources.

For further information on some of the terms used in the following paragraphs, refer to “Defining LUs” on page 81 and to Table 6 on page 82. See also the OPTCD=APPSTAT section of the INQUIRE macroinstruction description in this chapter.

Usage

The nine types of SETLOGON requests are START, STOP, QUIESCE, HOLD, NPERSIST, PERSIST, GNAMEADD, GNAMEDEL, and GNAMESUB. The option code in SETLOGON's RPL determines which type is used.

Although SETLOGON START, STOP, QUIESCE, HOLD, GNAMEADD, GNAMEDEL, and GNAMESUB cannot be used if the ACB was opened with MACRF=NLOGON, SETLOGON PERSIST and NPERSIST can be used.

The START version of SETLOGON causes any application program issuing INQUIRE OPTCD=APPSTAT to be told that your application program is active.

The first SETLOGON OPTCD=START issued after OPEN has been executed causes VTAM to schedule the LOGON exit routine for each CINIT request that has been received. It also allows subsequent CINIT requests to schedule the LOGON exit routine. The LOGON exit routine cannot be scheduled before the first SETLOGON OPTCD=START; therefore, any CINITs received prior to SETLOGON OPTCD=START are queued.

Additionally, the first SETLOGON OPTCD=START causes VTAM to mark the application program enabled for sessions in which it acts as the secondary logical unit. Before the first SETLOGON OPTCD=START occurs, the application program is disabled for sessions in which it is to act as the secondary logical unit. If queuing is specified by the session initiator, an attempt to initiate such a session is queued; otherwise, it is rejected. The first SETLOGON OPTCD=START causes any such queued session to become pending active, which should eventually result in a BIND request received in the SCIP exit routine. Note that the SCIP exit routine cannot be scheduled with a BIND request before the first SETLOGON OPTCD=START occurs.

Subsequent SETLOGON START requests can be used in conjunction with SETLOGON HOLD to pace session setup requests. (You might want to use pacing in the event that storage shortages occur.) When SETLOGON START is issued after SETLOGON HOLD, VTAM schedules the LOGON exit for each queued CINIT request, and schedules the SCIP exit for each queued BIND request. VTAM continues to drive the LOGON and SCIP exits as usual until the application issues SETLOGON HOLD or SETLOGON QUIESCE.

The STOP version of SETLOGON does not stop the scheduling of the LOGON or SCIP exit routines. However, any application program issuing INQUIRE OPTCD=APPSTAT for your ACB is told that new sessions should not be initiated with your application program. Thus, the STOP version can be used only for private application program protocols and is not enforced by VTAM.

The QUIESCE version of SETLOGON causes VTAM to indicate that the application program is inhibited for any new sessions. The only way to initiate new sessions is to close and reopen the ACB. However, any CINIT or BIND requests already queued at the application program logical unit are unaffected by SETLOGON OPTCD=QUIESCE. Thus, for these pending active sessions, OPNDST OPTCD=ACCEPT and OPNSEC can complete normally. An application program might want to use this type of SETLOGON at the end of a day's work, prior to closing the ACB; this would give the application program a chance to handle its current load of active and pending active sessions without any new ones being initiated. Any application program issuing INQUIRE OPTCD=APPSTAT for your ACB is told that your application program is shutting down and cannot accept new sessions. Any attempts to initiate sessions with your application program are rejected.

You can use the SETLOGON macroinstruction with the option codes, OPTCD=HOLD and OPTCD=START, to pace session setup requests. SETLOGON HOLD causes all subsequent CINIT and BIND requests to be queued and prevents

the scheduling of the LOGON and the SCIP exits for session setup requests. When SETLOGON START is issued after SETLOGON HOLD, VTAM schedules the LOGON exit for each queued CINIT request and the SCIP exit for each queued BIND request. VTAM continues to drive the LOGON and SCIP exits as usual until the application issues SETLOGON HOLD or SETLOGON QUIESCE.

Note: An application program should not issue a SETLOGON OPTCD=QUIESCE to inhibit sessions when VTAM is halting; VTAM automatically prevents any new sessions from being initiated.

For this application program, GNAMEADD indicates that VTAM recognizes an association between the network name in the ACB macroinstruction and the generic name coded on the GNAME parameter of the NIB macroinstruction; GNAMEDEL causes VTAM to terminate that association. The SETLOGON OPTCD=GNAMEADD must be issued before SETLOGON OPTCD=START is issued. The application can specify whether it wants to own the affinities for all the sessions that are established with it. The application does this by using the AFFIN keyword of the NIB macroinstruction. For more details, see "NIB—Create a node initialization block" on page 433.

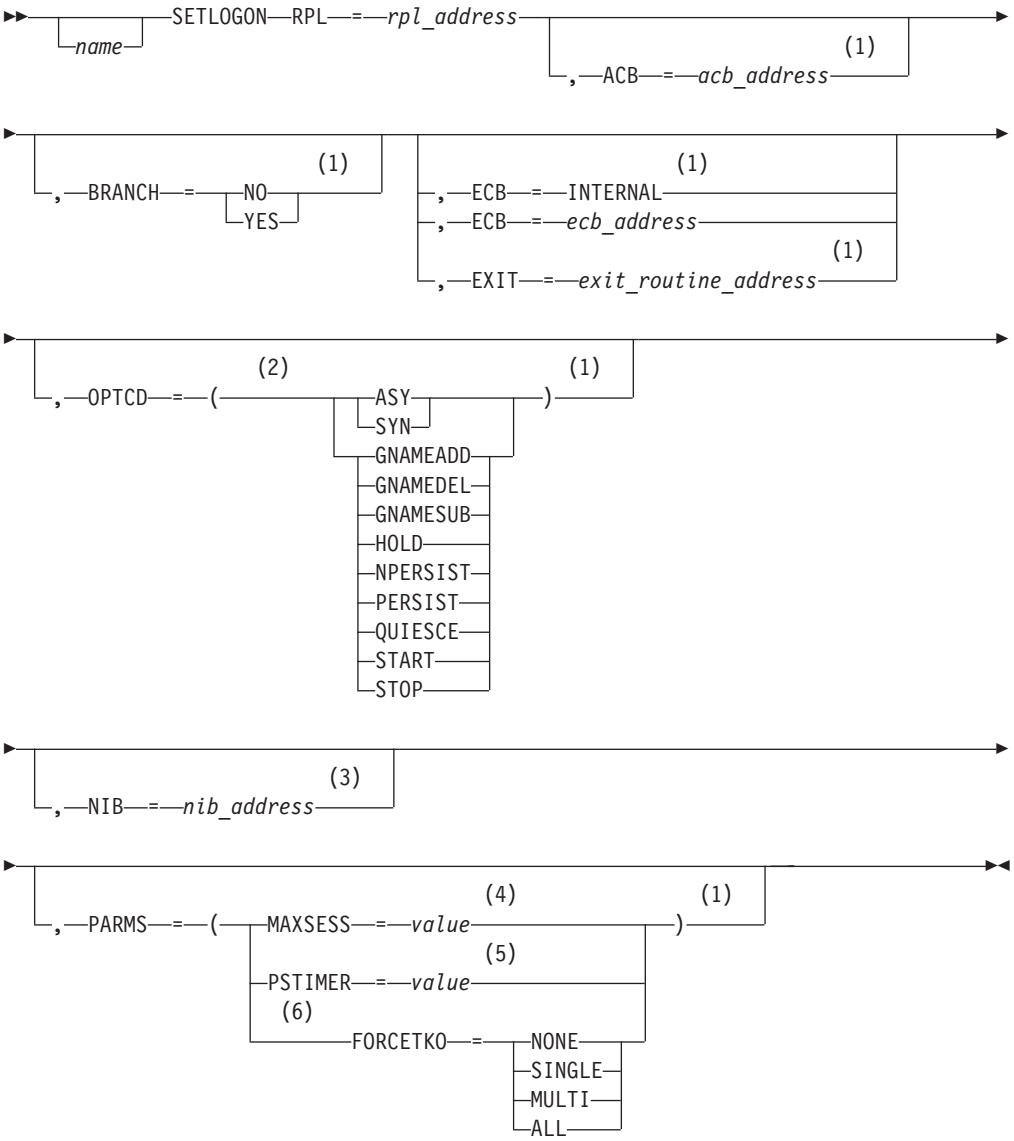
An application can be a subordinate of another application that is using a generic name. In this case, the subordinate application uses OPTCD=GNAMESUB to have its sessions included in the other application's session count for workload balancing purposes. When GNAMESUB is used, the NIB must indicate the application network name (NAME parameter) of the generic resource application to which this application is subordinate and the generic resource name (GNAME parameter).

An application that is capable of persistence uses SETLOGON OPTCD=PERSIST to enable persistent LU-LU session support. The application can also indicate support of receipt of various forced takeover requests by coding the FORCETKO operand to indicate the required support level on the SETLOGON OPTCD=PERSIST statement. This same application uses SETLOGON OPTCD=NPERSIST to disable persistence, but this SETLOGON OPTCD does not affect the setting of whether the application supports receipt of a forced takeover.

Before issuing the SETLOGON macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from the SETLOGON macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

3 NIB is valid only when OPTCD=GNAMEADD, OPTCD=GNAMEDEL, or OPTCD=GNAMESUB is coded.

4 PARMS=(MAXSESS=*value*) is valid only when OPTCD=GNAMEADD is coded.

5 PARMS=(PSTIMER=*value*) is valid only when OPTCD=PERSIST is coded. PSTIMER may be specified with FORCETKO on the same SETLOGON OPTCD=PERSIST invocation.

6 PARMS=(FORCETKO=NONE|MULTI|SINGLE|ALL) is valid only when OPTCD=PERSIST is coded. FORCETKO can be specified with PSTIMER on the same SETLOGON OPTCD=PERSIST invocation.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing SETLOGON is to perform.

The following RPL operands apply to the SETLOGON macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing SETLOGON.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

See the RPL macroinstruction for more information.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) SETLOGON operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) SETLOGON operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

NIB=*nib_address*
Indicates the NIB that specifies the generic name for the application. For OPTCD=GNAMESUB, the NIB also contains the application program network name of the particular generic resource instance. NIB= is valid only with the use of OPTCD=GNAMEADD, OPTCD=GNAMEDEL, or OPTCD=GNAMESUB.

OPTCD=GNAMEADD
OPTCD=GNAMEDEL
OPTCD=GNAMESUB
OPTCD=HOLD
OPTCD=QUIESCE
OPTCD=START

OPTCD=STOP
OPTCD=NPERSIST
OPTCD=PERSIST
Indicates information about the application program's session-establishment capability. Refer to the introductory description of the SETLOGON macroinstruction for details.

OPTCD=SYN
OPTCD=ASY
If the SYN option code is set, control is returned to the application program when the SETLOGON operation has completed. If ASY option code is set, control is returned as soon as VTAM has accepted the request. After the SETLOGON operation has been completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the SETLOGON operation, you should not use the SYN option if suspending the SETLOGON-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

PARMS=(MAXSESS=*value*)
This code, used only with OPTCD=GNAMEADD, indicates the maximum number of sessions allowed for the application. The total number of sessions includes sessions from subordinate generic resources. See 545 for information about subordinate generic resources.

PARMS=(PSTIMER=*value*)
This code, used only with OPTCD=PERSIST, indicates how long an application program can remain pending recovery. The recovering application must successfully issue an OPEN ACB before the timer expires or the sessions will be terminated. The initial value is 0, which means the safety timer is not used. The maximum timeout interval is 86400 seconds (24 hours). A default value does not exist. If PSTIMER is not specified, the current value is not changed. If a value is specified, PSTIMER must be coded on the same macroinstruction as OPTCD=PERSIST.

PARMS=(FORCETKO)
Indicates whether the application accepts some form of persistent sessions takeover requests when this application image is not pending some form of recovery currently. This parameter is valid only if OPTCD=PERSIST is also coded. There is no default for the FORCETKO operand, although an application will, unless otherwise indicated, support receipt of SNPS takeover requests only. If FORCETKO is not specified on the SETLOGON OPTCD=PERSIST, then the current setting for the application is unchanged.

PARMS=(FORCETKO=ALL)
The application accepts either SNPS or MNPS forced takeover requests when the application does not require some form of recovery processing.

PARMS=(FORCETKO=MULTI)
The application accepts only MNPS forced takeover requests when the application does not require recovery processing. SNPS forced takeover requests will be rejected.

PARMS=(FORCETKO=NONE)
The application does not accept SNPS or MNPS forced takeover requests.

PARMS=(FORCETKO=SINGLE)
The application accepts only SNPS forced takeover requests when the application does not require some form of SNPS recovery processing. MNPS forced takeover requests are rejected. An application, unless it specifies otherwise using SETLOGON OPTCD=PERSIST, is assumed to support this level of function.

Examples

```
         OPEN  ACB1
ADD      SETLOGON OPTCD=GNAMEADD,NIB=NIBGRSC,ACB=ACB1
:
:
BEGIN    SETLOGON RPL=RPL1,ACB=ACB1,OPTCD=START
:
:
HOLDEXITS SETLOGON RPL=RPL1,ACB=ACB1,OPTCD=HOLD
:
:
RESUME1  SETLOGON RPL=RPL1,ACB=ACB1,OPTCD=START
:
:
DELETE   SETLOGON OPTCD=GNAMEDEL,NIB=NIBGRSC,ACB=ACB1
:
:
TOOMANY  SETLOGON RPL=RPL1,ACB=ACB1,OPTCD=STOP
:
:
ADD2     SETLOGON OPTCD=GNAMEADD,NIB=NIBGRSC,ACB=ACB1
:
:
RESUME2  SETLOGON RPL=RPL1,ACB=ACB1,OPTCD=START
:
:
NOMORE   SETLOGON RPL=RPL1,ACB=ACB1,OPTCD=QUIESCE
:
:
ACB1     ACB   APPLID=APPLNAME,MACRF=LOGON
APPLNAME DC    05
         DC    CL5'STOCK'
:
:
NIBGRSC  NIB   GNAME=JOHNDOE
```

ADD identifies the application program with the application network name STOCK as a generic resource member using the generic resource name JOHNDOE. A generic resource member must issue OPTCD=GNAMEADD prior to establishing any LU-LU sessions.

Before BEGIN is executed, the application program's LOGON exit routine cannot be scheduled. Once BEGIN has completed, however, STOCK's LOGON exit routine is scheduled as each CINIT is received. STOCK might then also enter into a session as the secondary logical unit by issuing a REQSESS macroinstruction.

HOLDEXITS causes all CINIT and BIND requests to be queued and prevents the scheduling of the LOGON and SCIP exits for session setup requests.

RESUME1 causes VTAM to schedule the LOGON exit for each queued CINIT request, and schedules the SCIP exit for each queued BIND request.

DELETE removes STOCK as a generic resource member. After DELETE is completed, VTAM stops using STOCK to resolve session initiations from LUs that specify the generic resource name JOHNDOE. LUs can continue establishing sessions using the name STOCK. Additionally, any LU currently in session with STOCK can continue establishing parallel sessions using the generic resource name JOHNDOE.

TOOMANY causes VTAM to flag the application program as temporarily unwilling to accept CINITs. It does not prevent CINITs from being sent to STOCK

and causing STOCK's LOGON exit routine to be scheduled. If an application program that wants to initiate a session with the STOCK application program first issues INQUIRE OPTCD=APPSTAT, it receives feedback information indicating that session-initiation requests should not be issued for STOCK.

ADD2 specifies that VTAM use STOCK as a generic resource member of the generic resource name JOHNDOE. LUs can continue establishing sessions using the name STOCK. After ADD2 is completed, VTAM can use STOCK to resolve session initiations from LUs that specify the generic resource name, JOHNDOE.

RESUME2 reverses the effect of TOOMANY; application programs issuing INQUIRE OPTCD=APPSTAT receive feedback information indicating that session-initiation requests are being accepted (the same feedback information that results if INQUIRE is issued after BEGIN but before TOOMANY).

NOMORE prevents further sessions from being successfully initiated with STOCK. An INQUIRE issued by another application program would indicate this, and any attempt to initiate a session with STOCK would fail. If any CINITs are queued at STOCK when QUIESCE is issued, they remain queued and can be processed normally.

Completion information

A SETLOGON operation is successfully completed when the SSCP has recorded the application program's altered session-establishment capability. If ECB posting is used, the SCIP and LOGON exit routines can be scheduled before the application program recognizes that the processing of the SETLOGON has been completed successfully.

After the SETLOGON operation is completed, the following RPL fields are set:

- The value 21 (decimal) is set in the REQ field, indicating a SETLOGON request.
- If OPTCD=QUIESCE, the number of CINIT requests queued for the application program is set in the RECLEN field.
- If this is the first SETLOGON OPTCD=START request issued after opening the ACB, and a temporary storage shortage occurs during an attempt to schedule the LOGON exit routine, the SETLOGON fails (RTNCD,FDB2)=(X'08,'X'00'). The LOGON exit routine is successfully scheduled for all CINITs prior to the storage shortage. Any remaining automatic logons are not processed by additional SETLOGON requests. The application program can request that the VTAM operator issue a VARY command for each of the logons.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651.
- If the macroinstruction returns an error code, the SSENSEI, SSENSMI, and USENSEI fields can be set indicating system-sense information, system-sense modifier, and user-sense information. See Appendix B, “Return codes and sense fields for RPL-based macroinstructions,” on page 651 for more information about these fields.

Registers 0 and 15 are also set as indicated in Chapter 9, “Handling errors and special conditions,” on page 277.

SHOWCB—Extract the contents of control block fields

Purpose

SHOWCB extracts the contents of one or more ACB, EXLST, RPL, or NIB fields and places them into an area designated by the application program.

Usage

The SHOWCB user specifies the address of a control block and the names of the fields whose contents are to be extracted. In general, the field names are the same as the keywords of the ACB, EXLST, RPL, and NIB macroinstructions. Not all such field names are supported by SHOWCB. All of the fields applicable for SHOWCB are shown in Table 94 on page 553 at the end of the SHOWCB macroinstruction description. See Appendix J, "Summary of operand specifications," on page 857, for a list and explanation of the valid formats in which the SHOWCB operands can be specified.

Control block fields that can be operated on by SHOWCB are not limited, however, to fields that can be set by the application program in the ACB, EXLST, RPL, and NIB macroinstructions. Several additional fields whose contents are set only by VTAM can also be displayed with SHOWCB.

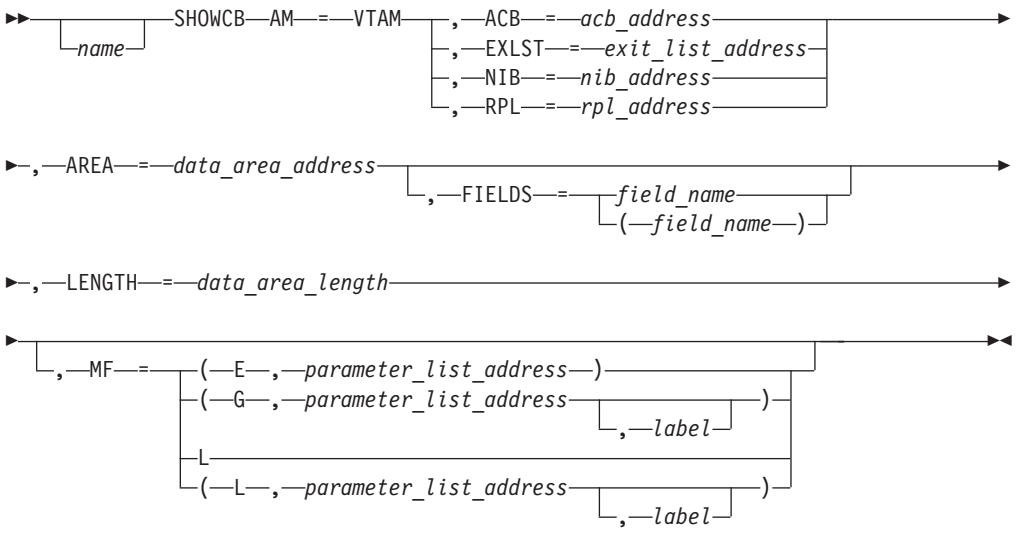
The user of SHOWCB must use the AREA and LENGTH operands to indicate the location and length of the area where the fields are placed. The contents of each field are placed there contiguously, in the order indicated by the FIELDS operand. If the area is too short to hold all of the fields, SHOWCB does not modify the area but returns error codes in register 0 and 15. Table 94 on page 553 shows the required lengths for all the control block fields that can be displayed with SHOWCB.

Note: The FDBK2 parameter on the SHOWCB macroinstruction represents the RPLFDB2 field.

List, generate, and execute forms of the SHOWCB macroinstruction are available; they are designated by the MF operand. See "Optional and required operands for the alternative forms of SHOWCB" on page 870 for more information on this macroinstruction.

The SHOWCB macroinstruction can be issued by an application program running in either 24- or 31-bit addressing mode. To use 31-bit addressing, the application program must use the VTAM mapping macroinstructions as well as GETMAIN and FREEMAIN.

Syntax



Input parameters

ACB=*acb_address*
EXLST=*exit_list_address*
NIB=*nib_address*
RPL=*rpl_address*

Indicates the type and location of the control block whose fields are to be extracted. One of these operands must be specified unless a control block length (and only the length) is being extracted. That is, if FIELDS=ACBLEN, FIELDS=EXLLEN, FIELDS=RPLLEN, or FIELDS=NIBLEN is specified, no specific control block need be specified.

AM=VTAM

Identifies the macroinstruction as capable of manipulating a VTAM control block. This operand is required.

AREA=*data_area_address*

Indicates the location of the storage area in the application program where the contents of the control block field or fields are to be placed. This work area must begin on a fullword boundary.

FIELDS=*field_name*

Indicates the control block field or fields whose contents are to be extracted.

For *field name*, code one of the field names that appear in the first column of the table that appears at the end of this macroinstruction description (Table 94 on page 553). Most of these field names correspond to keywords of the ACB, EXLST, RPL, and NIB macroinstructions. Only those fields associated with one control block (those for the control block whose address is supplied in the first operand) can be specified.

LENGTH=*data_area_length*

Indicates the length (in bytes) of the storage area designated by the AREA operand.

If this length is insufficient, SHOWCB returns a value of 4 in register 15 (unsuccessful completion) and a value of 9 in register 0 (insufficient length). The required length for each field is shown in the second column of Table 94 on page 553.

MF=E, G, or L

Indicates that an execute, generate, or list form of SHOWCB is to be used. Omitting this operand causes the standard form of SHOWCB to be used. See Appendix K, "Forms of the manipulative macroinstruction," on page 865, for a description of the execute, generate, and list forms of SHOWCB.

Examples

```
SHOW1    SHOWCB NIB=NIB1,FIELDS=NAME,AREA=NAME1,                    C
               LENGTH=8,AM=VTAM
```

SHOW1 extracts the contents of NIB1's NAME field and places it in NAME1.

```
SHOW2    SHOWCB RPL=RPL1,FIELDS=(FDBK,ARG,AREA,RECLEN),             C
               AREA=(3),LENGTH=16,AM=VTAM
```

SHOW2 extracts the contents of RPL1's FDBK, ARG, AREA, and RECLEN fields and places them (in that order) in a storage area. The address of this storage area must be in register 3 when SHOW2 is executed. Note that LENGTH indicates a storage area length great enough to accommodate all four fields.

Completion information

After SHOWCB processing is completed, VTAM sets register 15 to indicate successful or unsuccessful completion. If the operation is completed successfully, register 15 is set to 0, and register 0 contains the total number of bytes that SHOWCB extracted and placed in the work area. If the operation completes unsuccessfully, register 15 is set to either 04 or 08. If register 15 is set to 04 or 0C, register 0 is also set, indicating the specific nature of the error (see Appendix I, "Return codes for manipulative macroinstructions," on page 855, for additional information).

Control block fields applicable for SHOWCB

The field names shown in the first column of Table 94 are the values that can be supplied for the FIELDS operand of the SHOWCB macroinstruction. The lengths shown in the second column are the number of bytes of storage that must be reserved for each field; the sum of all the fields to be displayed by SHOWCB should be the value for the LENGTH operand.

Table 94. Control block fields that can be tested with SHOWCB

ACB fields

Field name	Length (bytes)	Description
APPLID	4	Address of application program's symbolic name
PASSWD	4	Address of password
EXLST	4	Address of exit list
ACBLEN	4	Length of ACB, in bytes (2 bytes, right-adjusted)
ERROR	4	OPEN and CLOSE completion code (1 byte, right-adjusted)

EXLST fields

Field name	Length (bytes)	Description

Table 94. Control block fields that can be tested with SHOWCB (continued)

ACB fields

LERAD	4	Address of exit routine
SYNAD	4	Address of exit routine
DFASY	4	Address of exit routine
RESP	4	Address of exit routine
SCIP	4	Address of exit routine
TPEND	4	Address of exit routine
RELREQ	4	Address of exit routine
LOGON	4	Address of exit routine
LOSTERM	4	Address of exit routine
NSEXIT	4	Address of exit routine
EXLLEN	4	Length of exit list, in bytes (2 bytes, right-adjusted)

RPL fields

Field name	Length (bytes)	Description
ACB	4	Address of ACB
NIB	4	Address of NIB
ARG	4	CID of session
AREA	4	Address of I/O work area
AREALEN	4	Length of AREA work area, in bytes
RECLEN	4	Length of data in AREA work area, in bytes
AAREA	4	Address of alternate I/O work area
AAREALN	4	Length of AAREA work area, in bytes
ARECLEN	4	Length of data in AAREA work area, in bytes
ECB	4	ECB or address of ECB
EXIT	4	Address of RPL exit routine
RTNCD	4	Recovery action return code (1 byte, right-adjusted)
FDBK2	4	Specific error return code (1 byte, right-adjusted)
FDBK	4	Additional status information (1 byte, right-adjusted)
USER	4	The data originally placed in an NIB's USERFLD
REQ	4	Request type code (1 byte, right-adjusted)
RPLLEN	4	Length of RPL, in bytes (1 byte, right-adjusted)
SEQNO	4	Sequence number (2 bytes, right-adjusted)
SSENSMO	4	Outbound system-sense modifier value (1 byte, right-adjusted)
USENSEO	4	Outbound user-sense value (2 bytes, right-adjusted)
SSENSMI	4	Inbound system-sense modifier value (1 byte, right-adjusted)
USENSEI	4	Inbound user-sense value (2 bytes, right-adjusted)

Table 94. Control block fields that can be tested with SHOWCB (continued)

ACB fields

IBSQVAL	4	Inbound sequence number for STSN request (2 bytes, right-adjusted)
OBSQVAL	4	Outbound sequence number for STSN request (2 bytes, right-adjusted)
SIGDATA	4	Information included with a signal request

NIB fields

Field name	Length (bytes)	Description
NAME	8	Symbolic name of logical unit
USERFLD	4	Arbitrary data associated with session
CID	4	Communication ID
NIBLEN	1	Length of NIB, in bytes
DEVCHAR	8	Device characteristics. See Appendix E, "Control block formats and DSECTs," on page 739.
EXLST	4	Address of exit list
RESPLIM	4	Maximum number of concurrent SEND (POST=RESP) macros
LOGMODE	8	Logon mode name
BNDAREA	4	Address of BIND area.

Note: Refer to the notes in Table 135 on page 857, Table 137 on page 858, and Table 138 on page 860 for information about fields that are not supported by the SHOWCB macroinstruction.

SIMLOGON—Initiate a session, application program acts as the PLU

Purpose

The SIMLOGON macroinstruction is used to initiate sessions in which the application program issuing SIMLOGON acts as the PLU. SIMLOGON processing sends an Initiate request to the SSCP. If successful, this causes a CINIT request to be sent from the SSCP to the application program, which might result in the LOGON exit routine being scheduled. See "LOGON exit routine" on page 101 for a description of how the application program receives the CINIT request.

Usage

The SIMLOGON macroinstruction uses the RPL to pass information to VTAM. The RPL points to a list of NIBs which contain the names of the logical units with which sessions should be initiated. The RPL indicates whether sessions should be initiated for as many NIBs in the list as possible, or just for one of them. The RPL also indicates whether the session-initiation request can be queued.

For a detailed description of the operation of the SIMLOGON macroinstruction, refer to "SIMLOGON macroinstruction" on page 87.

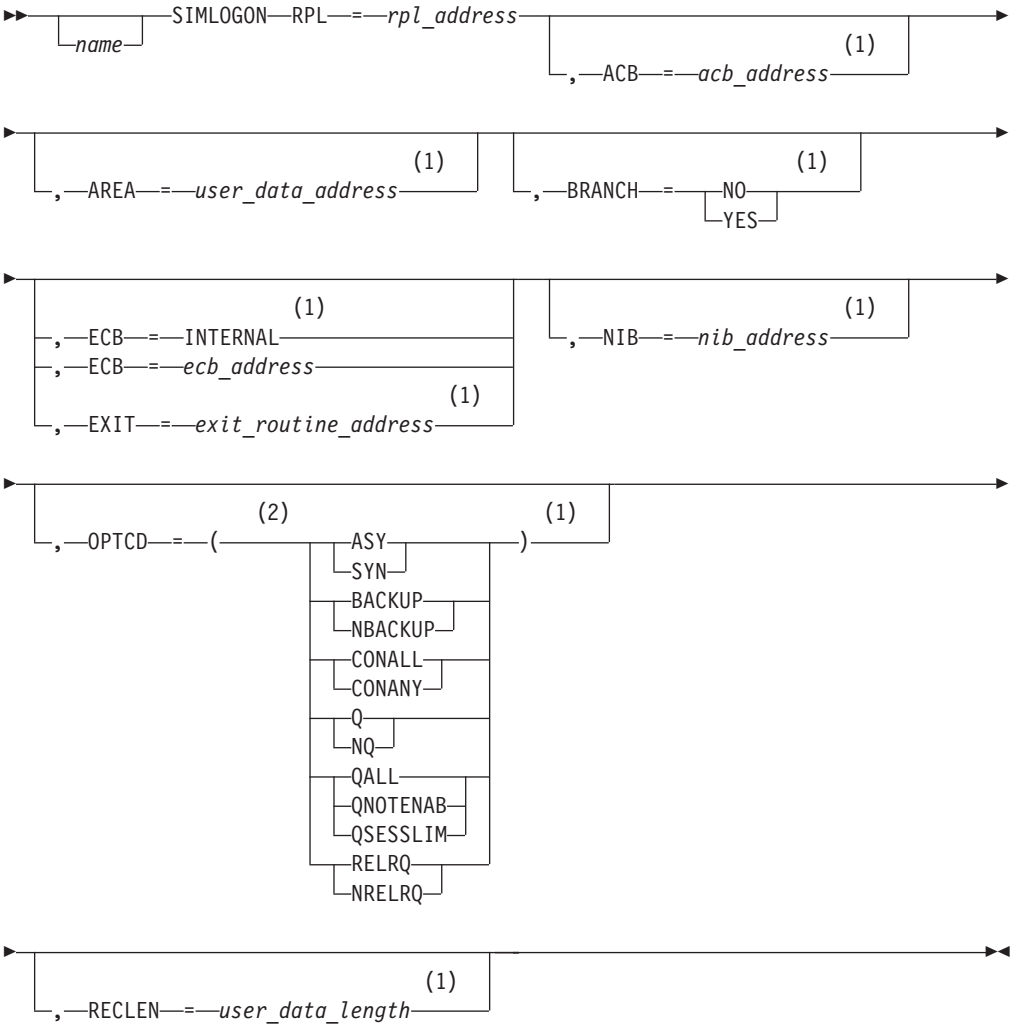
Before issuing the SIMLOGON macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from the SIMLOGON macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

The SIMLOGON macroinstruction is used by the application program to initiate primary and backup XRF sessions from the PLU.

If PARMS=(NQNAMES=YES) on the ACB macroinstruction, and the NIB is specified with a network identifier in the NIBNET field, the network identifier is used along with the LU name in NIBSYM to build a session initiation request.

Syntax



Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing SIMLOGON is to perform.

The following RPL operands apply to the SIMLOGON macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing SIMLOGON.

AREA=*user_data_address*
Indicates the location of the data that VTAM is to pass to the application program in the user data field of CINIT. The contents and format of the data are determined by the application program. They are equivalent to the user data field of an Initiate or a character-coded logon.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY) SIMLOGON operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) SIMLOGON operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter. For details about these operands, refer to the RPL macroinstruction description in this chapter.

NIB=*nib_address*
Indicates the NIB whose NAME field identifies the SLU for the session being initiated and whose LOGMODE field specifies the logon mode name to be used, and if NQNAMES=YES, whose NIBNET field contains the network where the SLU resides. If the NIB field contains the address of a list of NIBs,

that is LISTEND=NO (in the NIB pointed to), processing depends on the CONALL or CONANY option code. USERFLD can be used to specify a correlator to relate network services requests to this SIMLOGON.

OPTCD=BACKUP
OPTCD=NBACKUP

OPTCD=BACKUP is used to specify the initiation of a backup XRF session. Only one backup XRF session can be created per primary XRF session.

OPTCD=BACKUP is used only if either:

- A primary XRF session exists for the same SLU.

  This is the "normal" case, when the active XRF application creates the primary XRF session and then signals the alternate XRF application to start a backup XRF session.

- A backup XRF session exists for the same SLU.

  This is the exceptional case, when the NCP takes down the primary XRF session, but does not terminate its session with the SLU, keeping the backup session up pending a switch request state. In this case, if the primary XRF session desires to "recover" itself, it must first restart itself as a backup by issuing a SIMLOGON OPTCD=BACKUP, and then issuing a SESSIONC CONTROL to become the primary XRF session. In this case, the original backup session remains intact throughout the primary session recovery.

OPTCD=NBACKUP is the default and results in a session initiation request that does not specify a backup XRF session.

OPTCD=CONANY
OPTCD=CONALL

When CONANY is set, a session is initiated for the first available logical unit in the NIB list. Control is passed to the application program's LOGON exit routine, if one exists, when the resulting CINIT has been generated. When CONALL is set, a session is initiated for each available logical unit in the NIB list. If there is only one NIB, the setting of this option code does not matter.

OPTCD=Q
OPTCD=NQ

If OPTCD=NQ is specified, then the Initiate requests sent to the SSCP indicate sessions might not be queued, and the Initiate succeeds only if the logical unit is immediately available. If OPTCD=Q is specified, then the Initiate requests sent to the SSCP indicate sessions can be queued if the logical unit is not enabled or is at its session limit. (OPTCD=QALL or OPTCD=QSESSLIM or OPTCD=QNOTENAB determines whether both or just one of these conditions causes the request to be queued.) In that case, the Initiate succeeds if the logical unit is active, connected, and not inhibited for sessions in which it acts as the SLU. (See "Defining LUs" on page 81 for the definitions of these states.) Once the logical unit becomes available, a pending-active session is created immediately.

OPTCD=QALL
OPTCD=QSESSLIM
OPTCD=QNOTENAB

This option is meaningful only if the Q option is set. IF QALL is set, the session can be queued if the logical unit is not enabled or is at its session limit. If QSESSLIM is set, the session can be queued only if the logical unit is at its session limit. If QNOTENAB is set, the session can be queued only if the logical unit is not enabled for a session in which it is the SLU.

OPTCD=RELRQ

OPTCD=NRELRQ
This option is meaningful only if the Q option code is set. If RELRQ is set and if the device-type logical unit with which the session is being requested is in session with another application program, VTAM invokes that application program's RELREQ exit routine. If NRELRQ is set, the other application program is not notified of your request for the logical unit. For further details, see "SIMLOGON macroinstruction" on page 87.

OPTCD=SYN
OPTCD=ASY
If the SYN option code is set, control is returned to the application program when the SIMLOGON operation has completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. Once the SIMLOGON operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field. Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the SIMLOGON operation, you should not use the SYN option if suspending the SIMLOGON-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

Note: When using SIMLOGON OPTCD=ASY, an application program task could be suspended if an RPL exit routine is not provided. See "Initializing a session" on page 44 for details.

RECLEN=*user_data_length*
Indicates how many bytes of user data are to be sent. The value in RECLEN must be decimal 255 or less. If no user data is to be sent, RECLEN should be set to 0 and the AREA field is ignored.

Examples

```
SIM1     SIMLOGON RPL=RPL1,ACB=ACB1,NIB=NIBLIST1,                    C
               AREA=LGNMSG,RECLEN=60,EXIT=CHECKPGM,                  C
               OPTCD=(ASY,CONALL,NRELRQ,Q)
         .
         .
         .
LGNMSG   DC    CL60'LOGON FROM NIBLIST1 STATION'
ACB1     ACB   MACRF=LOGON
NIBLIST1 NIB   NAME=STATIONA,LISTEND=NO,                             C
               LOGMODE=BATCH
         NIB   NAME=STATIONB,LISTEND=NO
         NIB   NAME=STATIONC,LISTEND=YES
```

SIM1 initiates sessions between ACB1 and all of the logical units represented in NIBLIST1. Each Initiate contains a 60-byte user data field taken from LGNMSG. The logon mode name for STATIONA is BATCH; STATIONB and STATIONC use a default logon mode name. The CONALL option code indicates that Initiate requests are to be generated for all the available logical units represented in the list. NRELRQ indicates that if any of the logical units of NIBLIST1 is in session with an ACB other than ACB1, that ACB's RELREQ exit routine is not to be scheduled. After the SIM1 operation is completed, control is transferred to CHECKPGM.

Completion information

The SIMLOGON operation is successfully completed when the response is received for the Initiate request sent to the SSCP. This response is sent when a queued or pending active session is created, or when it is recognized that a session cannot be initiated. The sending of the CINIT request (and, therefore, the scheduling of the LOGON exit routine) might not occur immediately after the SIMLOGON macroinstruction is posted. Also, if ECB posting is used, the LOGON exit routine or NSEXIT exit routine can be scheduled for the requested session before the application program recognizes that the SIMLOGON has completed successfully.

When the SIMLOGON operation is completed, the following RPL fields are set:

- The value 22 (decimal) is placed in the REQ field, indicating a SIMLOGON request.
- The RTNCD and FDB2 fields are set as indicated in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651.
- If the macroinstruction returns an error code, the SSENSEI, SSENSMI, and USENSEI fields can be set indicating system-sense information, system-sense modifier, and user-sense information. See Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651 for more information about these fields.
- If the RTNCD and FDB2 fields are zeroes, check NIBSLWRK in each NIB to determine which is successful.

OPTCD=Q is not allowed when OPTCD=BACKUP is specified, because session queueing is not supported for backup XRF sessions. An error is posted with return code (RTNCD,FDB2)= (14,7D) if OPTCD=(BACKUP,Q) is specified.

Registers 0 and 15 are also set as indicated in Chapter 9, "Handling errors and special conditions," on page 277.

TERMSESS—Request session termination, application program is SLU

Purpose

The TERMSESS macroinstruction is used to terminate sessions in which the application program is acting as the SLU.

Usage

If OPTCD=UNBIND is specified with the TERMSESS macroinstruction, then VTAM ends the session by sending an UNBIND to the PLU. If TERMSESS OPTCD=COND, OPTCD=UNCOND, or OPTCD=TERMQ is issued, a TERMINATE request results. The PLU APPL has its LOSTERM EXIT driven for termination of the session. The SLU APPL is notified that the session has ended when the SCIP or the NSEXIT exit routine is driven.

Note:

1. If you are using OPTCD=UNBIND with the TERMSESS macroinstruction, you must specify the CID address in either the ARG or NIB operand. The NIBCID field is only used if PARSESS=YES is coded on the APPL definition statement. If you use the NIB operand, NAME cannot be used.

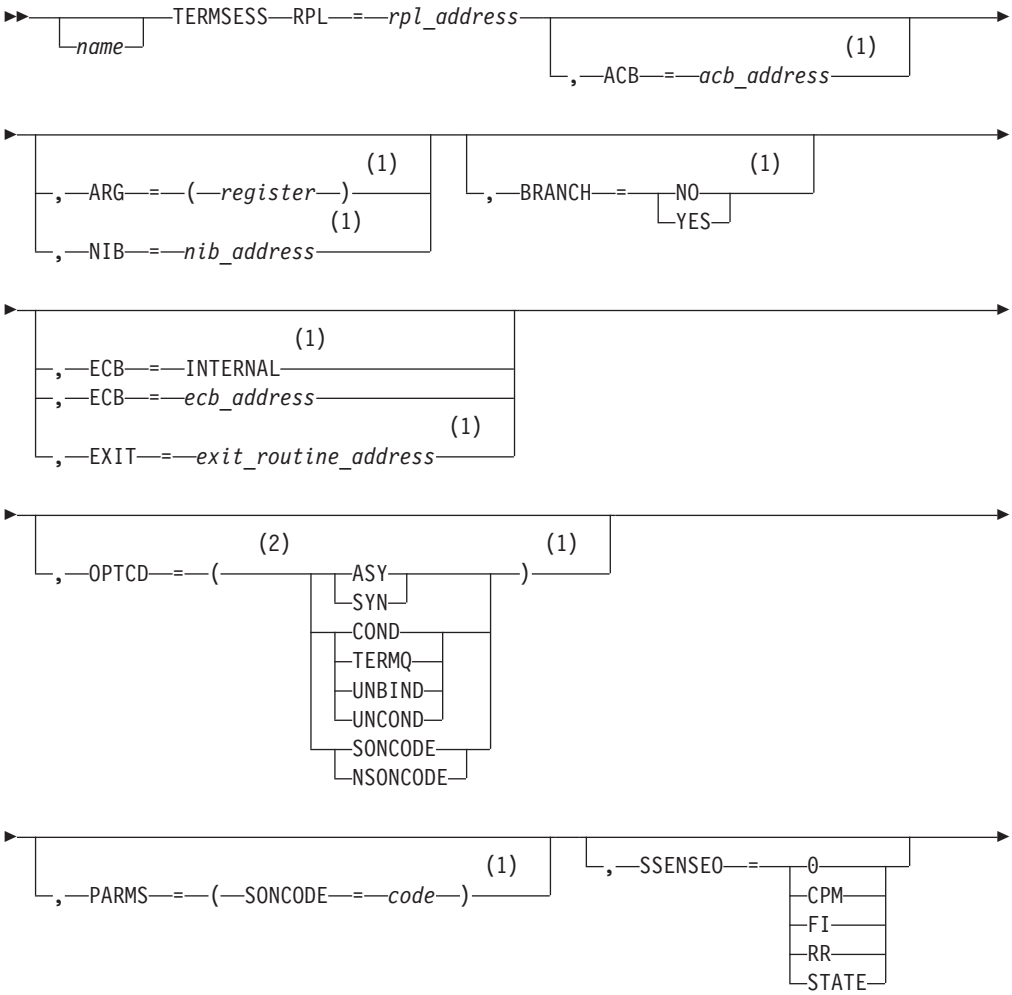
2. If the SLU application issues TERMSESS in lieu of SESSIONC to reject a pending active session (BIND received, BIND response not sent), then OPTCD=UNBIND or OPTCD=TERMQ must be used.
3. If OPTCD=TERMQ is specified, then a NIB must be supplied, which identifies the NAME of the session partner, which identifies the session to be terminated.

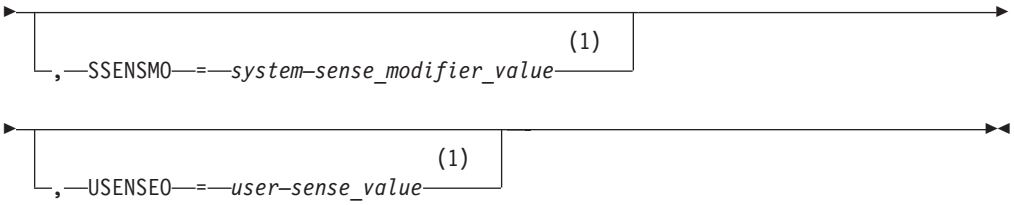
For a detailed description of the parameters and operation of the TERMSESS macroinstruction, refer to "TERMSESS macroinstruction" on page 99.

Before issuing the TERMSESS macroinstruction, the application program must set register 13 to the address of an 18-word save area. Refer to Appendix H, "Summary of register usage," on page 853, for information pertaining to the register contents upon return of control.

VTAM receives control from the TERMSESS macroinstruction in the addressing mode of the application program that issued the macroinstruction and returns control to the application program in that same mode.

Syntax





Notes:

1 Operand value can be placed in its RPL field either by specification on an RPL macroinstruction operand or by explicitly setting the field using the IFGRPL DSECT.

2 You can code more than one suboperand on OPTCD, but code no more than one from each group.

Input parameters

RPL=*rpl_address*
Indicates the RPL that specifies which kind of processing TERMSESS is to perform.

The following RPL operands apply to the TERMSESS macroinstruction:

ACB=*acb_address*
Indicates the ACB that identifies the application program issuing TERMSESS.

ARG=(register)
Indicates the register containing the CID of the session to be terminated.

Note:

1. The NIB and the ARG parameters occupy the same physical field (RPLARG) in the RPL. If the last macroinstruction operand used to set or modify this field was ARG=(*register*), or if the field has been left unchanged since VTAM inserted a CID into it, VTAM recognizes that this field contains a CID. If the last operand used to set or modify this field was NIB=*address*, VTAM recognizes that the field contains an NIB address.
2. If your application uses the RPL DSECT, IFGRPL, you must clear the RPLNIB bit if a CID is inserted into the RPLARG field.

BRANCH
For application programs running in supervisor state under a TCB, BRANCH indicates whether authorized path processing is to be used. See "Authorized path" on page 300.

BRANCH=YES
When the macroinstruction is issued, VTAM processes the macroinstruction using authorized path. For programs running under an SRB rather than under a TCB, the macroinstruction is processed in this manner automatically, regardless of the actual setting of the BRANCH field.

BRANCH=NO
When the macroinstruction is issued, VTAM does not process the macroinstruction using authorized path.

See "RPL—Create a request parameter list" on page 487 for more information.

ECB
Indicates that an ECB is posted when an asynchronous (OPTCD=ASY)

TERMSESS operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction.

ECB=*event_control_block_address*
Specifies that VTAM is to post an event control block (ECB). *Event_control_block_address* is the location of the ECB to be posted. The ECB can be any fullword of storage aligned on a fullword boundary.

ECB=INTERNAL
Specifies that VTAM is to post an internal ECB.

EXIT=*exit_routine_address*
Indicates the address of an RPL exit routine that is scheduled when an asynchronous (OPTCD=ASY) TERMSESS operation is posted as being complete. You cannot specify both ECB and EXIT on a single macroinstruction. For details about the EXIT operand, refer to the RPL macroinstruction description in this chapter.

NIB=*nib_address*
Indicates the NIB whose NAME field identifies the sessions to be terminated. If OPTCD=TERMQ is specified, a NIB must be supplied that identifies the NAME of the session to be terminated, and if NQNAMES=YES a NIB must be supplied whose NIBNET field contains the network identifier where the logical unit resides.

Note: If your application uses the RPL DSECT, IFGRPL, you must set the RPLNIB bit if an NIB address is inserted into the RPLARG field.

OPTCD=COND
OPTCD=UNCOND
OPTCD=UNBIND
OPTCD=TERMQ
If OPTCD=COND is set, the logical unit acting as the primary end of the session can take any action it desires, including issuing a CLSDST macroinstruction, if and when it wants to terminate the session.

Normally, if OPTCD=UNCOND is set, the primary logical unit automatically terminates the session and sends an UNBIND request to the SLU. However, in some network outage situations, the PLU cannot be ordered to send an UNBIND; the SLU is then sent a CLEANUP by its SSCP. This in turn causes the SLU to terminate the session by sending an UNBIND to the PLU.

If OPTCD=UNBIND, an UNBIND request is sent from the SLU to the PLU to terminate the session; control returns to the application program when the response to the UNBIND request is received.

If PARMS=(NQNAMES=YES) on the ACB macroinstruction, and the NIB is specified with a network identifier in the NIBNET field, the network identifier is used along with the LU name in NIBSYM to determine the target of the UNBIND.

If OPTCD=TERMQ is coded, VTAM will terminate only queued or pending active sessions (active sessions will not be terminated). When TERMQ is specified, the session partner name must be used via the NIB. CID cannot be used for this option. VTAM will send a TERMINATE for this option.

OPTCD=NSONCODE
OPTCD=SONCODE
If OPTCD=NSONCODE is coded, VTAM uses an UNBIND SON code of 01. If OPTCD=SONCODE is coded, VTAM uses the SON code specified in the RPL with the PARMS=(SONCODE=code) operand.

OPTCD=SYN
OPTCD=ASY
If the SYN option code is set, control is returned to the application program when the TERMSESS operation has been completed. If the ASY option code is set, control is returned as soon as VTAM has accepted the request. Once the TERMSESS operation has completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field.

Refer to the RPL macroinstruction description in this chapter for details about OPTCD=SYN or OPTCD=ASY.

Because it might take VTAM a relatively long time to complete the TERMSESS operation, you should not use the SYN option if suspending the TERMSESS-issuing task or SRB for this time is undesirable. Use the ASY option code, instead.

PARMS=(SONCODE=code)
VTAM uses the 1-byte UNBIND type code on an UNBIND RU. See the description of the UNBIND RU in *SNA Formats* for definitions of the UNBIND type codes (SON codes). VTAM does not validate the code specified in this parameter.

If PARMS=(SONCODE=FE) is specified, system-sense and user-sense codes are set with the existing SSENSEO, SSENSMO, and USENSEO RPL fields.

SSENSEO
This field is set by VTAM for a Logical Unit Status (LUSTAT) request and informs the logical unit of the type of error that caused the exception condition. These error types are described in Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651. SSENSEO=0 is the default.

This field can also provide application-specified sense values for negative responses to CINIT or UNBIND. Refer to the sections on the CLSDST or SEND macroinstructions in this chapter for additional information.

SSENSMO=*system-sense_modifier_value*
The value set in this field is used in conjunction with the SSENSEO setting to describe the specific type of error that caused the exception condition. The meanings assigned to the SSENSMO values are described in detail in *SNA Formats* If this operand is omitted, the SSENSMO field defaults to 0.

This field can also be used to provide application-specified sense values for negative responses to CINIT or UNBIND. Refer to the SEND macroinstruction in this chapter.

Specify any decimal integer 0–255 inclusive, or specify a 1-byte hexadecimal constant.

USENSEO=*user-sense_value*
This field is set by VTAM for a Logical Unit Status (LUSTAT) request. In most instances, the user-sense field is user-defined and can be used to inform the logical unit that an exception condition is being indicated for an application-program-related error that is not an SNA-defined error, or it can be used to further modify the SNA-defined system-sense and system-sense modifier values. See Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651, for more information. If this operand is omitted, the USENSEO field defaults to 0.

This field can also be used to provide application-specified sense values for negatives response to CINIT or UNBIND. Refer to the SEND macroinstruction in this chapter.

Specify any decimal integer 0–65535 inclusive, or specify a 2-byte hexadecimal or character constant.

Examples

```
TERMRTN  TERMSESS RPL=RPL1,NIB=NIB2,OPTCD=(ASY,COND),                C
               EXIT=TERMFINS
⋮
NIB2     NIB   NAME=PRIAPPL
```

If PRIAPPL is a VTAM application program, TERMRTN causes the LOSTERM exit routine of PRIAPPL to be scheduled with a reason code of 20 (decimal). PRIAPPL might ignore the request or issue a CLSDST macroinstruction when it wants to terminate the session with the application program that issued the TERMSESS macroinstruction. Upon completion of the TERMSESS macroinstruction, the RPL exit routine, TERMFINS, is scheduled.

Completion information

A TERMSESS operation successfully completes when either:

- The SSCP responds to the Terminate request generated by TERMSESS for TERMSESS OPTCD=COND or UNCOND or TERMQ. This can be before or after the UNBIND for the sessions is received in the SCIP exit routine.
- The current session terminates for TERMSESS OPTCD=UNBIND.

  After the TERMSESS macroinstruction completes, VTAM sets the following RPL fields:

- The value 44 (decimal) in the REQ field, indicating a TERMSESS request.
- If the macroinstruction returns an error code, the SSENSEI, SSENSMI, and USENSEI fields can provide system-sense information, system-sense modifier information, and user-sense information. See Appendix E, “Control block formats and DSECTs,” on page 739 for more information about these fields.
- The RTNCD and FDB2 fields are set as indicated in Appendix E, “Control block formats and DSECTs,” on page 739.

Registers 0 and 15 are also set as indicated in Chapter 9, “Handling errors and special conditions,” on page 277.

If the TERMSESS is issued for a session that has been terminated, a return code (RTNCD,FDB2)=(0C,0B) can be posted.

TESTCB—Test the contents of a control block field

Purpose

TESTCB compares the contents of a specified ACB, RPL, EXLST, or NIB field with a value supplied with the macroinstruction and sets the PSW condition code accordingly.

Usage

The user of the TESTCB macroinstruction indicates a particular control block, identifies a single field within that control block, and supplies the value against which the contents of that field are to be tested. Table 95 on page 568 lists the control block fields that can be tested.

Note: The FDBK2 parameter on the TESTCB macroinstruction represents the RPLFDB2 field.

The operands for testing control block fields are used in much the same way as operands for modifying or setting control block fields in macroinstructions like MODCB or GENCB. For example, RECLEN=200 in a MODCB macroinstruction places the value 200 in the RECLEN field of an RPL; if RECLEN=200 is specified in a TESTCB macroinstruction, the contents of the RECLEN field are compared with the value 200. See Appendix J, "Summary of operand specifications," on page 857, for a list and explanation of the various formats in which the TESTCB operands can be coded.

The test performed by TESTCB is a logical comparison between the field's actual contents and the specified value. The condition code indicates a high, equal, or low result (with the actual contents considered as A in the A:B comparison). The TESTCB macroinstruction can be followed by any branching assembler instructions that are valid following a compare instruction.

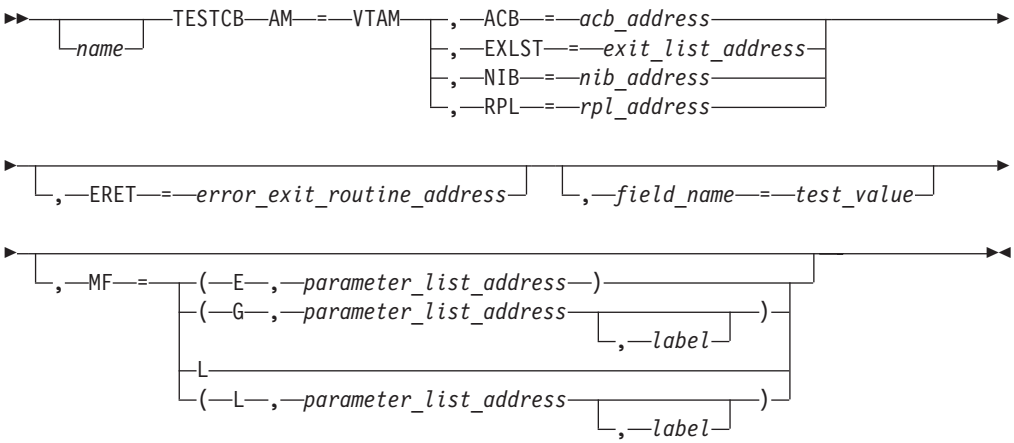
TESTCB can be used to test most control block fields whose contents can be set by the application program, as well as some of the control block fields whose contents are set by VTAM. The explanation of the *field name* operand indicates the fields that can be tested.

With the ERET operand of the TESTCB macroinstruction, the application program can supply the address of an error-handling routine. This routine is invoked if some error condition prevents the test from being performed correctly.

List, generate, and execute forms of TESTCB are available; they are designated by the MF operand. (See Appendix K, "Forms of the manipulative macroinstruction," on page 865, for more information.)

The TESTCB macroinstruction can be issued by an application program running in either 24- or 31-bit addressing mode. To use 31-bit addressing, the application program must use the VTAM mapping macroinstructions as well as GETMAIN and FREEMAIN.

Syntax



Input parameters

ACB=*acb_address*
EXLST=*exit_list_address*
NIB=*nib_address*
RPL=*rpl_address*

Indicates the type and location of the control block whose field is to be tested.

This operand is normally required but can be omitted if a control block length is being tested. (To test the control block lengths, specify ACBLEN, EXLLEN, RPLLEN, or NIBLEN for the TESTCB macroinstruction.) Because every control block of a given type is the same length, you do not have to indicate which ACB, EXLST, RPL, or NIB you are testing.

AM=VTAM

Identifies this macroinstruction as a VTAM macroinstruction. This operand is required.

ERET=*error_exit_routine_address*

Indicates the location of a routine to be entered if TESTCB processing encounters a situation that prevents it from performing the test.

When the ERET routine receives control, register 15 contains a return code. If this return code indicates an error, register 0 contains an error code that indicates the nature of the error. These codes, summarized in the following, are described fully in Appendix I, "Return codes for manipulative macroinstructions," on page 855. Register 14 contains the address of the ERET exit routine and the remaining registers are unchanged.

Note: If this operand is omitted, the program instructions that follow the TESTCB macroinstruction should check register 15 to determine whether an error occurred (indicating that the PSW condition code is meaningless) or not. To make this check without disturbing the condition code, a branching table based on register 15 can be used.

field_name=test_value

Indicates a control block field and a value that its contents are to be tested against. For *field name*, code one of the field names that appear in the table at the end of this macroinstruction description (Table 95 on page 568).

The rules for coding *test value* are defined and summarized in Appendix J, "Summary of operand specifications," on page 857.

MF=E, G, or L
Indicates that an execute, generate, or list form of TESTCB is to be used. Omitting this operand causes the standard form of TESTCB to be used. See Appendix K, "Forms of the manipulative macroinstruction," on page 865, for a description of the execute, generate, and list forms of TESTCB.

Examples

```
TESTCB   ACB=ACB1,PASSWD=(6),AM=VTAM
TESTCB   EXLST=EXLST1,SYNAD=SYNADPGM,AM=VTAM
```

RPL option codes or NIB processing options (including combinations of them) can also be tested. The test results in an equal condition code if all of the specified options are present. The following example shows how to test for the presence of the SPEC and CS option codes of an RPL. The second example illustrates how to code a similar test for the CONFTXT processing option of an NIB.

```
TESTCB   RPL=RPL1,OPTCD=(SPEC,CS),AM=VTAM
TESTCB   NIB=NIB1,PROC=(CONFTXT),AM=VTAM
```

Completion information

After TESTCB processing is finished and control is either passed to the ERET error routine or returned to the next sequential instruction, register 15 indicates whether the test was completed successfully. If the test completed successfully, register 15 is set to X'00'; if it completed unsuccessfully, register 15 is set to either X'04' or X'08'.

If register 15 is set to X'04' or X'0C' register 0 is also set indicating the specific nature of the error (see Appendix I, "Return codes for manipulative macroinstructions," on page 855, for additional information).

Control block fields applicable for TESTCB

The field names shown in the first column of Table 95 are the values that can be coded for the *field name* operand of the TESTCB macroinstruction. The second column indicates the number of bytes that each field occupies. No lengths are shown for fields that can only be tested using fixed values (for example, MACRF=LOGON or CONTROL=QEC).

Table 95. Control block fields that can be tested with TESTCB

ACB fields

Field name	Length (bytes)	Description
APPLID	4	Address of application program's symbolic name
PASSWD	4	Address of password
EXLST	4	Address of exit list
ACBLEN	2	Length of ACB, in bytes
ERROR	1	OPEN and CLOSE completion codes
OFLAGS	—	ACB open-closed indicator (OFLAGS=OPEN)
MACRF	—	Logon request status (MACRF=LOGON or NOLOGON)

EXLST fields

Table 95. Control block fields that can be tested with TESTCB (continued)

ACB fields

Field name	Length (bytes)	Description
LERAD	4	Address of exit routine
SYNAD	4	Address of exit routine
DFASY	4	Address of exit routine
RESP	4	Address of exit routine
SCIP	4	Address of exit routine
TPEND	4	Address of exit routine
RELREQ	4	Address of exit routine
LOGON	4	Address of exit routine
LOSTERM	4	Address of exit routine
NSEXIT	4	Address of exit routine
EXLLEN	2	Length of exit list, in bytes

RPL fields

Field name	Length (bytes)	Description
ACB	4	Address of ACB
NIB	4	Address of NIB
ARG	4	CID of session
AREA	4	Address of I/O work area
AREALEN	4	Length of AREA work area, in bytes
RECLEN	4	Length of data in AREA work area, in bytes
AAREA	4	Address of alternate I/O work area
AAREALN	4	Length of AAREA work area, in bytes
ARECLEN	4	Length of data in AAREA work area, in bytes
ECB	4	ECB or address of ECB
EXIT	4	Address of RPL exit routine
RTNCD	1	Recovery action return code
FDBK2	1	Specific error return code
FDBK	1	Additional status information
DATAFLG	1	Alias for FDBK
IO	—	RPL internal ECB post bit on (IO=COMPLETE)
USER	4	USERFLD data
REQ	1	Request type code
RPLLEN	2	Length of RPL, in bytes
BRANCH	—	SRB indicator (BRANCH=YES or NO)
OPTCD	—	RPL option code
CRYPT	—	Enciphered data flag
SEQNO	4	Sequence number

Table 95. Control block fields that can be tested with TESTCB (continued)

ACB fields

SSENSEO	—	Outbound system-sense value
SSENSMO	1	Outbound system-sense modifier value
USENSEO	2	Outbound user-sense value
SSENSEI	—	Inbound system-sense value
SSENSMI	1	Inbound system-sense modifier value
USENSEI	2	Inbound user-sense value
IBSQAC	—	Inbound action code for STSN request
OBSQAC	—	Outbound action code for STSN request
IBSQVAL	4	Inbound sequence number for STSN request
OBSQVAL	4	Outbound sequence number for STSN request
POST	—	Scheduled or responded output (POST=SCHED or RESP)
RESPOND	—	Response indicator (RESPOND)
CONTROL	—	Control command (CONTROL=DATA) indicator
CHAIN	—	Chain indicator (CHAIN)
CHNGDIR	—	Change-direction indicator (CHNGDIR)
BRACKET	—	Bracket indicator (BRACKET)
RTYPE	—	Receive-type indicator (DFSYN, DFASY, RESP)
STYPE	—	Send-type indicator (STYPE=REQ or STYPE=RESP)
SIGDATA	4	Information included with a signal request
CODESEL	—	Type of character encoding (CODESEL=STANDARD or CODESEL=ALT)

NIB fields

Field name	Length (bytes)	Description
NAME	8	Symbolic name of logical unit
USERFLD	4	Arbitrary data associated with session
CID	4	Communication ID
NIBLEN	1	Length of NIB, in bytes
DEVCHAR	8	Device characteristics. See Appendix E, "Control block formats and DSECTs," on page 739.
EXLST	4	Address of exit list
RESPLIM	4	Maximum number of concurrent SEND (POST=RESP) macros
LOGMODE	8	Logon mode name
BNDAREA	4	Address of BIND area
MODE	—	Mode indicator (MODE=RECORD)
LISTEND	—	NIB list termination flag (LISTEND=YES or LISTEND=NO)
SDT	—	Start-data-traffic flag (SDT=APPL or SDT=SYSTEM)

Table 95. Control block fields that can be tested with TESTCB (continued)

ACB fields		
PROC	—	Processing option codes
COND	—	Session-established flag (CON=YES)
ENCR	—	Cryptographic level indicator (ENCR=REQD,ENCR=SEL or ENCR=NONE)

Note: Refer to the notes in Table 135 on page 857, Table 137 on page 858, and Table 138 on page 860 for information about fields that are not supported by the TESTCB macroinstruction.

Chapter 14. Logic of a simple application program

Figure 93 on page 574 shows the logic of a VTAM application program that receives a request for a session with a logical unit (LU), establishes the session, reads input from any session, processes the input, prepares a reply for output, and writes the output on the session.

Logic of Sample Program 1

Sample program 1 demonstrates use of:

- A LOGON exit routine and the acceptance of a request for a session.
- A request to receive input from any session. (It is assumed in this example that parallel sessions are not used; that is, the application program can have only a single session with each LU.)
- Synchronous communication requests.
- Continue-any and continue-specific modes.
- A SEND macroinstruction to send a response rather than a data request.
- A request to schedule output.
- A RESP exit routine to handle a response received from the LU each time the LU receives data.
- A TPEND exit routine.

For simplicity, error recovery routines and other special routines are omitted; these routines are discussed and shown in coded form in sample program 1. See Chapter 15, “Sample code of a simple application program,” on page 579 for a coded VTAM application program that is based on sample program 1's general logic.

Sample program 1 might be usable for an application program in which each transaction between the program and the LU consists of a short inquiry and a short reply. Because the program waits for the processing for one LU to be completed before issuing a request for input from any other LU in session with it, the program might not adequately serve a large number of LUs communicating with the program at the same time.

The following logic flow diagram assumes that either the user logs on from a terminal device, or the operator uses the VARY LOGON command.

© Copyright IBM Corp. 2000, 2013

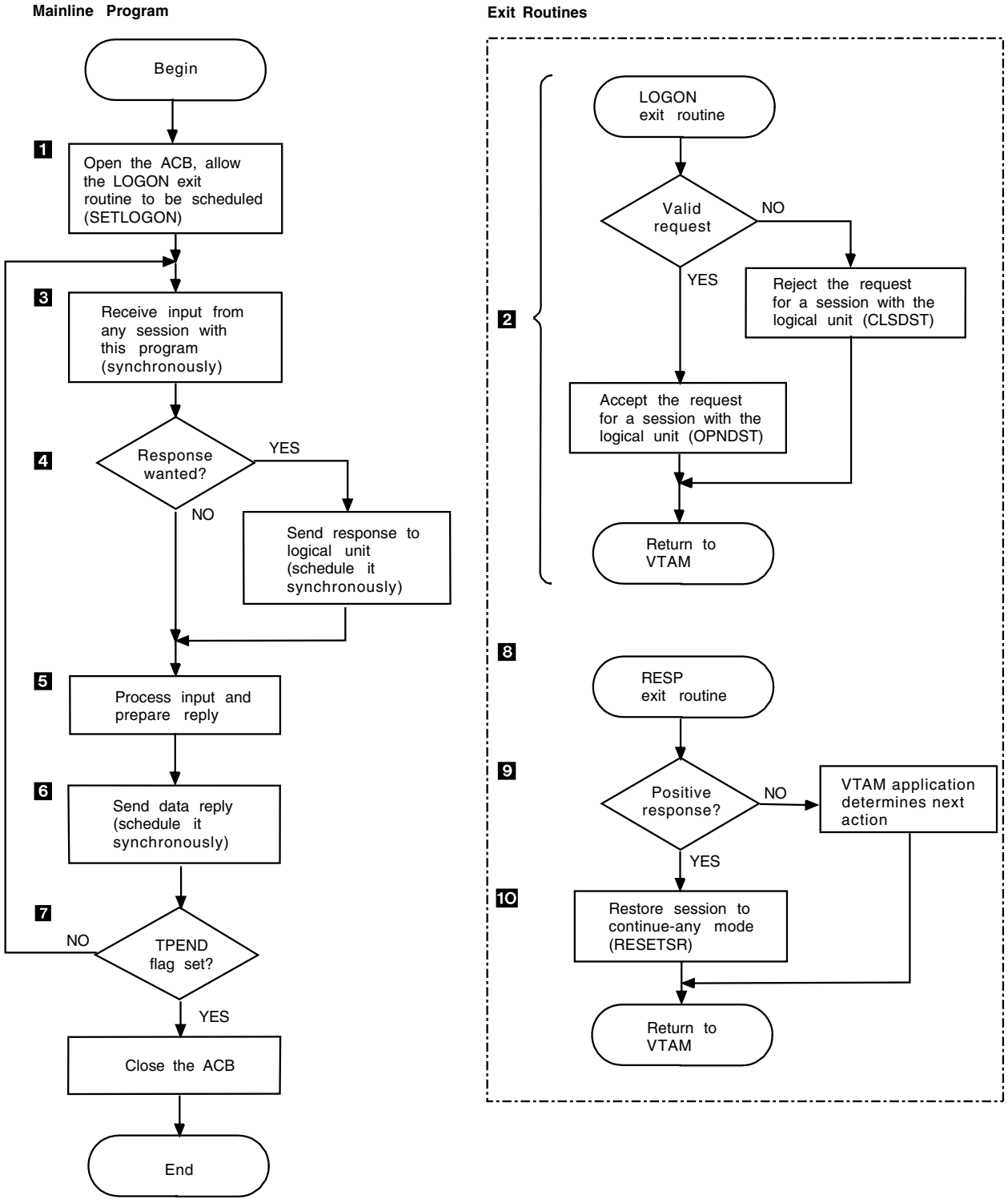


Figure 93. General logic of Sample Program 1

The following notes are keyed to Figure 93.

1 ACB1, defined in the program with the ACB macroinstruction, is opened with an OPEN macroinstruction. For example, this might be coded:

```
OPEN     ACB1
```

ACB1 contains:

```
ACB1     ACB   AM=VTAM,APPLID=APPL1,EXLST=EXLST1,                C
               MACRF=LOGON
```

APPL1 contains:

```
APPL1    DC    X'05'
         DC    C'PROG1'
```

PROG1 is the name of the APPL statement used to define the program during VTAM definition.

EXLST1 contains the names of VTAM exit routines used by the application program; some of these routines are shown in Figure 93 on page 574.

After the OPEN macroinstruction, a SETLOGON macroinstruction is used to tell VTAM that it can now schedule the LOGON exit routine for the program. The macroinstruction might be coded:

```
SETLOGON RPL=RPL1,OPTCD=START
```

2 The LOGON exit routine, whose address is specified in the LOGON operand of the EXLST macroinstruction (whose address is in turn specified in the ACB macroinstruction), is scheduled when VTAM receives a session-establishment request from or on behalf of an LU. Providing no other asynchronous exit routine is being executed, the LOGON exit routine is given control. When the LOGON exit routine is completed, either the next-scheduled exit routine receives control or the mainline program regains control at its next sequential instruction.

An INQUIRE macroinstruction can be used to obtain a user logon message when there is one. The logon message is the user data field of the request (for example, an Initiate from the LU) that originally requested the session:

```
INQ1     INQUIRE RPL=RPL1CONN,OPTCD=LOGONMSG,                   C
               ACB=ACB1,NIB=NIB1,AREA=LGNMSG,                   C
               AREALEN=100
```

By examining the logon message in LGNMSG, the exit routine determines whether the LU should be in session with the program. If so, an OPNDST macroinstruction like this can be issued:

```
OPNDST   RPL=RPL1CONN,OPTCD=(ACCEPT,SPEC),NIB=NIB1
```

An NIB and an RPL are required for the OPNDST request; if the OPNDST is synchronous, the same NIB and RPL can be reused by the OPNDST each time the LOGON exit routine is entered. If necessary, the suggested session parameter can be obtained by using an INQUIRE macroinstruction with OPTCD=SESSPARM specified; the NIB can then be properly initialized based on this information prior to issuing the OPNDST macroinstruction. (The MODCB macroinstruction or the ISTDNIB DSECT can also be used to initialize the NIB.)

If the LU is not to be allowed to use the program, a CLSDST macroinstruction must be issued to notify VTAM that the LU's request for a session is being rejected. PARMS=(SONCODE=*code*) can be coded on the CLSDST macroinstruction to provide a 1-byte code explaining why the session initiation request was rejected. Although not shown in the flowchart, it might be desirable to establish a session with the LU (using

OPNDST), write an appropriate request to it explaining to the LU why the session is being terminated, and then terminate the session.

3 In this example, the first request to receive input from any session with the program is issued in the mainline program. The request is synchronous; that is, the program indicates that it waits until input is received from one of the sessions. In making the request, the program identifies an RPL and an input area. The macroinstruction can be coded:

```
RECANY   RECEIVE RPL=RPL1,AREA=AREA1,AREALEN=100,                C
               RTYPE=DFSYN,OPTCD=(SYN,ANY,CS)
```

or it can be coded:

```
RECANY     RECEIVE   RPL=RPL1
```

where the RPL is coded:

```
RPL1     RPL   AM=VTAM,ACB=ACB1,AREA=AREA1,                      C
               AREALEN=100,RTYPE=DFSYN,                          C
               OPTCD=(SYN,ANY,CS)
```

The input request can be coded with some operands appearing in the RPL and others in the macroinstruction. If an operand that appears in the RPL is also coded in the macroinstruction, the value in the macroinstruction replaces the value in the RPL and is in effect not only for the current operation but for subsequent operations that use the RPL (unless changed by a manipulative macroinstruction, by assembler instructions using a DSECT, or by a subsequent request using the RPL).

RTYPE=DFSYN is specified to ensure that the receipt of a data request or of a normal-flow data-flow-control request completes the RECEIVE. (The receipt of a response in this sample program causes the RESP exit routine to be entered.)

OPTCD=CS is specified to ensure that the session with the LU whose input is read by the RECEIVE is put into continue-specific mode until its inquiry has been successfully answered. Thus, if the session is still in continue-specific mode when the next RECEIVE OPTCD=ANY is issued, input from that session does not satisfy the continue-any-mode request. The session is put back into continue-any-mode when a response has been received acknowledging that the reply to the inquiry arrived successfully; in this sample program, this is done in the RESP exit routine.

Assume that an LU sends in the first inquiry to the program. The synchronous RECEIVE is completed. If register 15 contains 0, the operation was successful. If register 15 contains some value other than 0, an error or special condition occurred. A LERAD or SYNAD exit routine in the program might have been entered, and might have returned a code in register 15 or register 0 that indicates further action for the program to take. Whether or not a LERAD or SYNAD was entered, information is available in various feedback fields of the RPL for analysis. One of the errors that can occur is that the request arrived as an exception request; this information is available as one of the feedback return codes in the RPL.

Assuming that the operation was successful, the identity of the session whose input was received is provided in the ARG field of the RPL. Data is located in AREA1, and a SHOWCB or TESTCB or assembler instructions can be used to determine its length (by examining the RECLEN field of the RPL).

4 The LU that sends the data can indicate that the program should issue a

response to verify that the input has been received. There might be some occasions when the LU wants a response, perhaps to verify that a data base update request has been received in order to free its buffers. On other occasions, such as an inquiry request, the LU might not want a response (or want a response only if an exception condition occurs); the answer to its inquiry will be forthcoming soon and will be implicit assurance that the inquiry arrived. The application program can examine the RESPOND field of the RPL to determine whether, and under what conditions, a response is required. If completion information following the RECEIVE indicates that the input was received normally and the RESPOND field indicates that a definite response is required, it is sent with a SEND macroinstruction that can be coded:

```
SENDRESP SEND  RPL=RPL1,STYPE=RESP,OPTCD=SYN
```

If completion information following the RECEIVE indicates that an exception request was received (in which case there will be no input data to process), and the RESPOND field indicates that a response is requested, it is sent with a SEND macroinstruction that might be coded:

```
SENDRESP SEND  RPL=RPL1,STYPE=RESP,OPTCD=SYN,                    C
               RESPOND=EX
```

If the LU wants a response only in the event of an exception, the RESPOND field will be set to EX after the RECEIVE is posted complete and does not have to be reset in the SEND. Before issuing this SEND, the application program places sense information defining the exception in the RPL.

The same RPL used for the RECEIVE request can be reused for the SEND. Because a response is being sent (STYPE=RESP), no data area or length is needed. For a response, POST=SCHED is assumed. The operation is specified to be synchronous. However, because it is only being scheduled, the operation will usually take a relatively short time. As soon as the operation has been scheduled and the SEND is completed, the RPL can be reused.

5 The inquiry is analyzed and a reply is prepared by a processing routine. Disk I/O might be required. If so, the program waits until the reply is ready.

6 The reply is then sent with a SEND that requests the transmission of data (STYPE=REQ). In this sample program, the same area used to receive input is used for output. The macroinstruction can be coded:

```
SENDDATA SEND  RPL=RPL1,STYPE=REQ,RESPOND=(NEX,FME),             C
               OPTCD=SYN,POST=SCHED
```

Because the RPL control block currently contains the value AREA=AREA1, this operand need not be specified in the SEND macroinstruction. So that the application program can determine whether the LU receives the data successfully, a response from the LU is requested (RESPOND=(NEX,FME)). The macroinstruction requests is completed synchronously (OPTCD=SYN) as soon as the sending of the output has been scheduled. Completion of the reply to the inquiry is determined as a result of the program's receiving the definite response 1 in the RESP exit routine.

With the reply under way, a branch is made back to the RECEIVE so that input that might have been read into VTAM's buffers from another session (that is not in continue-specific mode) is read into the application program for processing.

7 If the VTAM network is being halted, the program's TPEND exit routine (not shown) is scheduled and entered by VTAM. This routine can indicate to the mainline program that the program is to terminate. The TPEND exit routine or the mainline program later can send final requests to LUs and do other close processing depending on whether the closedown is immediate or a routine end-of-day closedown. The mainline program, discovering the closedown requirement, must issue a CLOSE macroinstruction to disassociate the application program from the network; any sessions not yet terminated as a result of the CLOSE are terminated. The CLOSE macroinstruction must be issued in the mainline program and not in the TPEND exit routine. The program terminates by returning control to the operating system.

8 When a response to the data sent in step 6 is received by VTAM, the RESP exit routine is scheduled and entered. On entry, register 1 points to a parameter list that points to a read-only RPL in VTAM's storage, whose feedback fields can be examined to determine the kind of response received.

9 If a negative response was received, the program can determine from sense information in the RPL whether to retry the operation, terminate the session, or take some other action.

10 If a positive response was received, the session can be returned to continue-any mode so that the next RECEIVE for input from any session (at step 3) includes this session as one whose input can be read by the application program. This is done by issuing:

```
RESETSR  RPL=RPL1R,OPTCD=CA,RTYPE=DFSYN
```

The RESP exit routine must have its own RPL available. The identity of the session to be returned to continue-any mode must be put in the RPLARG field of the exit routine's RPL. The RESP exit routine then returns control to VTAM.

Chapter 15. Sample code of a simple application program

This chapter contains the assembler language instructions for a VTAM application program, SAMP1. The logic for this program is similar to the logic for sample program 1 discussed in Chapter 14, "Logic of a simple application program," on page 573. The program includes DSECTs for the ACB, EXLST, NIB, and RPL control blocks. You can obtain the same results using manipulative macroinstructions instead of DSECTS.

Note: The program is presented primarily to provide sets of integrated examples, showing how the VTAM macroinstructions are used in real coding. The program is not intended to be coded and used by an installation.

What SAMP1 does

The SAMP1 application communicates with one or more logical units (LUs) in a network. SAMP1 is activated by input from an LU. (The LU might have created the input request as the result of terminal operator input received by the LU.) Depending on the code at the beginning of each request, SAMP1 performs a simple action, such as sending the request back to the LU.

Although SAMP1 can communicate with a number of different LUs during its execution, it is synchronous in its operation; that is, a reply is sent to one LU before a request is accepted from another LU. The use of synchronous processing simplifies program design but makes LUs unnecessarily interdependent, particularly during the I/O associated with OPNDST and CLSDST.

A program that would be executed in an LU with which SAMP1 might communicate is not shown.

How SAMP1 relates to Sample Program 1 (Chapter 14)

SAMP1 is based on the general logic of Sample Program 1 described in Chapter 14, "Logic of a simple application program," on page 573.

However, the logic of SAMP1 differs slightly from the logic of Sample Program 1. The request to receive input from any LU is specified as asynchronous in SAMP1. This allows the TPEND ECB to be checked following each issuance of a RECEIVE macroinstruction and the program to be closed if the TPEND ECB has been posted. In the logic in Sample Program 1 in Chapter 14, "Logic of a simple application program," on page 573, there is no opportunity to notice that the TPEND ECB has been posted while waiting for input to arrive and the synchronously specified RECEIVE to complete. Although the RECEIVE in SAMP1 is specified as asynchronous, it is effectively synchronous as far as handling LU input and output is concerned; the multiple waits wait only for posting of the one RECEIVE ECB or the TPEND ECB. (A CHECK macroinstruction is required to test for successful completion of the RECEIVE and to free the RPL for reuse.)

The TPEND, LERAD, SYNAD, and LOSTERM exit routines are not shown in Chapter 14, "Logic of a simple application program," on page 573, in order to simplify the example in that chapter. They are included here because a complete program requires them. Although SAMP1's LOSTERM, LERAD, and SYNAD exit routines are not as complete in their ability to handle errors and special conditions

© Copyright IBM Corp. 2000, 2013

as some installations can require, they provide an idea of the linkage and processing instructions that a LOSTERM, LERAD, or SYNAD exit routine might contain.

Sample Program 1 in Chapter 14, “Logic of a simple application program,” on page 573, demonstrates general logic that a simple program might contain; no data or request interface is described. Because SAMP1 is a complete program, it requires a defined data interface.

This version of SAMP1 uses codes in binary rather than hexadecimal in order to simplify the LU-LU operator interface. This is implemented in SAMP1 by means of TM and BO instructions in place of CLI and BE instructions.

Data interface between SAMP1 and LUs

Input: SAMP1 expects to receive a data request in this format from any LU with which it has a session:

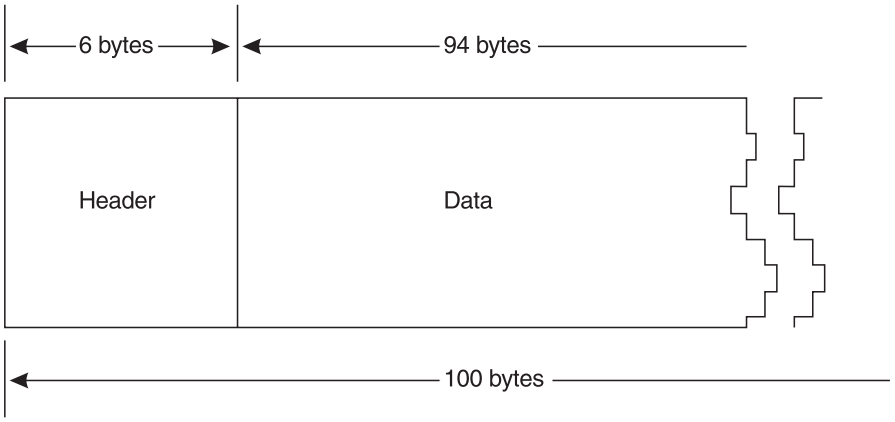


Figure 94. Data request format

The header is in this format:

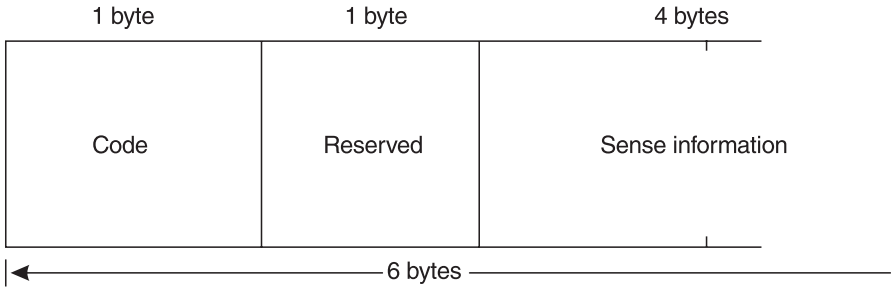


Figure 95. Header format

The sense information is present on input only if the code is B'xxxx xxx1'; otherwise, the sense information field is reserved. The sense information is in this format:

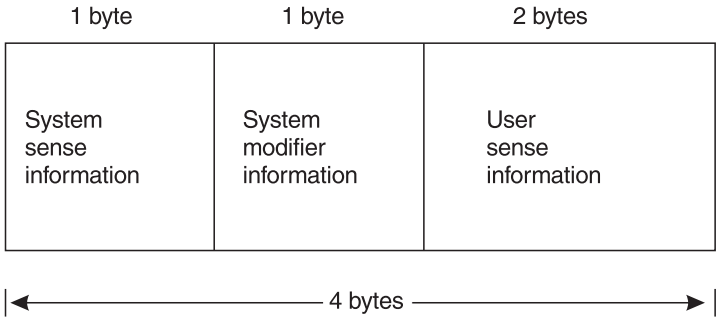


Figure 96. Sense information format

The sense information in the header allows the receiving of an exception request by SAMP1 to be simulated when the LU (or the terminal operator associated with the LU) requests it. The code contains B'xxxxxxx1' and the sense information in the header contains what VTAM would place in the SSENSEI, SSENSMI, and USENSEI fields of the RPL if a real exception request were to be received. To send the response to the simulated exception request, SAMP1 must move the sense information from the request header to the SSENSEO, SSENSMO, and USENSEO fields of the RPL that is used to send the response.

If the code contains B'xxxx x1x1' (that is, deliberate exception and echo), the LU not only expects a negative response to be returned, but wants a request sent that includes the sense information in the header of the request that just arrived (the simulated exception request). This causes the request sent now by SAMP1 to be interpreted by the LU as a simulated exception request, in turn causing the program in the LU to return a negative response back to SAMP1, driving the RESP exit routine (which calls the SYNAD exit routine).

In addition to the header information, up to 94 bytes of data can be received in the input request (excess data is truncated and ignored).

When SAMP1 receives a response to an output request it has sent, VTAM schedules its RESP exit routine. Whether the response is positive or negative, no input area is required; the information is present in fields of the read-only RPL.

Output: SAMP1 sends a request using the same area and format that is used for input. On output, it sets the code to:

B'1xxx x011'
Means this is the simulated exception request that you requested.

B'1000 0100'
Means forward this request to the terminal operator and then send it back to me. (This is used only from the LOGON exit routine and ensures that the mainline program RECEIVE is driven whether or not there is a terminal operator.)

No output area is required when sending either a positive or a negative response.

Notes on SAMP1

The following notes supplement the general logic description of Sample Program 1 in Chapter 14, "Logic of a simple application program," on page 573, and the prologue and comments provided with the source listing of "SAMP1" on page 585.

Mainline program

The request input area, AREA1, is initialized to asterisks to aid in debugging. The request length as well as the contents are evident if asterisks (rather than blanks) are printed when the request is presented to the terminal operator who is driving the program. (Other printable non-alphabetic characters could have been used as well.)

After the RECEIVE is completed, a CHECK is issued. If an error or special condition has occurred, the LERAD or SYNAD exit routine is entered. If the exit routine is able to recover successfully, it sets register 15 to 0; the mainline program is unaware that the exit routine was entered. If the exit routine is not able to recover successfully, it terminates the session with the LU or performs a SESSIONC CONTROL=CLEAR and a SESSIONC CONTROL=SDT, and sets register 15 to nonzero. The mainline program continues with other input, looping back to reissue the RECEIVE.

The RECEIVE can be completed by receipt of an expedited-flow (DFASY) or normal-flow (DFSYN) request. For example, receipt of Request Shutdown (or any other DFASY request) causes SAMP1 to issue CLSDST for that LU before reissuing the RECEIVE.

A check is made to see whether a simulated exception request has arrived; if so, a deliberate negative response must be returned. If a real exception request were to be received (requiring a negative response), the SYNAD exit routine would be scheduled as a result of the CHECK. In this case, the SYNAD exit routine would send a negative response and possibly use the SESSIONC macroinstruction to clear data traffic, reset the session to CA mode, and set an unsuccessful recovery indication in register 15 to allow the mainline program to reissue its RECEIVE.

Responses: SAMP1 is intended to handle all the valid combinations of no responses, exception response only, and definite responses. When instructed to echo the data (B'xxxx x1xx' in the first byte of the data header), it leaves the response types (RESPOND=*values*) unchanged, making it possible to force SAMP1 to send a request to the LU asking for no response or exception response only. In such a case, SAMP1 makes sure that the LU's session is in continue-any mode on completion of SEND rather than on receipt of a response by the RESP exit routine.

When SAMP1 asks for a definite response, it leaves the LU's session in continue-specific mode until that response is processed by the exit routine. The next request (or requests) might be queued in VTAM's pageable buffers, but is ignored until the response has been handled.

Function management and data-flow-control protocols: SAMP1 does not modify the settings related to FM protocols in the RPLs, except when requested to send an exception response. This means that fields like FMHDR, CHAIN, BRACKET, CODESEL, and CHNGDIR are echoed back to the LU, ignoring the fact that this can be a protocol violation. In addition, SAMP1 processes (for example, by echoing) each chain request separately.

Ending the program: Two ways exist to end the program. Either a request can be sent from the LU that says, "CLOSE ACB," or the VTAM operator can halt VTAM, causing the TPEND exit routine to be driven. In the first case, a TPEND flag is set; in the second, a TPEND ECB is posted. The mainline program checks both of these during each of its loops, branching to close the ACB if a close indication is found. Prior to closing the ACB, the TPEND flag is set to X'FF' to prevent any undesired activity while the CLOSE macroinstruction is being executed by VTAM. The TPEND flag is checked by the LOGON, RESP, and LOSTERM exit routines when each is entered to make sure the exit routine has not been scheduled while closing the ACB is in progress. If the flag is set, the exit routine returns immediately to VTAM.

LOGON exit routine

Notice in the manipulative macroinstruction version how the symbolic name of the LU for which a logon request has been received is placed in the NIB prior to establishing the session. NAME=(*,0(4)) is specified in the MODCB macroinstruction.

The last 4 bytes of the 8-byte symbolic name are placed in the USERFLD of the NIB (USERFLD=(*,4(4)) specified in the MODCB macroinstruction) for aid in debugging. These bytes identify the specific LU more easily than the CID (located in the RPL ARG field). This part of the symbolic name is available in the USER field of the RPL, except in the case of the RPL used for OPNDST.

An arbitrary validation of the LU is used: a check that the first 3 characters of the logon message are XYZ.

The IBM-supplied macroinstruction, ISTDNIB, is used to generate a dummy control section for the NIB. This enables SAMP1 to move the LU's symbolic name and associated user field into the NIB. A CSECT statement must follow the ISTDNIB statement in the constants area to allow SAMP1 CSECT to resume. None of the field names in SAMP1 begins with any of the reserved combinations (NIB, RPL, ACB, among others).

A conditional completion code of nonzero following INQUIRE at label INQUIRE, indicating a logon message that is too long, causes rejection of the session with the LU.

The session with the LU is established with an OPNDST. (Note that OPTCD=SYN is used on OPNDST. This, coupled with the fact that the macroinstruction is issued in an exit routine, means that all processing waits for the responses involved.) Then SEND is used to send a "logon accepted" message to the LU. This request is initialized to contain X'84' in the code byte of the header, meaning "Forward this

data to the terminal operator and then send it back to the VTAM application program". This version of SAMP1 includes the symbolic name of the LU in this request. This can help in cases where symbolic names are assigned dynamically.

RESP exit routine

Because the read-only RPL whose address is provided on entry to the RESP exit routine cannot be used to reset the LU to CA mode, an RPL (PRPLR) is reserved in the RESP exit routine for this purpose. The address of the ACB is obtained from the parameters passed on entry. Because only one asynchronous exit routine can be executing at a time and all RPL-based requests in SAMP1's exit routines are synchronous, one RPL could have been shared among all the exit routines.

In the event a negative response is received, the RESP exit routine sets up the correct linkage and calls the SYNAD exit routine directly. If the SYNAD exit routine is able to recover successfully (SAMP1 does not attempt to resend any requests), it sets register 15 to 0, and the RESP exit routine restores registers 1–12 and resets the session to CA mode. If it is not able to recover successfully, any necessary action, such as terminating the session with the LU, is taken in the SYNAD exit routine. The RESP exit routine need only return to VTAM.

LERAD and SYNAD exit routines

SAMP1's LERAD and SYNAD exit routines are re-enterable because the LERAD and SYNAD exit routines can be entered as the result of RPL-based requests issued by both the mainline program and exit routines other than LERAD and SYNAD. For example, the RECEIVE in the mainline program could cause the SYNAD exit routine to be entered. While the SYNAD is being executed as an extension of the mainline program, it could be interrupted and the LOGON exit routine given control. The OPNDST in the LOGON exit routine could cause the SYNAD exit routine to be re-entered, thus destroying any storage that might have values for the SYNAD exit routine as an extension of the mainline program. For this reason, SAMP1's LERAD and SYNAD exit routines obtain unique storage areas each time they are entered.

If the SYNAD exit routine is scheduled as the result of an RPL-based request that is issued within the SYNAD exit routine (that is, if the SYNAD exit routine is entered recursively), SAMP1 terminates with a dump. To determine recursion, SAMP1 uses the leftmost bit of register 1, which also contains the RPL address on entry to the SYNAD exit routine. Before issuing any RPL-based macroinstruction in the exit routine, this bit is set on; on entry, if this bit is found to have been set, the program terminates.

In the manipulative macroinstruction version, a series of TESTCB and MODCB macroinstructions is used to determine the SSENSEI setting to allow it to be set in the SSENSEO field before sending a negative response. This is required because the value for SSENSEO cannot be specified using register notation or in the FIELDS parameter of SHOWCB because it is a bit-encoded field.

The LERAD exit routine illustrates three cases that cause entry to LERAD but are not logic errors in the context of SAMP1, which is programmed to ignore them. These are cases that can arise due to the use of asynchronous exit routines.

LOSTERM exit routine

A LOSTERM exit routine determines why the session with the LU was lost and takes appropriate action. The SAMP1 LOSTERM exit routine does an immediate CLSDST for all LOSTERM reason codes.

SAMP1

```
*********************************************************************
*  SAMP1 (SAMPLE PROGRAM 1) IS DESIGNED TO BE
*  RELATIVELY EASY TO UNDERSTAND.  IT ILLUSTRATES:
*
*     0  OPENING AND CLOSING A PROGRAM, INCLUDING A TPEND
*        EXIT ROUTINE.
*
*     0  ESTABLISHING SESSIONS WITH LOGICAL UNITS IN A LOGON
*        EXIT ROUTINE.
*
*     0  RECEIVING AND SENDING REQUESTS AS SYNCHRONOUS OPERATIONS.
*        NOTE THAT THE RECEIVE USES OPTCD=ASY AND AN ECB FOLLOWED
*        BY A MULTIPLE WAIT, IN ORDER TO ALLOW WHAT TPEND DOES
*        (FOR EXAMPLE: SET A CLOSEDOWN SWITCH FOR MAINLINE) TO TAKE
*        EFFECT EVEN DURING A LULL IN COMMUNICATION ACTIVITY.
*
*     0  RESETTING A SESSION TO CONTINUE-ANY MODE.
*
*     0  RECEIVING AND SENDING RESPONSES TO REQUESTS.
*
*     0  RESP, TPEND, LOSTERM, LERAD, AND SYNAD EXIT ROUTINES.
*
*     0  THE LINKAGE BETWEEN THE MAINLINE PROGRAM, VTAM, AND
*        EXIT ROUTINES.
*
*  SAMP1 (SAMPLE PROGRAM 1) UTILIZES THE FOLLOWING MACROS TO REMOVE
*  OPERATING SYSTEM DEPENDENCIES
*
*     0  ENTER    - ENTRY POINT LINKAGE FOR ROUTINES
*     0  EXIT     - EXIT POINT LINKAGE FOR ROUTINES
*     0  GETSTOR  - OBTAIN STORAGE FROM OPERATING SYSTEM
*     0  FREESTOR - RETURN STORAGE TO OPERATING SYSTEM
*     0  CHKECB   - CHECKS ECB
*     0  ABTERM   - HANDLES ABNORMAL TERMINATION
*
*  SAMP1 (SAMPLE PROGRAM 1) IS ORGANIZED INTO:
*
*     0  MAINLINE PROGRAM.
*     0  LOGON EXIT ROUTINE.
*     0  RESP EXIT ROUTINE.
*     0  LERAD EXIT ROUTINE.
*     0  SYNAD EXIT ROUTINE.
*     0  TPEND EXIT ROUTINE.
*     0  LOSTERM EXIT ROUTINE.
*********************************************************************
         MACRO
&NAME    ENTER &SAVAREA=,&SAVE=,&TPEND=,&XTRA=0,&R14=,&OS=
*********************************************************************
*
*   MACRO:  ENTER - ROUTINE ENTRY LINKAGE
*
*   PARAMETERS:
*      SAVAREA:  GET - GET THE STORAGE FOR A SAVEAREA
*                NONE - NO SAVEAREA, DO NOT STORE REGISTERS
*                SAVEAREA - NAME OR ADDRESS OF SAVEAREA
*
*      SAVE:  YES - SAVE THE CALLERS REGS IN THE SAVEAREA
*                       POINTED TO BY R13
```

```
*                NO - DO NOT SAVE THE CALLERS REGS
*                MAINLINE - DO NOT SAFE CALLERS REGS AND
*                           DO NOT DROP REG 12
*
*      TPEND:  CHECK - CHECK THE TPENDFLG, AND IF IT HAS
*                      BEEN TURNED ON, THEN EXIT W/O ANY ACTION
*                NULL - DO NOT CHECK THE TPENDFLG
*
*      XTRA:  NUMBER - # OF EXTRA BYTES TO GET WHEN GETTING
*                      SAVEAREA
*
*      R14:  (REG) - REGISTER TO SAVE R14 IN FOR RETURN
*                ADDRESS - ADDRESS OF PLACE TO SAVE R14
*
*      OS:  MVS - GENERATE LINKAGE AND MACROS FOR MVS
*           VSE - GENERATE LINKAGE AND MACROS FOR VSE
*
*   RETURNS:
*      NORMAL:
*         R12:    BASE
*         R13:    SAVEAREA (IF REQUESTED)
*                 CALLERS REGS SAVED IN CALLERS SAVEAREA IF REQUESTED
*         R2,R3,R6,R15: MAY BE DESTROYED
*         R14(P)        P CONTAINS R14 IF REQUESTED
*
*      ABNORMAL:
*         ABEND:    CAN'T GET STORAGE, OR PROCESSING ERROR
*         IMMEDIATE EXIT: TPEND=CHECK IS SPECIFIED AND TRUE
*
*********************************************************************
         GBLC  &SYSTEM
         LCLC  &SAVED
         GBLA  &TPRET
         GBLC  &NEEDRET
&NEEDRET SETC  '
         AIF ('&SAVE' EQ 'MAINLINE').SKIPDRP
         DROP  R12      FROM PREVIOUS USING
.SKIPDRP ANOP
         AIF ('&OS' EQ '').SKIPSYS
&SYSTEM  SETC  '&OS'
.SKIPSYS ANOP
&NAME    DS    0H
.*
         AIF ('&SAVE' NE 'YES').SKIPSAV
         STM   R14,R12,12(R13)         SAVE CALLER'S REGS
&SAVED   SETC  'YES'
.SKIPSAV ANOP
         BALR  R12,0                   ESTABLISH BASE
         USING *,R12                   ESTABLISH ADDRESSABILITY
         AIF ('&TPEND' NE 'CHECK').SKIPCHK
         L     R6,=A(TPENDFLG)
         TM    0(R6),X'FF'             HAS TPEND BEEN DRIVEN?
&TPRET   SETA  &TPRET+1
&NEEDRET SETC  'YES'
         BO    RET&TPRET               YES, SO JUST EXIT
.SKIPCHK ANOP
         AIF ('&R14' EQ '').SKIPR14
         AIF ('&R14'(1,1) EQ '(').DOLOAD
         ST    R14,&R14                SAVE RETURN
ADDRESS
         AGO   .SKIPR14
.DOLOAD  ANOP
         LR    &R14,R14
.SKIPR14 ANOP
         AIF ('&SAVAREA' EQ 'NONE').SKIPMST
         AIF ('&SAVAREA' NE 'GET').SKIPGET
         LR    R2,R0                   SAVE R0
```

```
         LR    R3,R1                   SAVE R1
         GETSTOR 72+&XTRA              GET SAVEAREA
         LTR   R15,R15                 OK?
         BZ    ER&SYSNDX               YES, CONTINUE
         ABTERM 4
ER&SYSNDX DS 0H
         LR    R15,R1                  SETUP R15
         LR    R0,R2                   RESTORE R0
         LR    R1,R3                   RESTORE R1
         AGO   .SKIPLD
.SKIPGET ANOP
         AIF ('&SAVAREA' EQ '').SKIPLD
         L     R15,=A(&SAVAREA)        GET ADDRESS
OF OUR SAVEAREA
.SKIPLD ANOP
         AIF ('&SAVE' EQ 'MAINLINE').SKIPFOR
         ST    R13,4(R15)              SAVE BACKWARD
POINTER
.*
         AIF ('&SAVE' NE 'YES').SKIPFOR
         ST    R15,8(R13)              SAVE FORWARD
POINTER
.SKIPFOR ANOP
         LR    R13,R15                 SETUP SAVEAREA
POINTER
.SKIPMST ANOP
         MEND*****************************************************************
*
*   MACRO:  EXIT - ROUTINE EXIT LINKAGE
*
*   PARAMETERS:
*   SAVEAREA:  FREE - FREE THE STORAGE FOR A SAVEAREA
*
*   RESTORE:  YES - RESTORE THE CALLERS REGS
*              NO - DO NOT RESTORE THE CALLERS REGS
*
*   XTRA:  NUMBER - # OF EXTRA BYTES TO FREE WHEN FREEING
*                       SAVEAREA
*
*   R14:  (REG) - REGISTER THAT R14 WAS SAVED IN
*               ADDRESS - ADDRESS OF PLACE R14 WAS SAVED
*
*   RETURNS:
*   NORMAL:
*     R15:  RETURN CODE IF SPECIFIED
*     OTHER REGS RESTORED IF SPECIFIED
*
*   ABNORMAL
**     ABEND FOR FREEMAIN FAILURES
*
*****************************************************************
         MACRO
&NAME    EXIT  &RESTORE=,&R14=,&SAVAREA=,&RC=YES,&XTRA=0,&EOJ=NO
         GBLC  &SYSTEM
         GBLA  &TPRET
         GBLC  &NEEDRET
&NAME    DS    0H
         AIF ('&RESTORE' NE 'YES').SKIPRS1
         L     R2,4(R13)               GET BACKWARD
SAVEAREA
         AIF ('&RC' NE 'YES').SKIPRC1
         ST    R15,16(R2)              SAVE RETURN
CODE
         AGO   .SKIPRC1
.SKIPRS1 ANOP
         AIF (('&RC' NE 'YES') OR ('&RESTORE'
NE 'YES') ).SKIPRC1
```

```
         LR    R3,R15                  SAVE RETURN
CODE
.SKIPRC1 ANOP
         AIF ('&SAVAREA' NE 'FREE').SKIPFRE
         FREESTOR LEN=72+&XTRA,AREA=(R13)
.SKIPFRE ANOP
         AIF ('&RESTORE' NE 'YES').SKIPRS2
         LR    R13,R2                  SETUP OLD SAVEAREA
         LM    R14,R12,12(R13)         RESTORE ALL
REGS
         AGO   .SKIPRC2
.SKIPRS2 ANOP
         AIF (('&RC' NE 'YES') OR ('&RESTORE' NE 'YES') ).SKIPRC2
         LR    R15,R3
         AIF (('&SYSTEM' EQ 'VSE') AND ('&EOJ'
EQ 'YES')).DOEOJ
.SKIPRC2 ANOP
         AIF ('&R14' EQ '').DORET
         AIF ('&R14'(1,1) NE '(').DOLOAD
         LR    R14,&R14                GET RETURN
ADDRESS
         AGO   .DORET
.DOLOAD  ANOP
         L     R14,&R14                GET RETURN
ADDRESS
.DORET   ANOP
         BR    R14
         AGO   .TRYRET
.DOEOJ   ANOP
         EOJ
.TRYRET  ANOP
         AIF ('&NEEDRET' NE 'YES').SKIPRET
RET&TPRET DS 0H
         AIF ('&RESTORE' NE 'YES').SKIPRS3
         LM    R14,R12,12(R13)
.SKIPRS3 ANOP
         AIF (('&SYSTEM' EQ 'VSE') AND ('&EOJ'
EQ 'YES')).DOEOJ2
         BR    R14
         MEXIT
.DOEOJ2  ANOP
         EOJ
.SKIPRET ANOP         MEND
*********************************************************************
*
*   MACRO:  GETSTOR - GET STORAGE
*
*   PARAMETERS:
*      LEN:  NUMBER - # OF BYTES OF STORAGE TO GET
*
*   RETURNS:
*      IF LEN BYTES OF STORAGE ARE SUCCESSFULLY OBTAINED,
*      THEN R1 CONTAINS THE ADDRESS OF THE AREA AND R15
*      CONTAINS ZERO
*
*      IF THE STORAGE CANNOT BE OBTAINED THEN THE PROGRAM
*      IS ABNORMALLY TERMINATED
*
*
*********************************************************************
         MACRO
&NAME    GETSTOR &LEN
         GBLC  &SYSTEM
         AIF ('&SYSTEM' EQ 'VSE').GETVSE
         GETMAIN R,LV=&LEN
         MEXIT
.GETVSE  ANOP
```

```
         GETVIS ADDRESS=(R1),LENGTH=&LEN
         MEND*****************************************************************
*
*   MACRO:  FREESTOR - FREE STORAGE
*
*   PARAMETERS:
*      LEN:  NUMBER - # OF BYTES OF STORAGE TO FREE
*
*      AREA:  ADDR - ADDRESS OF AREA TO FREE
*
*   RETURNS:
*      IF LEN BYTES OF STORAGE ARE SUCCESSFULLY FREED, THEN
*      R15 CONTAINS ZERO
*
*      IF THE STORAGE CANNOT BE FREED THEN THE PROGRAM
*      IS ABNORMALLY TERMINATED
*
*****************************************************************
         MACRO
&NAME    FREESTOR &LEN=,&AREA=
         GBLC  &SYSTEM
         AIF ('&SYSTEM' EQ 'VSE').FREEVSE
         FREEMAIN R,LV=&LEN,A=&AREA
         MEXIT
.FREEVSE ANOP
         FREEVIS ADDRESS=&AREA,LENGTH=&LEN
         MEND*****************************************************************
*
*   MACRO:  CHKECB - CHECK ECB
*
*   PARAMETERS:
*      ECB:  SPECIFIES ECB TO CHECK
*
*   RETURNS:
*      TM CONDITION CODE IS SET DEPENDING ON ECB BEING POSTED
*
*****************************************************************
         MACRO
&NAME    CHKECB &ECB
         GBLC  &SYSTEM
         LCLA  &POST
         LCLA  &BYTE
         AIF('&SYSTEM' EQ 'VSE').CKVSE
&POST    SETA  X'40'
&BYTE    SETA  0
         AGO .CHECK
.CKVSE   ANOP
&POST    SETA  X'80'
&BYTE    SETA  2
.CHECK   ANOP
&NAME    TM    &ECB+&BYTE,&POST
         MEND
*****************************************************************
*
*   MACRO:  ABTERM - ABNORMAL TERMINATE
*
*   PARAMETERS:
*      1ST POSITIONAL PARAMETER IS THE ABEND CODE, DEFAULTS TO ZERO
*
*   RETURNS:
*      (IT DOESN'T)
*
*****************************************************************
         MACRO
&NAME    ABTERM
         GBLC  &SYSTEM
         LCLC  &CODE
```

```
         AIF (N'&SYSLIST EQ 1).SETCODE
&CODE    SETC  '0'
         AGO   .DOIT
.SETCODE ANOP
&CODE    SETC  '&SYSLIST(1)'
.DOIT    ANOP
         AIF ('&SYSTEM' EQ 'VSE').ABVSE
&NAME    ABEND &CODE,DUMP
         MEXIT
.ABVSE   ANOP
&NAME    LA R10,&CODE
         JDUMP
         MEND
*********************************************************************
*
*        NAME = MAINLINE PROGRAM
*
*   FUNCTION = OPENS THE ACB, ISSUES SETLOGON, RECEIVES INPUT FROM
*      ANY ESTABLISHED SESSION, SENDS A RESPONSE IF REQUESTED,
*      FORWARDS INPUT TO A PROCESSOR, SENDS A REPLY PREPARED BY THE
*      PROCESSOR, AND LOOPS BACK TO RECEIVE MORE INPUT AFTER SENDING
*      THE REPLY.  CLOSES THE ACB (CLOSES THE PROGRAM) IF A TPEND ECB
*      IS POSTED BY VTAM HALT OR IF 'CLOSE ACB' IS ENTERED AS A
*      A REQUEST.
*
*      NOTE: THE PROGRAM HANDLES ONE REQUEST OF AN INPUT CHAIN AT A
*      TIME.  BE CAREFUL WITH HDX-FF PROTOCOLS.
*
*   ENTRY POINT = SAMP1
*
*   INPUT = REQUESTS RECEIVED FROM SESSIONS ESTABLISHED WITH LOGICAL
*      UNITS; A POSTED TPEND ECB.  EACH REQUEST CONTAINS A 6-BYTE
*      HEADER DESCRIBING THE ACTION TO BE TAKEN.  (SEE EQUATES IN
*      MAINLINE PROGRAM CONSTANTS.)  NOTE THAT DFASY INPUT
*      CAUSES A CLSDST.  BRACKET PROTOCOL IS NOT SUPPORTED AND
*      CAUSES UNPREDICTABLE RESULTS.
*
*   OUTPUT = REQUESTS AND RESPONSES SENT TO LOGICAL UNITS AS A RESULT
*      OF INPUT REQUESTS.  PROGRAM TERMINATION AND A DUMP IF THE
*      PROGRAM CANNOT CONTINUE.
*
*   EXTERNAL REFERENCES = OPEN, SETLOGON, RECEIVE, CLSDST,
*      WAIT(M), CHECK, AND SEND.
*
*   EXIT, NORMAL = BR 14
*
*   EXIT, ABNORMAL = DUMP (DIRECTLY OR BY SYNAD OR LERAD).
*
*   ATTRIBUTES = NOT SERIALLY REUSABLE
*
*   REGS USED
*
*     3 = RETURN ADDRESS
*     4 = WORK REG
*     5 = A(PRPL)
*    12 = BASE REG
*    13 = A(SAVE0)
*
*********************************************************************
*********************************************************************
*
SAMP1    CSECT
         ENTER SAVE=MAINLINE,SAVAREA=SAVE0,OS=MVS
*********************************************************************
*                                                                   *
*                           OPEN THE ACB                            *
*                                                                   *
```

```
*********************************************************************
         GLBC  &SYSTEM
OPNACB   EQU   *
         SLR   R15,R15
         OPEN  PACB                    ASSOCIATE THE PROGRAM WITH VTAM
         LTR   R15,R15                 TEST FOR ERRORS
         BZ    OPENOK
*********************************************************************
* IT WOULD BE NORMAL HERE TO TEST FOR AN INVALID APPLID AND LET THE  *
* VTAM OPERATOR KNOW OF THE PROBLEM RATHER THAN CAUSING AN ABEND.    *
*********************************************************************
DUMP     ST    R1,R1CONTS              SAVE THE CONTENTS OF REG 1
         ABTERM
VERSION  DC    C'DATE OF LAST CHANGE 09/16/91'
OPENOK   LA    R5,PRPL                 SET UP BASE FOR RPL DSECT
         USING IFGRPL,R5
         SETLOGON RPL=PRPL,OPTCD=START ALLOW LOGON REQUESTS
         LTR   R15,R15                 TEST FOR ERRORS
         BNZ   DUMP
*********************************************************************
* SOLICIT INPUT FROM ANY LOGICAL UNIT.                               *
*********************************************************************
RECANY   MVI   AREA1,C'*'              SET ASTERISK IN 1ST BYTE OF
*                                      AREA1 (FOR DEBUGGING PURPOSES)
         MVC   AREA1+1(L'AREA1-1),AREA1  ROLL IT
         RECEIVE RPL=PRPL,AREA=AREA1,AREALEN=100,
               OPTCD=(ASY,ANY,CS),ECB=RCVECB,      RESP HANDLED BY EXIT
               RTYPE=(DFSYN,DFASY)
*********************************************************************
* THE PARAMETER OPTCD=CS IS USED ON THE RECEIVE-ANY TO FORCE A       *
* ROTATION OF THE SESSIONS WHICH COULD POSSIBLY SATISFY THE RECEIVE. *
* WITHOUT THIS PARAMETER, A BUSY SESSION COULD UNINTENTIONALLY LOCK  *
* OUT OTHER SESSIONS SENDING DATA.                                   *
*********************************************************************
         LTR   R15,R15                 TEST FOR ACCEPTANCE
         BNZ   DUMP                    DUMP IF NOT ACCEPTED
         CHKECB RCVECB
         BO    CHECK                   YES, BYPASS WAITM SVC
         AIF ('&SYSTEM' EQ 'VSE').WAITVSE
         WAIT  ECBLIST=ECBLST
         AGO   .COMCODE
.WAITVSE ANOP
         WAITM RCVECB,TPENDECB
.COMCODE ANOP
         CHKECB TPENDECB
         BO    RETURN1                 YES, GO TO CLOSE ACB
CHECK    EQU   *
         OI    RESETCAF,X'FF'          INIT RESETSR CA NEEDED FLAG
         CHECK RPL=PRPL                NO, CHECK COMPLETION OF RECEIVE
         LTR   R15,R15                 TEST FOR SUCCESSFUL COMPLETION
         BNZ   RECANY                  NO, CONTINUE WITH NEXT INPUT
*********************************************************************
* THE SYNAD EXIT ROUTINE WILL EITHER ISSUE A CLSDST TO TERMINATE THE *
* FAILING SESSION OR CLEAR THE EXCEPTION AND RESTORED THE SESSION TO *
* CONTINUE ANY MODE.                                                 *
*********************************************************************
         TM    RPLSRTYP,RPLDFASY       DFASY RECEIVED?
         BNO   TESTRRN                 NO
         CLSDST RPL=PRPL,OPTCD=SYN
*********************************************************************
* IGNORE THE POSSIBLE FAILURE OF THE CLSDST MACRO.  THE SYNAD/LERAD  *
* ROUTINES WILL HANDLE ANY ERRORS.                                   *
*********************************************************************
         B     CHCKTPND                ALLOW FOR PROGRAM CLOSEDOWN
TESTRRN  EQU   *
         TM    RPLVTFL2,RPLRRN         RRN RESPONSE WANTED
         BO    TESTEXCP
```

```
         TM    RPLVTFL2,RPLNFME        TEST FOR NO RESPONSE
         BO    PROCESS
TESTEXCP EQU   *
         TM    AREACODE,AEXCEPT        EXCEPTION RESPONSE WANTED?
*********************************************************************
* THE PRECEDING TEST WAS CHANGED FROM A 'CLI' INSTRUCTION TO A 'TM'  *
* INSTRUCTION TO PROCESS EBCDIC HEADERS.                             *
*********************************************************************
         BNO   RESPTEST                NO, CHECK FOR DEFINITE RESPONSE
         MVC   RPLSSEO,AREASENS        SET SYS SENSE OUTPUT
         MVC   RPLSSMO,AREASENS+1
         MVC   RPLUSNSO,AREASENS+2
         OI    RPLVTFL2,RPLEX
         NI    RPLOPT5,X'FF'-RPLDLGIN  SET OPTCD=CA FOR SENDD
         B     SENDRESP                SEND THE EXCEPTION RESPONSE
RESPTEST TM    RPLVTFL2,RPLEX
         BNO   SENDRESP                DEFINITE RESP SO LEAVE OPTCD=CS
         NI    RPLOPT5,X'FF'-RPLDLGIN  SET OPTCD=CA FOR SENDD
         B     PROCESS
SENDRESP EQU   *
         SEND RPL=PRPL,STYPE=RESP,OPTCD=(SYN,SPEC) SEND PREPARED RESP
         LTR   R15,R15                 TEST FOR SUCCESSFUL COMPLETION
         BNZ   DUMP                    DUMP IF SEND COULD NOT BE
*                                       SCHEDULED
         NI    RESETCAF,X'00'          TURN OFF RESETSR CA NEEDED FLAG
PROCESS  EQU   *
TEST1    EQU   *
CLOSETST CLC   AREADATA(9),=C'CLOSE ACB'   IS CLOSE ACB REQUESTED
         BNE   TEST2
         OI    TPENDFLG,X'80'          SET ON TPEND FLAG TO CLOSE ACB
TEST2    EQU   *
         OI    AREACODE,ASAD
         TM    AREACODE,AECHOB         IS TERMINAL ECHO NEEDED?
         BNO   TEST3
         OI    AREACODE,AECHO          YES, TURN ON ECHO FLAG
         NI    AREACODE,X'FF'-AECHOB   TURN OFF PLEASE ECHO BACK FLAG
SENDDATA EQU   *
SENDD    SEND  RPL=PRPL,STYPE=REQ,     NOTE THAT THIS
               OPTCD=SYN,POST=SCHED    LEAVES CA,CS AS SET ABOVE.
         LTR   R15,R15                 TEST FOR SUCCESSFUL COMPLETION
         BNZ   DUMP                    DUMP IF SEND COULD NOT BE SCHED
         NI    RESETCAF,X'00'          TURN OFF RESETSR CA FLAG
TEST3    EQU   *
         CLI   RESETCAF,X'FF'          IS A RESETSR CA NEEDED
         BNE   TEST4
         RESETSR RPL=PRPL,OPTCD=(CA,SYN)
         LTR   R15,R15
         BNZ   DUMP
TEST4    EQU   *
CHCKTPND CLI   TPENDFLG,X'80'          SEE IF TPEND IS SIGNALLED
         BNE   RECANY                  IF NOT, BRANCH BACK TO RECEIVE
RETURN1  MVI   TPENDFLG,X'FF'          SIGNAL CLOSE IN PROGRESS TO
*********************************************************************
*                                                                   *
*               CLOSE THE ACB AND EXIT THE PROGRAM                  *
*                                                                   *
*********************************************************************
         CLOSE  PACB                   CLOSE THE ACB
         EXIT  SAVAREA=SAVE0,EOJ=YES
*
*
*********************************************************************
*********************************************************************
*                                                                   *
*                       VARIABLE DECLARATIONS                       *
*                                                                   *
*********************************************************************
```

```
*
*********************************************************************
*                                                                   *
* ACB, RPL, AND EXLST                                               *
*                                                                   *
*********************************************************************
PACB     ACB   AM=VTAM,APPLID=APPL1,EXLST=EXLST1,MACRF=LOGON
EXLST1   EXLST AM=VTAM,LOGON=LOGON1,SYNAD=SYNAD1,LERAD=LERAD1,
                RESP=RESP1,TPEND=TPEND1,LOSTERM=LOSTERM1
PRPL     RPL   AM=VTAM,ACB=PACB
*********************************************************************
*                                                                   *
* CONSTANTS                                                         *
*                                                                   *
*********************************************************************
R1CONTS  DC    F'0'                    SAVE AREA FOR REG 1 IN DUMP
ECBLST   DC    A(RCVECB)
         DC    X'80'                   END OF ECB LIST MARKER
         DC    AL3(TPENDECB)
RCVECB   DC    F'0'                    ECB USED FOR RECANY
TPENDECB DC    F'0'                    ECB POSTED BY TPEND EXIT
TPENDFLG DC    X'00'                   SET BY MAINLINE TO FORCE CLOSE
RESETCAF DC    X'00'                   RESETSR CA NEEDED IF 00
SAVE0    DC    18F'0'                  SAVE AREA NEEDED FOR MAINLINE
*                                      PROGRAM
APPL1    DC    X'08'                   APPLID FOR ACB
         DC    CL8'PROG1'
AREAOFLO DC    C'*THIS SHOULD NOT BE DISPLAYED:  CHECK RECLEN'
*********************************************************************
*                                                                   *
* LOCAL STORAGE VARIABLES                                           *
*                                                                   *
*********************************************************************
         DS    0H
AREA1    DS    0CL100                  I/O DATA AREA
AREAHEAD DS    0CL6                    HEADER
AREACODE DS    XL1
RSV1     DS    XL1                     RESERVED
AREASENS DS    XL4                     SENSE FIELD WHEN AREACODE='01'
AREADATA DS    CL94                    DATA FIELD
*********************************************************************
*                                                                   *
* EQUATES FOR INPUT/OUTPUT                                          *
*                                                                   *
*********************************************************************
AEXCEPT  EQU   X'01'                   PLEASE RETURN AN EXCEPTION
*                                      RESPONSE AS SPECIFIED IN
*                                      AREASENS
AECHOB   EQU   X'04'                   PLEASE ECHO THIS BACK TO ME
ASAD     EQU   X'80'                   SEND THIS MESSAGE TO THE SCREEN
AECHO    EQU   X'02'                   THIS IS THE ECHO YOU REQUESTED
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
```

```
         LTORG
         EJECT
         IFGRPL AM=VTAM
         EJECT
         ISTUSFBC
         EJECT
         IFGACB AM=VTAM
         EJECT
         IFGEXLST AM=VTAM
*
*
*********************************************************************
*
*   NAME = LOGON EXIT ROUTINE
*
*   FUNCTION = ESTABLISH A SESSION AND SEND A 'LOGON ACCEPTED'
*      MESSAGE TO ANY LOGICAL UNIT THAT LOGS ON IF THE LOGON MESSAGE
*      STARTS WITH 'XYZ'; OTHERWISE, REJECT THE REQUEST FOR A SESSION.
*
*   ENTRY POINT = LOGON1
*
*   INPUT
*   REGISTERS
*      0    =   UNPREDICTABLE
*      1    =   POINTER TO A 4-WORD PARAMETER LIST
*      2-13 =   UNPREDICTABLE
*      14   =   ADDRESS TO RETURN CONTROL TO
*      15   =   ENTRY ADDRESS OF THIS ROUTINE
*   PARAMETER LIST - 6 WORDS
*      1    =   ACB ADDRESS
*      2    =   POINTER TO SYMBOLIC NAME OF LOGICAL UNIT
*      3    =   ZEROS
*      4    =   LENGTH OF LOGON MESSAGE
*      5    =   ADDRESS OF READ-ONLY RPL
*      6    =   CID OF PENDING ACTIVE SESSION
*
*   OUTPUT
*      A REQUEST TO VTAM TO ACCEPT OR REJECT THE SESSION; OR PROGRAM
*      TERMINATION AND A DUMP IF UNABLE TO CONTINUE.  IF SESSION IS
*      ESTABLISHED, A 'LOGON ACCEPTED' MESSAGE IS SENT TO THE LOGICAL
*      UNIT SPECIFYING EXCEPTION RESPONSE ONLY.
*
*   EXTERNAL REFERENCES = INQUIRE, OPNDST, CLSDST, SEND.
*
*
*   EXIT, NORMAL = BR 14
*
*   EXIT, ABNORMAL = DUMP
*
*   ATTRIBUTES = SERIALLY REUSABLE
*
*   REGS USED
*
*      3 =   RETURN ADDRESS
*      4 =   A(SYMBOLIC NAME OF LU)
*      5 =   A(PRPLCONN),IFGRPL
*      6 =   A(LOGON EXIT PARM LIST)
*      7 =   A(PNIB),ISTDNIB
*      8 =   LENGTH OF LOGON MESSAGE
*      9 =   ACB ADDRESS
*     12 =   BASE REG
*     13 =   A(SAVE2)
*
*********************************************************************
SAMP1    CSECT                          RESTART CSECT
LOGON1   ENTER SAVE=NO,SAVAREA=SAVE2,TPEND=CHECK,R14=SAVE1
*                                      LOGONS
```

```
         LR    R6,R1                   SAVE THE PARAMETER LIST ADDRESS
         L     R9,0(R6)                PICK UP ACB ADDRESS
         L     R4,4(R6)                POINT TO THE SYMBOLIC NAME OF
*                                      THE LOGICAL UNIT
         LA    R5,PRPLCONN             SET UP BASE FOR RPL DSECT
         LA    R7,PNIB                 LOAD BASE FOR NIB DSECT
         USING IFGRPL,R5
         USING ISTDNIB,R7
         MVC   NIBSYM,0(R4)
         MVC   NIBUSER,4(R4)
         MVC   NIBCID, 20(R6)          PUT CID INTO NIB FOR OPNDST
         MVC   RPLUSFLD,4(R4)          PUT USER FIELD IN OPNDST RPL ...
*                                      VTAM DOES NOT SET IT ON OPNDST
         MVC   FIRSTMID(8),0(R4)       PUT ID IN GOOD MORNING MESSAGE
         B     VALIDATE
CANCEL2  ST    R1,R1CONTS2             SAVE THE CONTENTS OF REGISTER 1
         ABTERM
*********************************************************************
* VALIDATE THE LOGON MESSAGE                                        *
*********************************************************************
VALIDATE EQU   *
         L     R8,12(R6)               PUT LENGTH OF LOGON MSG IN 8
         LTR   R8,R8                   IS LOGONMSG LENGTH ZERO?
         BZ    DISCONN                 YES -- TERMINATE THE SESSION
*********************************************************************
* CLEAR THE LOGON MESSAGE DATA                                      *
*********************************************************************
         MVI   MSGAREA,C'*'
         MVC   MSGAREA+1(79),MSGAREA
INQUIRE  INQUIRE RPL=PRPLCONN,OPTCD=LOGONMSG,NIB=PNIB,     OBTAIN
               AREA=MSGAREA,AREALEN=L'MSGAREA,             LOGON
               ACB=(R9)                                    MESSAGE
         LTR   R15,R15
         BNZ   CANCEL2
         LTR   R0,R0                   IS CONDITIONAL COMPLETION CODE 0
         BZ    COMPARE                 YES, CHECK MESSAGE
         B     DISCONN                 NO, SHOULD NOT OCCUR
COMPARE  CLC   MSGAREA(3),=C'XYZ'      CHECK PASSWORD IN USER LOGON
*                                      MESSAGE
         BNE   DISCONN                 IF NOT, CANNOT GRANT REQUEST
CONNECT  OPNDST RPL=PRPLCONN,OPTCD=(SYN,ACCEPT,CA)
         LTR   R15,R15                 SESSION ESTABLISHED SUCCESSFULLY
         BZ    SNDFIRST
         B     RETURN2
SNDFIRST EQU   *
         SEND RPL=PRPLCONN,AREA=FIRSTMSH,ACB=(R9),   SEND FIRST MESSAGE
               RECLEN=L'FIRSTMSG+6+L'FIRSTMID,OPTCD=CA, STILL SET
               RESPOND=(EX,FME)          INPUT MAY SATISFY RECANY.
*                                      ABANDON THIS SESSION .....
*                                      SYNAD WILL HAVE CLSDST FOR US
RETURN2  EXIT  RESTORE=NO,R14=SAVE1
*
*TERMINATE THE SESSION
*   IT MIGHT BE BETTER TO SEND A REJECTION MESSAGE TO THE VTAM
*   OPERATOR BEFORE CLOSING.
DISCONN  EQU   *
         CLSDST RPL=PRPLCONN,OPTCD=SYN,ACB=(R9),NIB=PNIB
*   IF CONTROL RETURNS HERE THERE IS NO NEED TO TEST FOR SUCCESS OR
*   FAILURE SINCE LERAD OR SYNAD COPE WITH FAILURE.
         B     RETURN2                 IF SO, BRANCH TO RETURN
*
*
*********************************************************************
*                                                                   *
*                      VARIABLE DECLARATIONS                        *
*                                                                   *
*********************************************************************
```

```
*
*********************************************************************
*                                                                   *
* RPL AND NIB                                                       *
*                                                                   *
*********************************************************************
PRPLCONN RPL   AM=VTAM
PNIB     NIB                           ALLOW USE OF RESP EXIT FOR LU
               PROC=(RESPX,TRUNC)      AND TRUNCATE EXCESS INPUT DATA
*********************************************************************
*                                                                   *
* CONSTANTS                                                         *
*                                                                   *
*********************************************************************
SAVESENS DC    F'0'                    SENSE FROM FAILED OPNDST
MSGAREA  DC    CL80'  '                AREA FOR LOGON MESSAGE
FIRSTMSH DC    XL6'840000000000'       HEADER CODE FOR DISPLAY ON
*                                      TERMINAL AND ECHO BACK TO
*                                      PROG1.
FIRSTMID DC    CL9'********-'
FIRSTMSG DC    C'LOGON ACCEPTED. VTAM PROG READY FOR FIRST INPUT'
*********************************************************************
*                                                                   *
* LOCAL STORAGE VARIABLES                                           *
*                                                                   *
*********************************************************************
SAVE1    DS    F                       SAVE REG14 RETURN ADDRESS
SAVE2    DS    18F                     SAVEAREA FOR VTAM EXITS
R1CONTS2 DS    F'0'                    SAVEAREA FOR REG 1 FOR DUMP
         LTORG
         EJECT
         ISTDNIB                       INVOKE NIB, DEVCH, AND PROC
                                       DSECT
*
*********************************************************************
*********************************************************************
*                                                                   *
*                      RESP EXIT                                    *
*                                                                   *
*********************************************************************
SAMP1    CSECT                         CONTINUE SAMP1 CSECT
*********************************************************************
*
*   NAME = RESP EXIT ROUTINE
*
*   FUNCTION = RECEIVE A RESPONSE TO THE REQUEST SENT IN THE MAINLINE
*      PROGRAM. IF THE RESPONSE IS NORMAL (POSITIVE), RESET THE SESSION
*      TO CONTINUE-ANY MODE SO THAT THE MAINLINE PROGRAM RECEIVE
*      OPTCD=ANY SPECIFIED ACCEPTS INPUT FROM IT.  IF THE RESPONSE IS
*      NEGATIVE, CALL SYNAD1 TO ANALYZE THE EXCEPTION AND TAKE
*      WHATEVER ACTION IS POSSIBLE.  SYNAD'S ACTION IS EITHER TO CLSDST
*      THE FAILING SESSION OR TO PERFORM A SESSIONC CONTROL=CLEAR AND
*      SDT.  IN BOTH CASES CONTROL IS RETURNED TO THIS EXIT AT LABEL
*      SYNRTURN.
*
*   ENTRY POINT = RESP1
*
*   INPUT
*   REGISTERS
*      0    =   UNPREDICTABLE
*      1    =   ADDRESS OF A 5-WORD PARAMETER LIST
*      2-13 =   UNPREDICTABLE
*      14   =   ADDRESS TO RETURN CONTROL TO
*      15   =   ENTRY ADDRESS TO THIS ROUTINE
*   PARAMETER LIST - 5 WORDS
*      1    =   ADDRESS OF THE ACB
*      2    =   THE CID OF THE LOGICAL UNIT
```

```
*      3    =   THE CONTENTS OF THE USERFLD (FROM
*               THE NIB SPECIFIED AT OPNDST)
*      4    =   UNPREDICTABLE
*      5    =   THE ADDRESS OF A READ-ONLY RPL THAT IS
*               USED TO DETERMINE WHAT KIND OF RESPONSE
*               HAS BEEN RECEIVED
*
*   OUTPUT = A RESETTING TO CONTINUE-ANY MODE FOR ANY SESSION
*      FROM WHICH A RESPONSE IS RECEIVED.
*
*   EXTERNAL REFERENCES = RESETSR, SYNAD1.
*
*   EXIT, NORMAL = BR 14
*
*   EXIT, ABNORMAL = DUMP
*
*   ATTRIBUTES = SERIALLY REUSABLE
*
*   REGS USED
*
*      3 = RETURN ADDRESS
*      4 = A(PRPLR),IFGRPL
*      5 = A(VRPL),IFGRPL
*      6 = WORK,A(RESP1 PARM LIST)
*      8 = CID
*      9 = A(ACB)
*     12 = BASE REG
*     13 = A(SAVE2)
*********************************************************************
RESP1    ENTER SAVE=NO,R14=SAVE4,TPEND=CHECK
*
         LR    R6,R1                  SAVE PARAMETER LIST ADDRESS
         L     R9,0(R6)               PICK UP ACB ADDRESS
         L     R5,16(R6)              PUT ADDR OF READ ONLY RPL IN R5
         LA    R4,PRPLR               INITIALIZE R4
         DROP  R5                     FROM LOGON EXIT USE
         USING IFGRPL,R4              BASE ON PRPLR
         MVC   RPLARG,4(R6)           MOVE CID TO PRPLR FOR RESETSR
         NI    RPLEXTDS,X'FF'-RPLNIB  TURN OFF NIB FLAG
         DROP  R4
         USING IFGRPL,R5              BASE ON READ-ONLY RPL
         TM    RPLVTFL2,RPLEX         NORMAL RESPONSE?
         BO    EXCEPTN
         B     RESET
CANCEL3  ST    R1,R1CONTS3            OTHERWISE, MUST TERMINATE AND
*                                     ABEND
RESET    RESETSR RPL=PRPLR,OPTCD=CA,  RESET THE SESSION FOR
               RTYPE=DFSYN,ACB=(R9)   DFSYN INPUT.  OPTCD-(SYN,SPEC)
         LTR   R15,R15                SEE IF RESETSR REQUEST ACCEPTED
         BNZ   CANCEL3                IF NOT, GO TO TERMINATE AND DUMP
RETURN3  EXIT RESTORE=NO,R14=SAVE4
*
EXCEPTN  EQU   *                      SET UP LINKAGE FOR SYNAD1
         STM   R14,R12,12(R13)        SAVE REGISTERS
         LA    R0,4                   SHOW EXTRAORDINARY COMPLETION
         L     R15,=A(SYNAD1)
         LR    R1,R5                  POINT TO READ-ONLY RPL
         BALR  R14,R15                CALL SYNAD ROUTINE
SYNRTURN LM    R1,R12,24(R13)         RESTORE RESP EXIT REGS
         LTR   R15,R15                SUCCESSFUL RECOVERY?
         BZ    RESET                  YES, ALLOW NEXT TRANSACTION IN
*********************************************************************
* SYNAD SETS R15-R12 IF THE CLSDST MACRO WAS ISSUED TO TERMINATE THE *
* SESSION; IN THIS CASE, NO RESETSR SHOULD BE ISSUED                 *
*********************************************************************
         B     RETURN3                RETURN TO VTAM
*
```

```
*
*********************************************************************
*                                                                   *
*                      VARIABLE DECLARATIONS                        *
*                                                                   *
*********************************************************************
*
*********************************************************************
*                                                                   *
* RPL                                                               *
*                                                                   *
*********************************************************************
PRPLR    RPL   AM=VTAM                COULD BE SAME ONE AS PRPLCONN
*                                     SINCE BOTH ARE SYNCHRONOUSLY
*                                     USED IN VTAM EXITS.
*********************************************************************
*                                                                   *
* CONSTANTS                                                         *
*                                                                   *
*********************************************************************
R1CONTS3 DC    F'0'                   SAVEAREA FOR REG 1 AT DUMP
*********************************************************************
*                                                                   *
* LOCAL STORAGE VARIABLES                                           *
*                                                                   *
*********************************************************************
SAVE4    DS    F                      SAVEAREA FOR EXIT RETURN ADDRESS
         LTORG
*
*
*********************************************************************
*********************************************************************
*
*   NAME = LERAD EXIT ROUTINE
*
*   FUNCTION = HANDLE TELEPROCESSING-ORIENTED LOGIC ERRORS
*
*   ENTRY POINT = LERAD1
*
*   INPUT
*   REGISTERS
*      0    =   RECOVERY ACTION RETURN CODE
*      1    =   RPL ADDRESS
*      2-12 =   UNPREDICTABLE
*      13   =   ADDRESS OF SAVE AREA SUPPLIED TO MACRO THAT
*               CAUSED LERAD ENTRY
*      14   =   RETURN ADDRESS
*      15   =   ADDRESS OF THIS ROUTINE'S ENTRY POINT
*
*   OUTPUT = NONE
*
*   EXTERNAL REFERENCES =   GETMAIN (MVS/VM), FREEMAIN (MVS/VM).
*
*   EXIT, NORMAL = BR 14
*
*   EXIT, ABNORMAL = DUMP
*
*   ATTRIBUTES = REENTRANT
*
*   REGS USED
*
*       3 = RETURN ADDRESS
*       6 = A(RPL),IFGRPL
*      12 = BASE REG
*      13 = A(SAVEAREA)
*********************************************************************
LERAD1   ENTER SAVAREA=GET,SAVE=NO,R14=(R10)
```

```
         DROP  R5                      USED IN RESP EXIT
         USING IFGRPL,R1
         CLI   RPLFDB2,X'12'
         BE    IGNORE
         CLI   RPLFDB2,X'13'
         BE    IGNORE
         CLI   RPLFDB2,X'60'           CLSDST W/SYMBOLIC NAME FAILED?
         BE    IGNORE                  YES SO IGNORE
         B     LEOVERID                BRANCH AROUND DUMP ID
         DC    C'LERAD1'               DUMP ID
R1DUMP   DC    F'0'                    REG1 CONTENTS AT DUMP
LEOVERID EQU   *
         ST    R1,R1DUMP               SAVE REG 1 FOR DUMP
         ABTERM
IGNORE   EQU   *
         SLR   R0,R0                   INDICATE SUCCESSFUL COMPLETION
         SLR   R15,R15                 INDICATE SUCCESSFUL COMPLETION
         EXIT  SAVAREA=FREE,RESTORE=YES,R14=(R10)
         LTORG

*********************************************************************
*
*   NAME = TPEND EXIT ROUTINE
*
*   FUNCTION = SET AN INDICATION FOR THE MAINLINE PROGRAM
*      TO CLOSE THE ACB AND TERMINATE
*
*   ENTRY POINT = TPEND1
*
*   INPUT
*   REGISTERS
*      0    =   UNPREDICTABLE
*      1    =   ADDRESS OF A 2-WORD PARAMETER LIST
*      2-13 =   UNPREDICTABLE
*      14   =   RETURN ADDRESS
*      15   =   ADDRESS OF THIS ROUTINE'S ENTRY POINT
*   PARAMETER LIST - 2 WORDS
*      1    =   ADDRESS OF THE ACB
*      2    =   A VALUE INDICATING WHY TPEND WAS ENTERED
*
*   OUTPUT  =  INDICATION TO CLOSE ACB SET FOR MAIN PROGRAM
*
*   EXTERNAL REFERENCES = POST.
*
*   EXIT, NORMAL = BR 14
*
*   EXIT, ABNORMAL = NONE
*
*   ATTRIBUTES = SERIALLY REUSABLE.
*
*    REGS USED
*
*       3 = RETURN ADDRESS
*       4 = A(TPENDECB)
*      12 = BASE REG
*
*********************************************************************
TPEND1   ENTER SAVE=NO,SAVAREA=NONE,R14=TPENDS14
         L     R4,=A(TPENDECB)         POINT TO MAINLINE'S CLOSEDOWN ECB
         POST  (R4)                    INDICATE TPEND REQUIRED
         EXIT  RESTORE=NO,R14,=TPENDS14
TPENDS14 DC    F'0'                    SAVE AREA FOR VTAM RETURN ADDRESS
         LTORG
*********************************************************************
*
*   NAME = SYNAD EXIT ROUTINE
*
```

```
*   FUNCTION = HANDLE ERRORS AND SPECIAL CONDITIONS OTHER THAN
*      TELEPROCESSING LOGIC ERRORS.  ATTEMPTS TO CLEAR
*      THE CONDITION OR TERMINATE THE SESSION.
*
*   ENTRY POINT = SYNAD1
*
*   INPUT
*   REGISTERS
*      0    =  RECOVERY ACTION RETURN CODE
*      1    =  RPL ADDRESS (HIGH-ORDER BIT ON IF RECURSIVE ENTRY)
*      2-12 =  UNPREDICTABLE
*      13   =  ADDRESS OF SAVE AREA SUPPLIED PRIOR TO CAUSING
*              SYNAD ENTRY
*      14   =  RETURN ADDRESS
*      15   =  ADDRESS OF THIS ROUTINE'S ENTRY POINT
*
*   OUTPUT  =  A VALUE SET IN REGISTER 15:
*       0   =  SUCCESSFUL RECOVERY
*       8   =  EXCEPTION REQUEST RECEIVED
*      12   =  CLSDST PERFORMED
*
*   EXTERNAL REFERENCES = SESSIONC, SEND,
*      RESETSR, CLSDST, GETMAIN (MVS/VM), FREEMAIN (MVS/VM),
*      AND EXECRPL.
*
*   EXIT, NORMAL = BR 14
*
*   EXIT, ABNORMAL = DUMP
*
*   ATTRIBUTES = QUASI-REENTERABLE.  THIS ROUTINE IS REENTERED
*      IF A MACROINSTRUCTION IT ISSUES FAILS.  AS INDICATED
*      BY THE HIGH-ORDER BIT OF REG 1 BEING ON UPON ENTRY
*      TO SYNAD1. THE PROGRAM TERMINATES AND A DUMP IS
*      REQUESTED.  OTHERWISE, IF SYNAD IS REENTERED,
*      PROCESSING CONTINUES.
*
*   REGS USED
*
*       3 = RETURN ADDRESS
*       4 = ACTION CODE
*       5 = A(RPL),IFGRPL
*       6 = A(GETMAIN RPL),IFGRPL
*       7 = REG0 RETURN CODE
*       8 = REG15 RETURN CODE, A(PARMLIST FOR MANIP MACROS)
*       9 = A(PACB)
*      10 = LINKAGE TO SESSIONC
*      11 = CID
*      12 = BASE REG
*      13 = A(GETMAIN SAVEAREA),SAVE5
*
*********************************************************************
SYNAD1   ENTER SAVAREA=GET,SAVE=NO,XTRA=SDXTRA,R14=(R10)
         DROP  R1                    USED IN LERAD EXIT
         LR    R5,R1                 GET RPL ADDRESS
         LR    R4,R0                 GET ACTION CODE
         USING IFGRPL,R5
         USING SDSECT,R13            SET BASE FOR REENTRANT WORKAREA
*********************************************************************
* CHECK FOR RECURSIVE ENTRY TO SYNAD                                *
*********************************************************************
         ST    R5,REGNWORK
         TM    REGNWORK,X'80'        IS THIS RECURSIVE ENTRY TO SYNAD?
         BO    CANCEL4               YES -- CANCEL
         OI    REGNWORK,X'80'        NO -- INDICATE RECURSION
         L     R5,REGNWORK           SAVE RPL ADDRESS
         LA    R0,SRPLEND-SRPL       SET LENGTH OF RPL IN R0
         GETSTOR (R0)
```

```
         LTR   R15,R15
         BNZ   CANCEL4
         MVC   0(SRPLEND-SRPL,R1),SRPL COPY SRPL
         ST    R1,REGNWORK            POINT TO SYNAD1'S OWN RPL
         OI    REGNWORK,X'80'         SET HIGH-ORDER BIT OF R6
         L     R6,REGNWORK            (RPLSYN ADDRESS) FOR RECURSION.
         L     R9,=A(PACB)            PICK UP ADDRESS OF ACB
         L     R11,RPLARG
         CH    R4,=H'16'              IS IT OVER MAX FOR SYNAD?
         BH    CANCEL4                YES,GIVE UP
         B     *+4(R4)                USE ACTION CODE IN BRANCH TABLE
         B     SNORM                  CODE=X'00' SHOULD NOT OCCUR
         B     SXTRA                  CODE=X'04' EXTRAORD. COMPLETION
         B     SRETRY                 CODE=X'08' RETRIABLE
         B     SDAMAGE                CODE=X'0C' DAMAGE
         B     SENVIR                 CODE=X'10' ENVIRONMENT ERROR
*
SNORM    SR    R7,R7                  INDICATE SUCCESSFUL
         SR    R8,R8                  COMPLETION.
SABNORM  L     R0,SAVE6               LENGTH OF STORAGE TO BE FREED
         SLL   R6,1                   GET RID OF HIGH-ORDER BIT
         FREESTOR AREA=(R6),LEN=(R0)
         LTR   R15,R15
         BNZ   CANCEL4
         EXIT  RESTORE=NO,SAVAREA=FREE,RC=YES
SYNADR1  DC    F'0'                    SPACE FOR R1
CANCEL4  ST    R1,SYNADR1
         ABTERM
*
SXTRA    EQU   *                       EXTRAORDINARY COMPLETION
SXPATHE  TM    RPLSSEI,RPLPATHI
         BO    SDISCONN
         CLI   RPLFDB2,X'03'
         BE    EXMSG                   YES--EXCEPTION REQUEST RECEIVED
*                                       NO--EXCEPTION RESPONSE RECEIVED
         LA    R10,SNORM               PREPARE FOR NORMAL RETURN
*
         DROP  R5
         USING IFGRPL,R6
SESSIONC SESSIONC  RPL=(R6),ACB=(R9),ARG=(R11),  CLEAR SESSION
               CONTROL=CLEAR,STYPE=REQ,OPTCD=SYN
         LTR   R15,R15
         BNZ   SDISCONN
         SESSIONC  RPL=(R6),CONTROL=SDT   START DATA TRAFFIC
         LTR   R15,R15                 SUCCESSFUL RECOVERY?
         BNZ   SDISCONN                NO, TERMINATE THE SESSION
         BR    R10                     YES, RETURN TO CALLER
         DROP  R6
         USING IFGRPL,R5
*
EXMSG    EQU   *              THIS CANNOT BE REACHED FROM THE RESP EXIT
         TM    RPLVTFL2,RPLNFME
         BO    STBAL
*********************************************************************
* MOVE SSENSEI TO SSENSEO                                           *
*********************************************************************
         MVC   RPLSSEO,RPLSSEI
STEND    EQU   *
         MVC   RPLSSMO,RPLSSMI
         SEND  RPL=(R5),STYPE=RESP,       SEND THE EXCEPTION RESPONSE
               OPTCD=SYN
         LTR   R15,R15
         BNZ   SDISCONN
         DROP  R5
         USING IFGRPL,R6
STBAL    BAL   R10,SESSIONC            GO THROUGH CLEAR AND SDT
         RESETSR RPL=(R6),RTYPE=DFSYN, RESTORE TO CA MODE
```

```
               OPTCD=(SYN,CA)
         LTR   R15,R15
         BNZ   CANCEL4
         LA    R8,8                    SIGNAL UNSUCCESSFUL COMPLETION
         B     SABNORM                 RETURN TO RECANY
*
SDISCONN EQU   *                       UNRECOVERABLE ERRORS
         CLSDST RPL=(R6),ACB=(R9),ARG=(R11)
         LTR   R15,R15
         BNZ   CANCEL4
         LA    R8,12                   SIGNAL UNSUCCESSFUL RECOVERY
         B     SABNORM
*
SRETRY   EQU   *
*********************************************************************
* RETRY REQUEST                                                     *
*********************************************************************
         EXECRPL RPL=(R5)              RETRY FAILED MACRO
         LTR   R15,R15
         BNZ   CANCEL4
         B     SNORM                   RETURN TO ORIGINAL NSI
*
SDAMAGE  EQU   *
         CLI   RPLREQ,RPLRCVCD
         BNE   SNORM                    NO, PRETEND COMPLETION WAS OK
         LA    R8,16                    YES, SET R15 CODE REG NONZERO
         B     SABNORM                  RETURN TO NSI, WHICH MAY BE
*                                       ABLE TO IGNORE THE ERROR
*
SENVIR   EQU   *
         CLI   RPLREQ,RPLSNDCD
         BE    SDISCONN                ATTEMPT TO CLSDST
         CLI   RPLREQ,RPLRSRCD
         BE    SDISCONN                ATTEMPT TO CLSDST LU
         LA    R8,20                   SET NONZERO CODE AND ALLOW
         B     SABNORM                 IN-LINE CODE TO RECOVER.
*                                      (MAY BE AN OPNDST OR INQUIRE)
*
*
*********************************************************************
*                                                                   *
*                     VARIABLE DECLARATIONS                         *
*                                                                   *
*********************************************************************
*********************************************************************
*                                                                   *
* BASED STORAGE AREA AND VARIABLES                                  *
*                                                                   *
*********************************************************************
SDSECT   DSECT
SAVE5    DS    18F                     NEW SAVEAREA
SHOWWORK DS    0F
SFDBK2   DS    F                       SPECIFIC REASON CODE
SSSENSMI DS    F                       SYSTEM SENSE MODIFIER INPUT
REGNWORK DS    F                       FOR RETRIABLE ERRORS
SAVE6    DS    F                       LENGTH OF RPL
SARG     DS    F                       ARG - CID VALUE
SDXTRA   EQU *-SHOWWORK
*
*********************************************************************
*                                                                   *
* RPL                                                               *
*                                                                   *
*********************************************************************
SAMP1    CSECT
SRPL     RPL   AM=VTAM                 RPL TO BE COPIED
SRPLEND  EQU   *                       END OF SRPL FOR LENGTH CALC.
```

```
SAMP1    CSECT
         LTORG
*
*
*********************************************************************
 *********************************************************************
 *
 *   NAME = LOSTERM EXIT ROUTINE
 *
 *   FUNCTION = HANDLE SITUATIONS IN WHICH A LOGICAL UNIT HAS
 *      UNEXPECTEDLY BECOME UNAVAILABLE
 *
 *   ENTRY POINT = LOSTERM1
 *
 *   INPUT
 *   REGISTERS
 *      0    =   UNPREDICTABLE
 *      1    =   ADDRESS OF A 4-WORD PARAMETER LIST
 *      2-13 =   UNPREDICTABLE
 *      14   =   RETURN ADDRESS
 *      15   =   ADDRESS OF THIS ROUTINE'S ENTRY POINT
 *   PARAMETER LIST - 4 WORDS
 *      1    =   ADDRESS OF THE ACB
 *      2    =   THE CID OF THE LOGICAL UNIT
 *      3    =   THE CONTENTS OF THE USERFLD (FROM THE NIB
 *               SPECIFIED AT OPNDST)
 *      4    =   A VALUE INDICATING WHY LOSTERM WAS ENTERED
 *
 *   OUTPUT = TERMINATION OF THE SESSION
 *
 *   EXTERNAL REFERENCES = CLSDST, DUMP
 *
 *   EXIT, NORMAL = BR 14
 *
 *   EXIT, ABNORMAL = DUMP
 *
 *   ATTRIBUTES = SERIALLY REUSABLE.
 *
 *   REGS USED
 *
 *       3 = RETURN ADDRESS
 *       4 = A(PRPLCONN)
 *       5 = A(ACB)
 *       6 = CID
 *       7 = A(TPENDFLG)
 *      12 = BASE REG
 *      13 = A(SAVE2)
 *
 *
 *********************************************************************
LOSTERM1 ENTER SAVE=NO,SAVAREA=NONE,TPEND=CHECK,R14=SAVELOST
         L     R4,=A(PRPLCONN)         POINT TO OPNDST/CLSDST RPL
         L     R5,0(R1)                PICK UP ACB ADDRESS
         DROP  R6
         USING IFGRPL,R4               BASE ON PRPLCONN
         MVC   RPLUSFLD,8(R1)          MOVE USER FIELD
         L     R6,4(R1)                PICK UP CID OF LOST TERMINAL
         LR    R8,R1                   POINT TO PARMLIST
LOSTCLOS EQU   *
         CLSDST RPL=(R4),ACB=(5),ARG=(R6),OPTCD=(RELEASE,SYN)
         LTR   R15,R15
         BZ    RETURNL
         ABTERM
RETURNL  EXIT  RESTORE=NO,R14=SAVELOST
*
*
*********************************************************************
```

```
*                                                                     *
*                       VARIABLE DECLARATIONS                         *
*                                                                     *
***********************************************************************
*
***********************************************************************
*                                                                     *
* CONSTANTS                                                           *
*                                                                     *
***********************************************************************
SAVELOST DC    F'0'
         LTORG
         END   SAMP1
```

Chapter 16. Logic of a more complicated application program

Sample program 2 is a more typical example of a VTAM application program than sample program 1.

Sample program 2 communicates with logical units (LUs) associated with 3600 Finance Communication System controllers and SNA PU type 1 3270 Information Display Systems controllers. Some LUs are associated with physical units (PUs) that are connected to VTAM by nonswitched remote lines through a communication controller that contains a network control program. Other LUs are associated with non-SNA 3270 terminals that are attached to VTAM through a channel. Additional information about communicating with 3270 terminals is given in Chapter 11, "Programming for the IBM 3270 Information Display System," on page 327. Note that in this sample program all the 3270 terminals use LU type 0 protocols.

Introduction

Figure 97 on page 606 shows a possible configuration with which sample program 2 might communicate.

The application program in the 3601 controller can be written to perform certain functions that would otherwise have to be performed by the VTAM application program. For example, the 3601 application program can screen inquiries for the correct format prior to forwarding them to the VTAM application program for processing, or it can collect inquiries from several terminals that form a work station and send them as one transmission to the VTAM application program.

The logic of sample program 2 is described at a high level in Figure 98 on page 608 and accompanying notes. The logic of special routines is described in more detail in subsequent figures and accompanying notes.

© Copyright IBM Corp. 2000, 2013

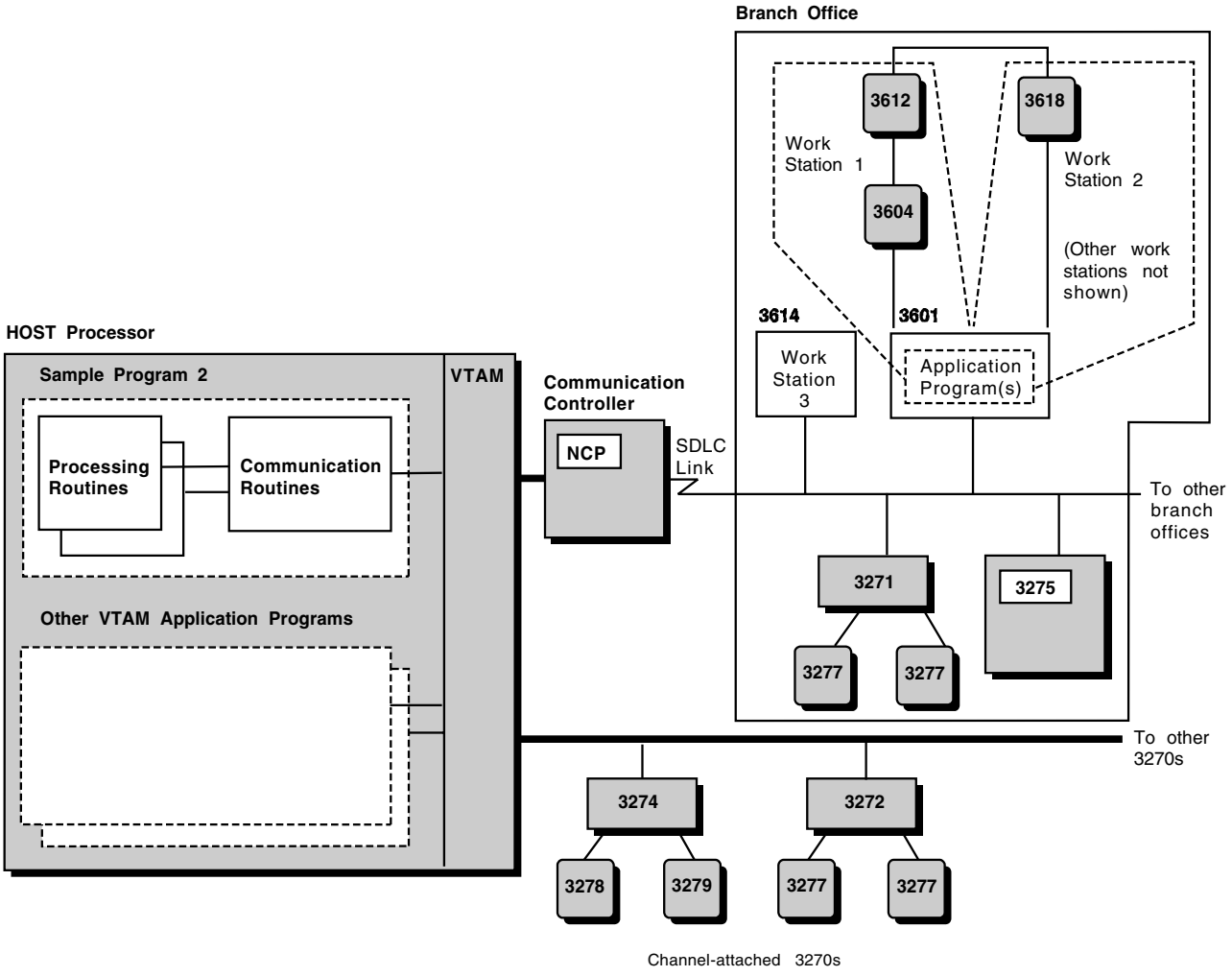


Figure 97. Possible data communication configuration for sample program

Sample program 2 uses the posting of ECBs (either by VTAM or within the program) and a central wait routine that discovers posted ECBs as a mechanism to handle a number of sessions concurrently without having to suspend all program execution while waiting for an I/O operation to be completed. After a request has been issued for an asynchronous operation, control is transferred to the wait routine, which discovers (or, if necessary, waits for) a posted ECB. The posted ECB can be associated with another session and the wait routine branches to a point related to further processing for the session for which an operation has been completed. An understanding of the details of this technique is assumed in this discussion.

Although not discussed in detail, it is likely that sample program 2 would use a separate control block for each session that was established with the program. This control block can include an input/output area. A separate RPL can also be associated exclusively with each session. The storage for these control blocks can be obtained from a fixed pool or can be obtained dynamically and initialized with the GENCB macroinstruction. This control block and RPL area can be obtained and related to a session for the duration of that session, for the duration of the program, for the duration of a transaction, or on some other basis. The ECB associated with a session can be located in the RPL or outside of it in some fixed

relationship, perhaps just in front of it. Chapter 3, “Organizing an application program,” on page 33, discusses some alternative approaches for control block storage management. In sample program 2, it is assumed that the storage for an ECB, RPL, and session-related control block is obtained and initialized in the LOGON exit routine and retained for the duration of the session with the program.

Organization and flow of Sample Program 2

Figure 98 on page 608 shows the principal routines in sample program 2; the notes following the figure indicate how sample program 2 works. More detailed logic is shown and discussed in subsequent figures and notes.

Figure 98 on page 608 shows the mainline program and the exit routines as separate groups of routines. This is a logical rather than a physical separation; exit routines are distinctive because they are entered only when an event occurs that requires handling by an exit routine. When an asynchronous exit routine is scheduled, VTAM suspends execution of the mainline program until the exit routine completes its processing and returns to VTAM. In general, only one asynchronous exit routine can be executed at a time; if an exit routine event occurs while an exit routine is being executed, the second exit routine is scheduled for entry only after the first exit routine is completed. The LERAD and SYNAD exit routines are exceptions to this general rule; they can be entered as the result of an RPL-based operation, such as OPNDST, RECEIVE, or CHECK in another exit routine (in which case, they can be viewed as extensions of the exit routine that caused them to be entered). For details, see “Normal operating system environment for a VTAM application program” on page 30.

Except for the LERAD and SYNAD exit routines, each exit routine must establish its own addressability, be executed, and then return to VTAM; VTAM's registers need not be saved or restored. A temporary branch to part of the mainline routine can be made from an exit routine and common code can be shared, but the exit routine would be considered to be in progress until control is returned to VTAM. The LERAD and SYNAD exit routines are furnished addressability as the result of loading registers (a user save area address is passed in register 13).

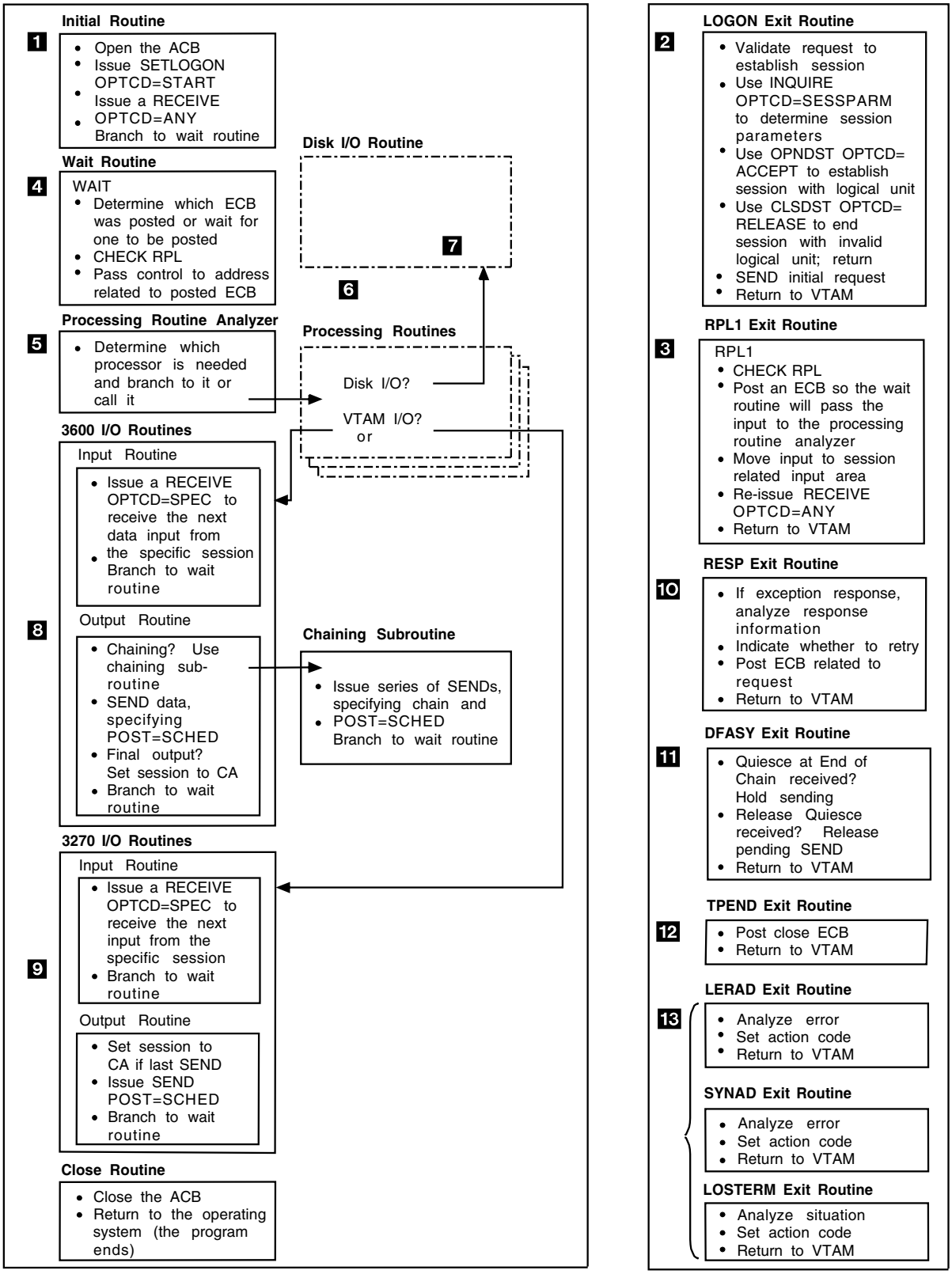


Figure 98. Organization and flow of Sample Program 2

Except for an RPL exit routine, whose address is specified in the RPL or RPL-based macroinstruction, the addresses of exit routines to be associated with the program

are defined in an EXLST macroinstruction. In addition, for DFASY, RESP, and SCIP exit routines, different exit lists can be defined for different sessions or sets of sessions. In sample program 2, one exit list is assumed; the address of this exit list is provided to VTAM in the EXLST operand of the ACB when the ACB is opened.

The following notes are keyed to the numbers in Figure 98 on page 608.

1 The ACB is opened and a SETLOGON OPTCD=START is issued. A RECEIVE to read input from any session is issued; the operation is to be completed asynchronously, and an RPL exit routine (RPL1) is designated for scheduling by VTAM when the operation is completed. The session whose input is read into sample program 2 is to be put into continue-specific mode. Thus, subsequent RECEIVEs to read input from any session, issued in the RPL exit routine, excludes the session whose input was just read and with whom the program is now in specific communication. The RECEIVE can be coded:

```
RECEIVE  RPL=RPL1ANY,AREA=AREAANY,AREALEN=100,                         C
               RTYPE=DFSYN,OPTCD=(ASY,ANY,CS),EXIT=RPL1
```

Issuing more than one RECEIVE OPTCD=ANY at this point can improve efficiency. If three RECEIVEs are issued using three different RPLs and data areas, when one RECEIVE is completed (thus causing the RPL1 exit routine to be scheduled and entered), the two outstanding RECEIVEs can allow scheduling of RPL1. The RECEIVE that is completed first can be reissued in the RPL1 exit routine.

The RPLs and input areas can be assembled in the program as fixed areas and reused each time the program issues a RECEIVE OPTCD=ANY.

A branch is made to the wait routine, which waits for the first input to arrive from a session. The initial routine is executed only once.

2 Both 3600 and 3270 LUs can cause the LOGON exit routine to be invoked. INQUIRE is used to determine which type of LU is having a session established. The particular type of SNA terminal product (3600, 3790, and 3270, for example) need not be identified by the VTAM application program. Instead, the VTAM application program distinguishes between types of LUs on the basis of the session parameter set associated with the LU. In this example, 3600 LUs have a different set of session parameters than 3270 LUs; the program can relate to an LU that is establishing a session with one of these sets, and use the appropriate routine to communicate with the LU. Storage that is to be associated with this session can be obtained from a pool or can be obtained dynamically from the system. (The storage can include an ECB, an RPL, and an area for additional session-related information.) The address of this storage or any other session-related information can be put in the USERFLD field of the NIB, using the MODCB macroinstruction; when the session is established, VTAM saves this address and returns it to the program following completion of each subsequent RPL-based macroinstruction on the session.

A session with an LU can be established as the result of a logical-unit-initiated logon, installation-initiated (automatic) logon, VTAM operator-initiated logon, or application-program-initiated logon. A 3600 logical-unit-initiated logon either could be the result of some logical-unit-operator action or could be initiated solely by the 3601 application program without involving an LU operator (in either case, the actual request would be transmitted by the 3601 application program). Figure 107 on page 694 shows part of the general sequence of events that

occur prior to and during a logon. Here is a sequence of events that can occur prior to and during a 3600 logical-unit-initiated logon:

1. The 3601 and its LUs, defined to VTAM during VTAM definition, are made an active part of the VTAM network (perhaps by a VTAM operator VARY command).
2. As a result of receiving an activation request for the 3601 controller, VTAM sends an Activate Physical Unit request to the 3601 controller. The 3601 acknowledges the request and responds that it is ready for operation. (This request is followed by an Activate Logical Unit request to one or more of the LUs [logical work stations] associated with that particular 3601.)
3. After the Activate Logical Unit request is received, the 3601 application program either can wait for a terminal operator at a 3601 work station to indicate that the LU (work station) is to be logged on to a host application program, or can issue a logon on its own initiative. This is done by sending an Initiate request to VTAM specifying the name of the VTAM application program with which the LU is to be in session. The Initiate request can also specify a logon mode name (indirectly specifying a session parameter set) and a user-defined logon message.
4. VTAM, receiving the Initiate request, schedules sample program 2's LOGON exit routine.
5. In the LOGON exit routine, after confirming the validity of the logon, the session is established, using an OPNDST OPTCD=ACCEPT macroinstruction. OPNDST causes VTAM to send a BIND request (containing the session parameter) and to issue a Start Data Traffic (SDT) request to the LU (assuming SDT=SYSTEM was specified in the NIB used for session establishment). On receipt of a response to the SDT, the LU is in session with the VTAM application program and the OPNDST is posted complete. (If SDT=APPL is specified in the NIB used for session establishment, the VTAM application program must itself send the initial SDT, using the SESSIONC CONTROL=SDT macroinstruction.)

OPNDST can be specified as a synchronous operation or an asynchronous operation. If the latter, OPNDST can identify either an ECB to be posted or an RPL exit routine to be scheduled as soon as the session is established.

Note: For better system performance, specify OPNDST asynchronously from EXLST exit routines.

For a synchronous operation, the same RPL and NIB can be used for each session being established through the LOGON exit routine.

So that the session parameter suggested to be associated with the session can be obtained (which in this program determines whether the 3600 or the 3270 I/O routines are used with the session), an INQUIRE OPTCD=SESSPARM macroinstruction can be issued. This allows the session parameter and user data (logon message) from the original Initiate to be inspected and perhaps saved in the storage that is to be associated with the session.

Optionally, the read-only RPL provided as input to the LOGON exit routine can be used to locate and inspect the CINIT RU. The CINIT contains the session parameter, user data, and other information.

It might be desirable to use the SEND macroinstruction to write an initial request on the session; this could be done from the LOGON exit routine or in an RPL exit routine following an asynchronously posted OPNDST operation.

3 As soon as the first input is received from a session, the operation started by the RECEIVE OPTCD=ANY issued in the initial routine is completed. The RPL1 exit routine is then scheduled and entered.

RPL1 can use the USER field or the ARG field of the RPL of the request just completed to locate the application program session-related control block. On entry to RPL1, the address of the RPL is in register 1.

A CHECK macroinstruction is required to free the RPL for reuse; it also causes LERAD or SYNAD exit routines to be entered if any error occurred.

RPL1 posts an ECB so that, subsequently, the wait routine in the mainline program determines that the input has been received and passes it to the processing routine analyzer. The RPL1 exit routine then reissues a RECEIVE OPTCD=ANY. Because a session-related RPL obtained in the LOGON exit routine is used for subsequent operations with the session, the RPL causing entry to RPL1 can be continuously reused by the RECEIVE in RPL1. This RECEIVE OPTCD=ANY operation is to be asynchronous with respect to the rest of the program and RPL1 is reentered each time the request is completed. The session whose input caused entry to RPL1 is now in continue-specific (CS) mode. The RECEIVE can be coded identically to the RECEIVE in the initial routine.

Although not shown in Figure 98 on page 608, the RPL exit routine sends a positive response to the request that caused it to be entered if a positive response is requested by the LU. If a response is to be sent for an exception condition, sending the negative response is probably performed in a SYNAD exit routine after a CHECK is issued.

An alternative to having an RPL exit routine for the RECEIVE OPTCD=ANY and related logic is to have this logic located in the mainline program and have an ECB posted. In sample program 2, one advantage to using an RPL exit routine is that input resulting from a RECEIVE OPTCD=ANY is handled sooner in an RPL exit routine than if an ECB were to be posted (which would require waiting until the next entry to the mainline program's wait routine). This gives some preference to handling the first input of a new transaction over transactions in progress.

4 The wait routine waits on a list of ECBs, with each ECB associated with a separate RPL. When an ECB is posted, the wait routine is activated, and the routine searches the ECB list to find the posted ECB and zeros it out. The routine then issues a CHECK macroinstruction. (The CHECK is bypassed here for the RECEIVE OPTCD=ANY that caused entry to the RPL1 exit routine, because CHECK has already been issued for that RPL). CHECK clears the RPL for reuse in the next request involving the session, and if an error occurred, the macroinstruction causes the LERAD or SYNAD routine to be entered. On return from CHECK, the feedback fields of the RPL contain information provided by VTAM; in addition, the LERAD or SYNAD routine can indicate action to be taken.

If the operation was successful, the wait routine branches to the address associated with the ECB. (In the case of the first input of a transaction, that address is the one for the processing routine analyzer.) When control is returned to the wait routine, the routine again searches the ECB list to see if another ECB has been posted. If a posted ECB is found, processing

continues as described above. If not, RECEIVE OPTCD=ANY is issued again, and the program enters the wait state.

5 The processing routine analyzer, which can consist of separate routines for different types of LUs, analyzes the input and branches to or calls the appropriate processor. This processor can be coded in a higher-level language, such as COBOL or PL/I.

6 The processing routine processes the input and prepares the output. This might require one or more disk I/O operations, which can be performed by calling a common disk I/O routine. When output is ready, or during a communication when the next input is required, the processing routine requests VTAM I/O, causing control to pass to an appropriate VTAM I/O routine.

7 The disk I/O routine requests a disk I/O operation asynchronously and uses the wait routine to wait for completion. This allows processing for other sessions to continue while a disk I/O operation for one session is under way.

8 Although not shown, a processing routine can return control to the next sequential instruction in the mainline program from which it was called; a branch can then be made to a common I/O routine, which in turn branches to a 3600 or a 3270 input or output routine. A special routine might be required to edit 3270 input and format 3270 output.

If the LU has the 3600 session parameter, an input or an output operation is requested as appropriate. The operation is specified as asynchronous; completion is determined when the ECB related to the session is posted. Before issuing the request, the address to which the wait routine should branch (the return address is the processing routine) is placed in the ECB-related session control block.

If additional input is requested, the input, when it arrives, is not used to satisfy the outstanding RECEIVE OPTCD=ANY request in RPL1 because the session is now in CS mode.

If output is requested, the data can be sent in a chain of requests; this is useful with output that is passed from the 3601 application program to a 3610 printer. The 3601 application program can store all requests of the chain in a buffer until the entire chain is received (or print each request as it arrives). The VTAM application program would ensure arrival of the entire chain by receiving a single positive response sent by the 3601 application program when the last request of the chain is received. This notifies the VTAM application program that the data transfer was successful.

If the output completes a transaction, the session is reset to continue-any (CA) mode so that input that begins the next transaction satisfies the RECEIVE OPTCD=ANY request that is issued in RPL1.

Details about the 3600 I/O routine are provided in Figure 99 on page 615 and in accompanying notes.

9 If the LU has the 3270 session parameter, different I/O routines are required. The size of I/O areas required can be different and the range of input that arrive can be wider. An additional requirement is the use of brackets for controlling the overlap of input and output with 3270. Details about the 3270 I/O routines are provided in Figure 101 on page 620 and in accompanying notes.

10 The RESP exit routine is scheduled and entered when a response arrives from an LU. A response is received by VTAM because the VTAM application program requested it in the RESPOND operand of the SEND macroinstruction. When the response is received, the operation, scheduled only in the 3270 or 3600 output routine, is now completed and the RESP exit routine can now post the ECB. If the operation was successful, the response is positive; if an error occurred, a negative response is indicated in the RPL. The RESP exit routine can set up parameters and branch to the SYNAD exit routine which analyzes the error and takes corrective action. The ECB is posted and control is returned to VTAM.

Details of the RESP exit routine are in Figure 102 on page 623 and accompanying notes.

11 In sample program 2, two kinds of expedited-flow data-flow-control requests can be received from a 3601 LU: a request to stop sending to the LU at the end of the chain that is currently being sent (a Quiesce at End of Chain [QEC] request) and a request to reinitiate sending after previously being requested to stop (a Release Quiesce [RELQ] request). The use of these requests might be desirable if a work station operator wants to interrupt a long series of printing so that the keyboard input can be entered. After handling the operator's request, the VTAM application program can resume printing. When either of these requests is received, VTAM schedules sample program 2's DFASY exit routine.

This exit routine does not apply to 3270 operation. Details of this routine are shown in Figure 103 on page 625 and accompanying notes.

12 The TPEND exit routine is scheduled and entered when the VTAM operator enters a HALT command or when VTAM terminates itself or is abnormally terminated. In addition to other possible processing, the TPEND exit routine posts a special close ECB so that, subsequently, the mainline program's wait routine branches to a CLOSE macroinstruction in the mainline program.

13 The LERAD, SYNAD, and LOSTERM exit routines handle different categories of errors or unusual situations. The LERAD or SYNAD exit routine can be entered as the result of any RPL-based request. The LOSTERM exit routine is scheduled asynchronously when certain situations occur, such as the deactivation of an LU by the terminal operator. Like other parts of the program, the LOSTERM exit routine can branch to the LERAD or SYNAD exit routine for problem analysis. The LERAD exit routine primarily handles logic errors; it is most likely for these to occur during the debugging stages of the program. This exit routine can gather information, format it, and save it for programmer analysis after the program ends. The SYNAD exit routine primarily handles physical errors; it determines what general action should be taken (for example, retry, end the session with the LU, terminate the program, or send a request to the VTAM operator) and either takes the action or passes an action code to the mainline program where the action is taken. The SYNAD exit routine can also record information related to situations that it handles for later problem analysis.

A number of error situations must be perceived and analyzed as the result of receiving a response from an LU; the response is analyzed following a SEND POST=RESP, following a RECEIVE RTYPE=RESP, or after a RESP exit routine is entered. Errors or special situations that result in negative responses cause the SYNAD exit routine to be entered when a synchronous macroinstruction or a CHECK macroinstruction is issued; the SYNAD exit

routine can determine the cause of the negative response by analyzing sense information in the RPL and then take appropriate action. The program, after determining that a negative response has been received, can also branch directly to the SYNAD exit routine.

Logic of the 3600 finance communication system I/O routine

This routine is entered directly or indirectly (perhaps from a common I/O branching routine in the mainline program) as the result of a request for input from a specific terminal with which a processor is currently engaged in a transaction.

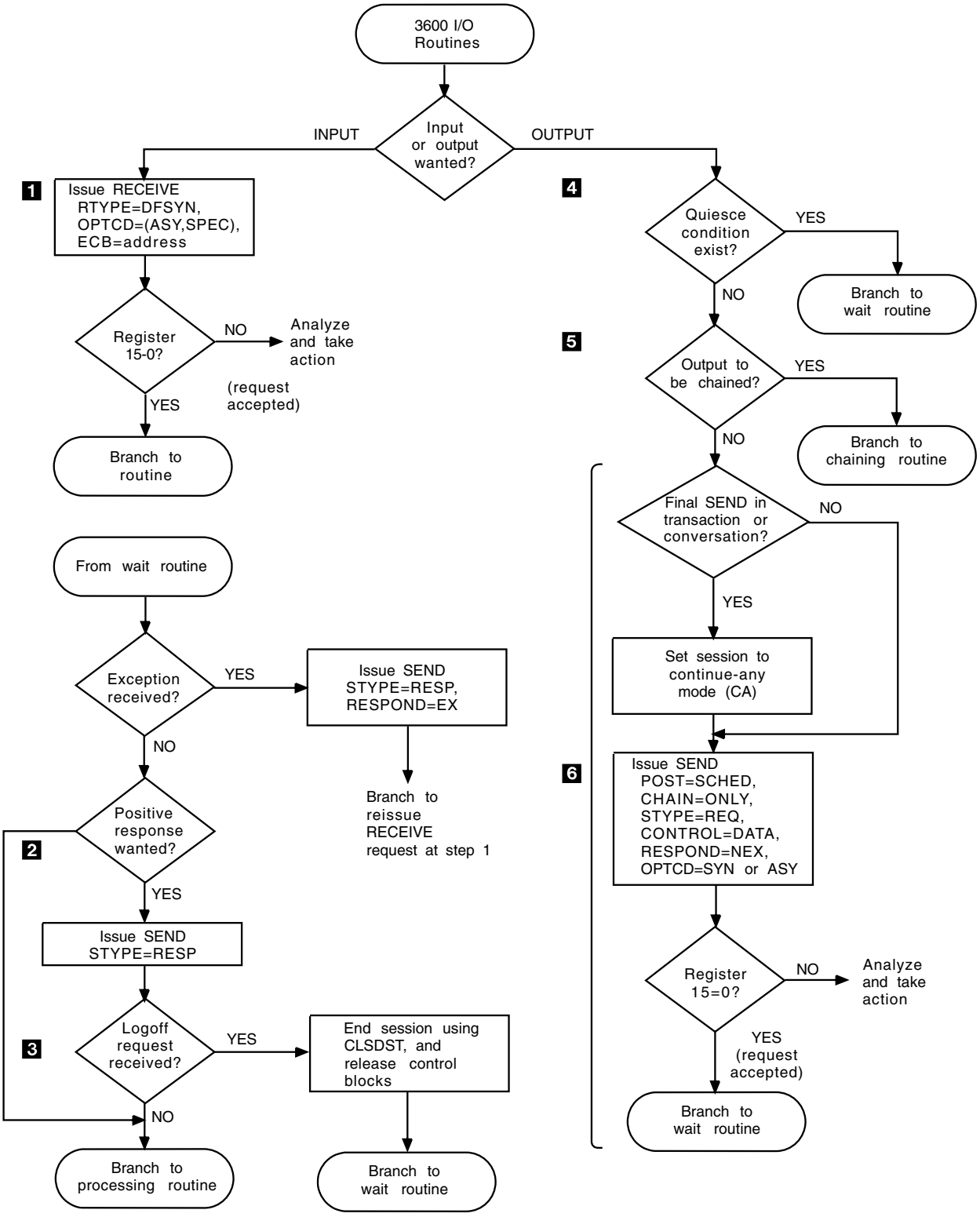


Figure 99. Logic of the 3600 I/O routine

Figure 99 shows the logic of the 3600 I/O routines. The following notes are keyed to this figure.

1 If the processor's request is for input, the information that must be passed to VTAM is set up and a RECEIVE is issued. The address of the session's

RPL is put in a register and the address of the input area associated with the session and the length of the area are put in other registers. Because data is to be read, RTYPE=DFSYN is specified. The operation is to be asynchronous and input is to be read only from the specific session (whose CID is located in the RPL's ARG field). The ECB associated with the session is specified for posting by VTAM when the operation is completed. After issuing the RECEIVE request, register 15 contains 0 if the request is accepted, or some other return code if it is not. If the request is accepted, the wait routine is returned to, after setting the next sequential instruction in this routine as the address to be branched to when the ECB is posted.

Note: For simplicity, most checks of register 15 are not shown in sample program 2.

2 When data is received from the LU, VTAM posts the ECB associated with the RECEIVE RPL. When the wait routine discovers the posted ECB, it branches to the indicated location in the 3600 I/O routine. The RESPOND field of the RPL can be tested to determine whether the LU wants a definite response returned to verify that the input was received. If so, a SEND is issued, indicating that a response is to be sent to the LU (STYPE=RESP).

The SEND that sends the response, if requested, is scheduled synchronously. VTAM assumes POST=SCHED. Because no response can be returned to a response, once the request to send the response is accepted, the VTAM application program considers the sending of the response as complete.

If input arrives unsuccessfully or out of sequence (indicating that some input was lost), VTAM completes the VTAM application program's input request with an indication that a negative response must be returned; no input is forwarded to the program. The application program sends the negative response. The input request can be reissued.

Although not shown, the VTAM application program can also return a negative response to input that is successfully received. This might be done when initial processing indicates an error in the format of the received input. Such a response is understood by both the application program and the LU. The SSENSEO, SSENSMO, and USENSEO fields of the RPL can be used to convey exception information.

3 If the input contains a request to log off, the session is ended by issuing a CLSDST macroinstruction, and the control blocks associated with the session are returned to the system or to a pool. Optionally, a request can be sent to the LU, confirming logoff, prior to issuing CLSDST.

The preceding description of the 3600 input routine assumes that a CHECK macroinstruction is issued in the wait routine upon completion of each requested input operation; if an error occurs, CHECK causes entry to the LERAD or SYNAD exit routine which takes appropriate action. This can include sending the negative response for step 2. The input routine can issue a request to receive any kind of input: a normal-flow data request or data-flow-control request (DFSYN), an expedited-flow data-flow-control request (DFASY), or a response (RESP). In this sample program, DFASY and RESP-type input are handled by VTAM-scheduled DFASY and RESP exit routines, but the logic in these routines could have been branched to after determining in the wait routine or 3600 input routine that DFASY or RESP information had been received. DFSYN means that either data or normal-flow data-flow-control requests can be received; although not

shown in this example, normal-flow data-flow-control requests such as Quiesce Complete (QC) might be receivable in some applications, in which case such requests have to be responded to.

4 When an output request from a processor is received by the 3600 output routine, the routine does not process the request if the logical unit has quiesced the VTAM application program; instead, the I/O routine branches to the wait routine. The processor must wait until the quiesce is released at which time the ECB for the session is posted, a pending send request detected, and the routine is reentered. This logic is discussed in "Logic of the DFASY exit routine of Sample Program 2" on page 624.

5 If the output request is chained to other output requests, a branch is made to a chaining output routine (see Figure 100 on page 618).

6 If output is not being chained, a SEND is issued that includes the operand CHAIN=ONLY. If the output completes a transaction, the session is returned to continue-any mode; its next input satisfies the RECEIVE OPTCD=ANY request issued in RPL1. The request can specify scheduling of the operation (POST=SCHED) with completion to be determined as the result of a positive or negative response (RESPOND=NEX) that causes scheduling of the RESP exit routine. (The RESP exit routine posts the ECB associated with the session, notifying the VTAM application program and the processor that the output request was completed.) The output routine then branches to the wait routine.

Logic of the 3600 chaining output routine

Figure 100 on page 618 shows the logic of the 3600 chaining output routine. The following notes are keyed to this figure.

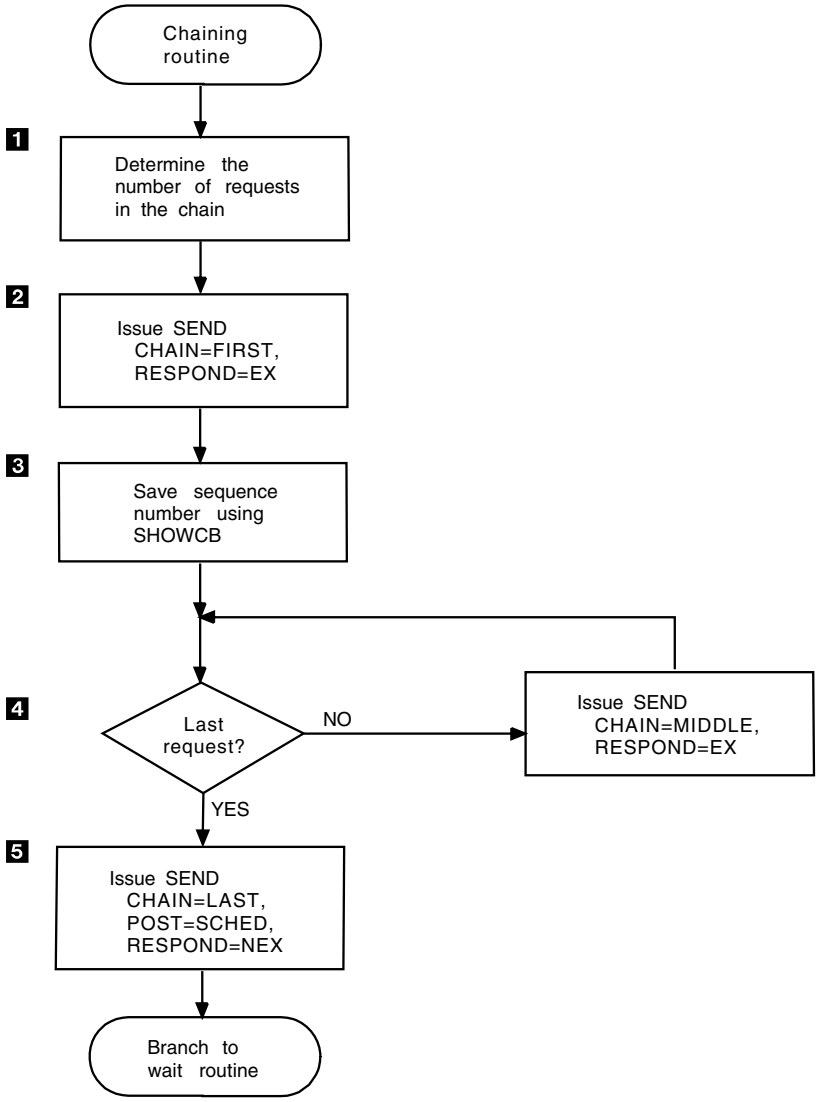


Figure 100. Logic of the chaining output routine

1 The number of requests in the chain might vary. Assuming that it varies in sample program 2, the number of chain requests must be determined so that the routine knows when to send the last request. It might be convenient to picture this routine being entered to send a report to an administrative line printer; this report can vary in length between 20 and 100 printer lines. Each line is sent to the 3601 LU as a chain request. The 3601 LU determines how many lines (chain requests) it collects before sending them on to the administrative printer.

This chaining routine can pass all of the data to be sent in a chain or only part of it. In other words, the routine is not necessarily sending an entire chain each time it is entered. The logic discussed here, however, assumes that all or, in a retry situation, the last part of a chain is being sent.

2 Because one of the advantages of chaining output is to reduce the number of required responses while still breaking output into requests that can be interspersed on the communication path, all SEND macroinstructions, other than the last one, specify that a response is to be returned only if an exception is noted (RESPOND=EX). When the last request is received, a

positive response is returned, and the VTAM application program recognizes that the entire chain arrived successfully. When RESPOND=EX is specified, the scheduling of output (POST=SCHED) is assumed by VTAM; it does not have to be specified.

3 In some cases, it might be necessary to save the sequence number of the first request sent in a chain. This number is available as soon as sending has been scheduled. It can be obtained from the SEQNO field of the RPL by using the SHOWCB macroinstruction. In case all or part of the chain must be resent (a negative response arrives in the RESP exit routine), the first-of-chain sequence number can be useful in determining where to start resending. It might also be necessary (not shown here) to reset the beginning sequence number for the session that is receiving the chain; this number is altered by using a SESSIONC macroinstruction. The sequence number can be saved in the control block associated with the session.

4 All requests except the first and last are middle requests (CHAIN=MIDDLE).

5 For the last request of the chain, the SEND macroinstruction must identify it as the last (CHAIN=LAST) and ask for the return of a response (RESPOND=NEX). Either POST=SCHED or POST=RESP can be specified.

When the response is received, if POST=SCHED was specified, the RESP exit routine posts an ECB, causing the wait routine to return to the processor that originated the output request. If POST=RESP was specified, VTAM posts an ECB or schedules an RPL exit routine.

Routine logic of the 3270 I/O routine

Figure 101 on page 620 shows the logic of sample program 2's 3270 I/O routine. With few exceptions, the VTAM application program need not distinguish between channel-attached non-SNA, BSC, and SDLC PU type 1 3270s. (All these terminals use LU type 0 protocols which are described in Chapter 11, “Programming for the IBM 3270 Information Display System,” on page 327.)

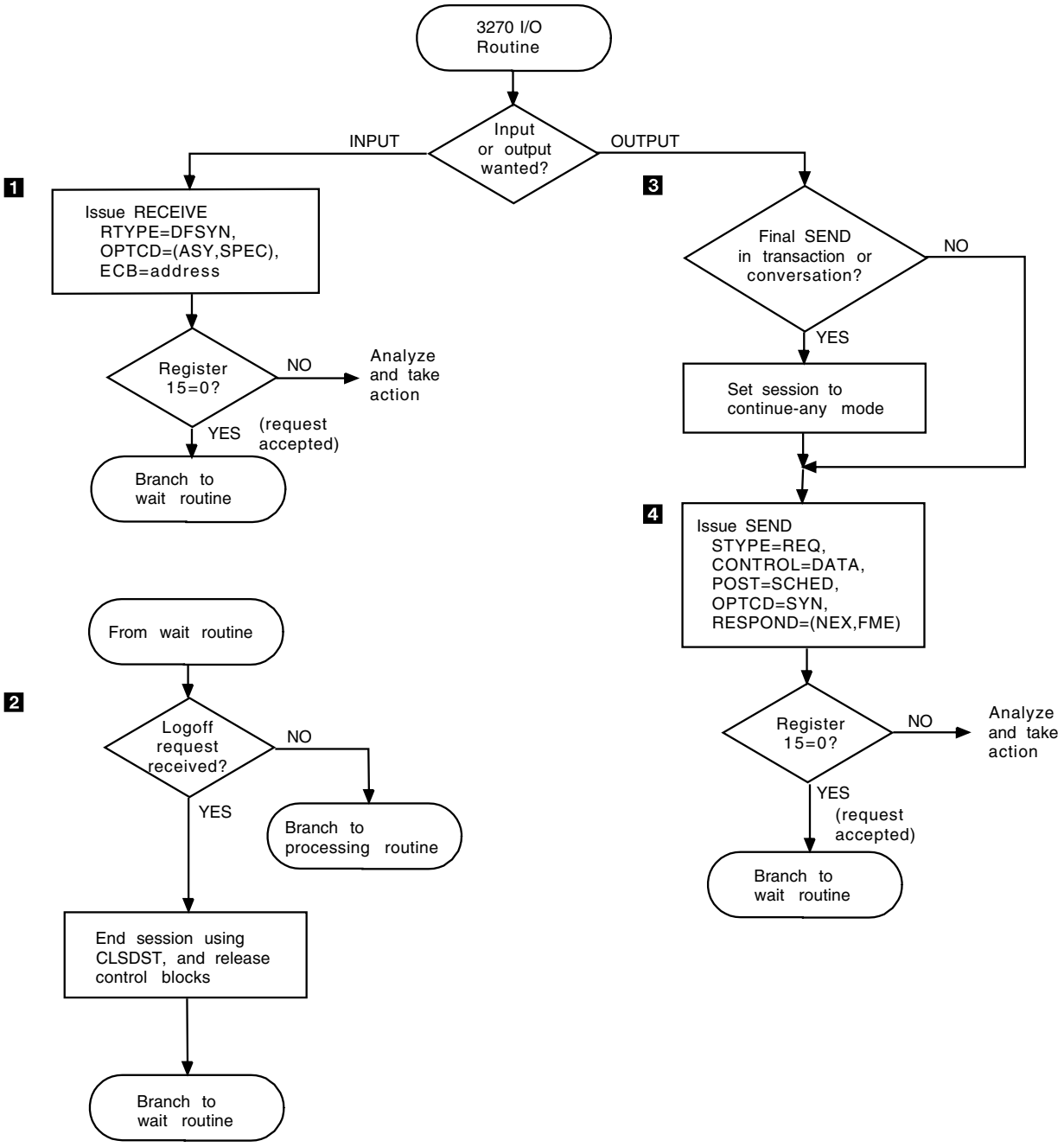


Figure 101. Logic of the 3270 I/O routine

Data received from a 3270 begins with an attention identifier (AID) character. Data sent to the 3270, whether local or remote, must begin with a 3270 command character, such as Erase, Erase and Write, or Erase All Unprotected; VTAM inserts an ESC character for BSC 3270s. The 3270 is different from other LUs in several ways, including the following:

- Because a 3270 does not contain an application program (a variable program), it cannot send control requests or responses. However, in some cases, VTAM provides responses to the VTAM application program on behalf of the 3270 as a result of receiving BSC responses to transmitted data or as a result of receiving indications that the 3270 is in a particular bracket state.

- The amount of data that can be sent to or received from the 3270 is limited by the physical characteristics of the 3270, whereas the amount of data that can be sent to or received from a 3601 is more indefinite.
- Chaining output to the LU type 0 3270 is not possible.
- Responses cannot be requested by LU type 0 3270 terminals.

The following notes are keyed to Figure 101 on page 620.

1 Except that the type and length of data can be different for a 3270 RECEIVE, this request is similar to step 1 discussed for the 3600 input routine.

2 This logic is similar to that of step 2 for the 3600 input routine except that, because the 3270 cannot request a response to input it has provided, no check is made to determine whether to send a response.

3 If 3270 output is requested by a processing routine, the 3270 I/O routine determines whether this output completes a transaction. If it does, the session is put back into continue-any mode so that the RECEIVE OPTCD=ANY specified in the RPL1 exit routine can receive input from this session when the terminal operator wishes to begin a new transaction.

4 A SEND macroinstruction is issued to send the output. (Although not shown, this routine might also have to determine from the processing-routine request what 3270 command character (for example, Erase and Write) is to precede the output data stream that the processing routine furnishes.) The sending of the output is scheduled synchronously (POST=SCHED, OPTCD=SYN); VTAM returns control after it has scheduled the output operation. A response is requested (RESPOND=(NEX,FME)) so that the VTAM application program can determine whether the operation was successful. The 3270 returns information enabling VTAM to provide the appropriate response in the RPL and to schedule the RESP exit routine. The RESP exit routine (Figure 102 on page 623) posts an ECB so that the mainline program's wait routine can determine that the operation completed, branching back to the processing routine that requested the output. An output request to a 3270 printer requires a definite response (RESPOND=(NEX,FME)); an output request to a display can specify either NEX or EX but cannot specify that neither a positive nor negative response is to be returned (RESPOND=(NEX,NFME)). After successfully scheduling output to the 3270, the 3270 output routine branches to the wait routine.

Logic of the RESP exit routine

Figure 102 on page 623 shows the logic of the RESP exit routine. This routine is entered when the response is received to an output request that has POST=SCHED specified in a 3600 or 3270 output routine. The output operations have been scheduled with responses to be returned by the LU so that completion of each operation can be determined. (It is also possible for all output operations to be specified with POST=RESP. In this case, the response is received by VTAM and its nature determined by the VTAM application program after ECB posting or RPL exit routine scheduling. No RESP exit routine is required.)

When the VTAM application program gets control in its RESP exit routine, a RECEIVE is not issued. The nature of the response is determined by examining the RESPOND and other fields of an RPL that are in VTAM's storage. The address of this RPL is in a parameter list whose address is in register 1 when the RESP exit

routine is entered. The session area (the ECB, RPL, and session-related control block) can be located by the address in the USER field of the VTAM RPL. (It contains whatever was placed in the USERFLD field of the NIB when the session was established.)

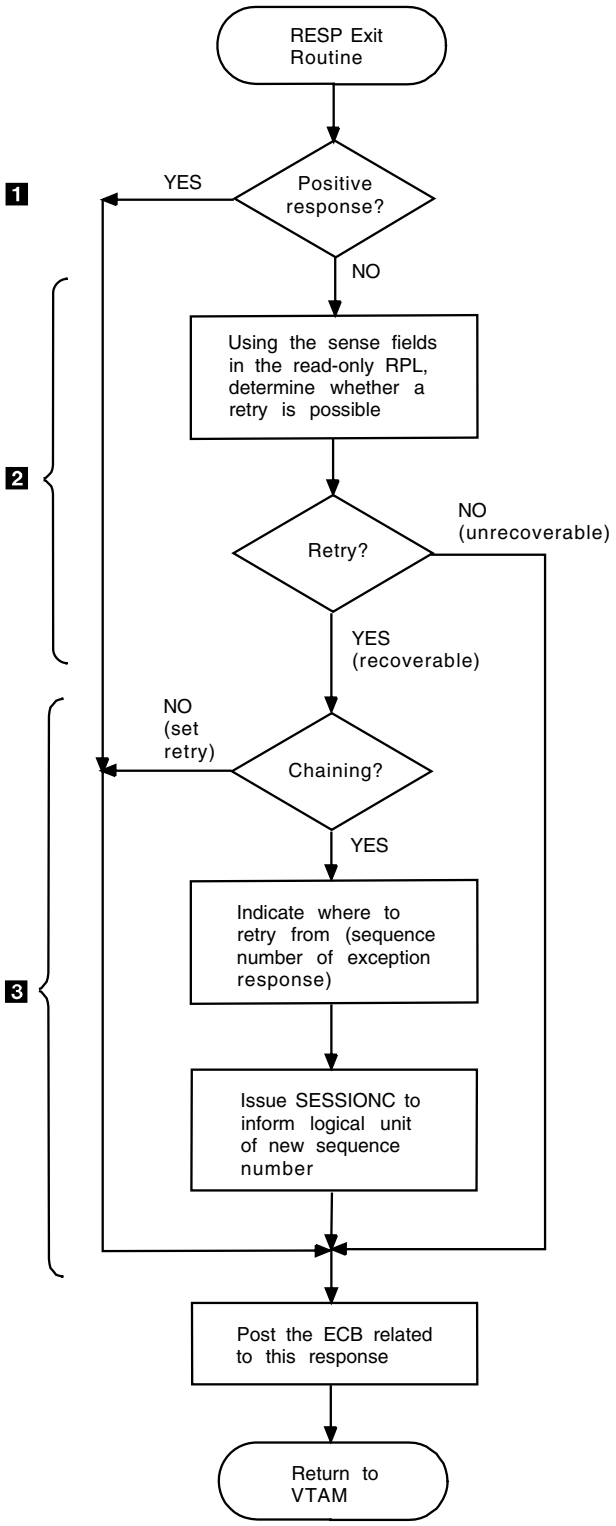


Figure 102. Logic of the RESP exit routine

The following notes are keyed to Figure 102.

1 If the response is positive, the appropriate ECB is posted and a return is made to VTAM. Even if other action must be taken because the response is negative, the ECB is posted so that the wait routine knows that the operation has been completed.

2 The RESP exit routine can use the SYNAD exit routine to analyze a negative response; if so, the user could set up the appropriate registers and branch directly to the SYNAD exit routine.

3 If the situation is defined by the SYNAD exit routine to be recoverable, the operation is retried. If it is part of a chaining operation, it might be necessary to save the sequence number of the output request to which a negative response was returned so that the chaining routine can determine the sequence number at which it starts a retry. If the session's inbound sequence number (outbound from the host) must be reset, a SESSIONC can be used to synchronize sequence numbers. This logic can also be in the SYNAD exit routine.

On completion, the RESP exit routine returns control to VTAM.

Logic of the DFASY exit routine of Sample Program 2

Figure 103 on page 625 shows the logic of the DFASY exit routine in sample program 2. The DFASY exit routine is entered when a request is received from the LU asking the program to quiesce (stop) sending to the LU or to resume sending, if sending was previously quiesced.

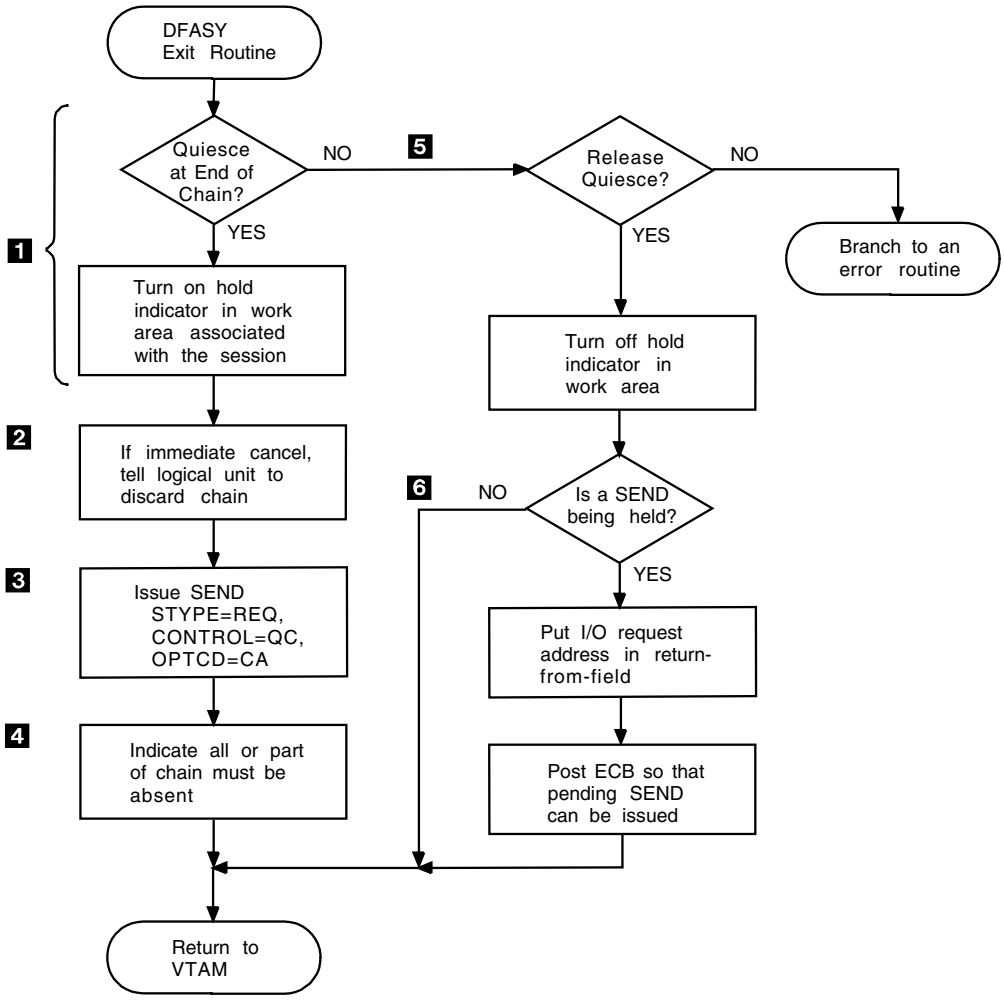


Figure 103. Logic of the DFASY exit routine

Quiescing can be done for two reasons:

- To ensure that, at a given time, only the LU or the VTAM application program can be sending. This use of quiescing is not demonstrated in this sample program. (Quiescing is only one means available to ensure that both sides do not send at the same time. The change-direction protocol can also be used.) In many cases, receiving a response ensures that both ends do not send at the same time.
- To interrupt a steady flow of input data so that an output operation can be performed. This is the use of quiescing that is demonstrated here. For example, a teller at a 3600 terminal might wish to temporarily interrupt a long printout so that an informational request (which does not require a reply) can be sent to the VTAM application program. As a result of a teller action, the 3601 LU for the teller's workstation sends a Quiesce-at-End-of-Chain request to the VTAM application program, which can then agree to stop sending and be ready to read input from the LU.

  The Quiesce-at-End-of-Chain and Release-Quiesce requests are sent as expedited-flow requests unaccompanied by data. VTAM schedules the VTAM application program's DFASY exit routine when one of these requests is received.

The following notes are keyed to Figure 103.

1 The type of request that caused the DFASY exit routine to be entered is available in the CONTROL field of the read-only RPL whose address is provided by VTAM on entry. If a Quiesce-at-End-of-Chain (QEC) request was received, this routine sets a hold indicator in the work area associated with the session. The session-related control block, as in the RESP exit routine, is located by the address in the USER field of the RPL.

2 If the quiesce is to be immediate, the exit routine can instruct the LU to discard the chain by issuing a SEND macroinstruction that specifies CONTROL=CANCEL. Alternatively, the next SEND would be set to CHAIN=LAST; the LU determines whether to use the chain requests previously received. If it is in the middle of a chain and not all of the chain is to be resent, the VTAM application program can note where sending is to resume when the quiesce condition is released.

3 The QEC request is acknowledged by sending back a Quiesce Complete (QC) request. So that the LU's input is able to complete the request to receive input from any session being recurrently issued in the RPL1 exit routine, the session is put back into continue-any mode (OPTCD=CA).

4 This flag might be required in addition to the hold indicator to determine where to resume sending. Refer to step 2.

5 If the request is a Release-Quiesce (RELQ) request, the hold indicator is turned off.

6 If further output is being held, the output routine is rescheduled for this session, and an ECB is posted so that the wait routine branches to it. Control is returned to VTAM.

Chapter 17. Sample code using authorized path

This sample program, SAMP3, shows an application program using the authorized path facility under the control of both a task control block (TCB) and a service request block (SRB). The logic for this sample application program is similar to the logic for sample program 2 in Chapter 16, "Logic of a more complicated application program," on page 605.

SAMP3 uses the authorized forms of the SEND, RECEIVE, and RESETSR macroinstructions to perform communication processing. SAMP3 shows:

- How to enter supervisor state
- Which operating system macroinstructions to use.

This sample can be used as a guideline for coding application programs that use the authorized path facility; the sample program is not intended to be coded and used as it is shown. For further details about authorized path, see "Authorized path" on page 300.

Notes on SAMP3

SAMP3 physically consists of one application program, but logically consists of two programs. The first logical program is labeled "AUTHPATH," and the second logical program is labeled "AUTHEXIT."

AUTHPATH begins by opening an ACB and establishing a session with an LU. In order to use the authorized forms of SEND, RECEIVE, RESETSR, or SESSIONC, the application program must have authorization to change from problem program state to supervisor state. See "Assigning operating system authorization" on page 296 for a description of how to become an authorized program.

SAMP3 changes into supervisor state by issuing the operating system macroinstruction MODESET. MODESET obtains the zero protection key needed to use authorized path. To schedule an SRB, an application program must be in supervisor state with a key of zero.

At this point, SAMP3 departs from normal VTAM application programming. It is now operating as a system program, using a system key. The application program issues the RECEIVE macroinstruction in its asynchronous form with an exit routine specified. The operand BRANCH=YES causes authorized path to be used. The exit routine runs under control of an SRB.

The exit routine, AUTHEXIT, issues the CHECK macroinstruction, which delimits processing of the RECEIVE macroinstruction issued in the mainline program. Then the input is tested for three possibilities:

- Echo
- No echo
- An error condition.

© Copyright IBM Corp. 2000, 2013

If an echo is desired, the SEND macroinstruction is used to return the data just received. This SEND uses authorized path because it is issued under control of the SRB under which the exit routine was scheduled. The mainline program is then posted to continue issuing the RECEIVE.

If no echo is desired, and an error has not been encountered, the RESETSR macroinstruction is issued to change the logical unit (LU) back to continue-any mode to allow it to be addressed by the next issuance of the RECEIVE macroinstruction. RESETSR uses authorized path because it is issued under control of the SRB under which the exit routine was scheduled.

If an error condition was encountered, the mainline program is posted to issue CLSDST and CLOSE ACB after offering the user the option of an ABEND dump.

Otherwise, the mainline program is posted to continue issuing the RECEIVE OPTCD=ANY to accept data from the LU. Because the exit routine runs under an SRB, the branch entry to POST, rather than the POST SVC, must be used, which in turn requires obtaining and releasing the local lock with the SETLOCK macroinstruction.

The mainline program checks the input for the string “LOGOFF” and, if it finds this input, issues CLSDST and CLOSE ACB to shut down the application program. (Notice that the CLSDST macroinstruction uses the BRANCH=NO operand to turn off the RPLBRANC flag turned on by the RECEIVE which used the BRANCH=YES operand.) If the “LOGOFF” string is not found, the mainline program continues issuing the RECEIVE and the exit again gains control.

SAMP3 assembler language code

```
AUTHPATH CSECT
         PRINT NOGEN
R15      EQU   15
R14      EQU   14
R13      EQU   13
R12      EQU   12
R11      EQU   11
R10      EQU   10
R9       EQU   9
R5       EQU   5
R4       EQU   4
R1       EQU   1
R0       EQU   0
         SAVE  (14,12),T,*
         LR    R12,R15           * GET BASE
         USING AUTHPATH,R12      * COVER
         LR    R9,R13            * MOVE SAVE PTR
         LA    R13,SAVE          * PT TO SAVE AREA
         ST    R13,8(0,R9)       * FORWARD PTR
         ST    R9,4(0,R13)       * BACKWARD PTR
         LA    R4,AUTHRPL        * BASE FOR RPL
         USING IFGRPL,R4         * COVER RPL
         WTO   'AUTHPATH APPLICATION ENTERED'  * TELL OPER WE'RE HERE
*
*        OPEN THE ACB
*
         OPEN  AUTHACB           * OPEN VTAM ACB
         MVI   THRU+3,4          * OUTPUT MSG 1, OPEN FAILED
         LTR   R15,R15           * GOOD
         BNZ   BAD               * NO
*
*        ESTABLISH SESSION
```

```
*
*
         OPNDST RPL=AUTHRPL,OPTCD=(SYN,ACQUIRE) * ESTABLISH SESSION
         MVI   THRU+3,8          * OUTPUT MSG 2, OPEN DEST FAILED
         LTR   R15,R15           * GOOD
*        BNZ   BAD               * NO
*
*        MODESET TO SUP STATE
*
         MODESET MODE=SUP        * SUPERVISOR STATE
*
*        ISSUE RECEIVE MACRO
*
AUTHRECV RECEIVE RPL=AUTHRPL,OPTCD=(ASY,Q,ANY,CS),BRANCH=YES,          C
               AREA=INPUT00,AREALEN=100,EXIT=AUTHEXIT,RTYPE=DFSYN
         MVI     THRU+3,12     * OUTPUT MSG 3, RECEIVE VALID CHECK FAIL
         LTR     R15,R15       * CHECK RETURN CODE FROM RECEIVE
         BNZ     BAD           * NOT 0, GO HANDLE SITUATION
*
*  HERE THIS APPLICATION COULD BE DOING SOME TYPE
*  OF PROCESSING BEFORE NOTIFICATION FROM AUTHEXIT.
*
*
         WAIT    1,ECB=THRU    * WAIT FOR AUTH EXIT
         CLI     THRU+3,0      * EVERYTHING OK IN EXIT?
         BNZ     BAD           * NO, HANDLE IT.
         CLC     LOGMSG,INPUT00+6 * USER WANT TO LOG OFF?
         BE      CLOSE1        * USER WANTS TO QUIT
         XC      THRU,THRU     * CLEAR ECB
         MVI     INPUT00,X'40'
         MVC     INPUT00+1(99),INPUT00 * CLEAR INPUT AREA
         B       AUTHRECV      * REPEAT RECEIVE
*
*        CLOSE SESSION AND ACB AND END PROGRAM
*
CLOSE1   CLSDST RPL=AUTHRPL,OPTCD=SYN,BRANCH=NO * TERMINATE SESSION
CLOSE2   CLOSE   AUTHACB       * CLOSE ACB
CLOSE3   LA      R1,MSG0       * ISSUE ENDED MESSAGE
         WTO     MF=(E,(1))
         L       R13,4(0,R13)  * POINT
         RETURN  (14,12),T,RC=0
BAD      L       R5,THRU       * USE POST CODE AS INDEX
         L       R1,MSGS(R5)   * PT TO APPROPRIATE MSG
         WTO     MF=(E,(1))
         WTOR    'ENTER 'Y'' FOR ABEND DUMP',REPLY,1,WTORECB
         WAIT    1,ECB=WTORECB
         CLI     REPLY,C'Y'    * DOES USER WANT A DUMP?
         BNE     CHEKCLOS      * DETERMINE PROPER CLOSE
         ABEND   100,DUMP,STEP * HALT PROGRAM, DUMP
CHEKCLOS CLI THRU+3,8          * CHECK CLOSING CODE
         BH      CLOSE1        * CLOSE EVERYTHING
         BE      CLOSE2        * JUST CLOSE ACB
         BL      CLOSE3        * CLOSE FAILED
AUTHEXIT DS      0H
         USING   AUTHEXIT,R15  * TEMPORARY BASE
         STM     14,12,SAV2+12 * SAVE CALLERS REGS
         LR      R12,R15       * LOAD BASE
         DROP    15
         USING   AUTHEXIT,R12
         LA      R13,SAV3      * VTAM DOES NOT PASS A SAVE AREA
*                               * SO SKIP STANDARD LINKAGE
         CHECK   RPL=AUTHRPL   * CHECK STATUS OF REQUEST
         MVI     POSTCODE+3,16 * SET CODE NOT EQUAL 0
         LTR     R15,R15       * RETURN CODE EQUAL 0?
         BNZ     GOPOST        * NO, GO HANDLE SITUATION
*
*        THE SEND BELOW USES THE AUTHORIZED PATH BECAUSE
```

```
*        IT IS ISSUED UNDER CONTROL OF THE SRB SCHEDULED BY
*        VTAM AS THE EXIT FOR THE AUTHORIZED PATH RECEIVE.
*
         MVC     OUTPUT01,INPUT00 * MOVE WHAT WE READ TO OUTPUT AREA
         TM      INPUT00,X'10' * ERROR RESPONSE REQUESTED?
         BC      1,CLOSIT      * YES, CLOSE DEST AND QUIT
         TM      INPUT00,X'04' * DOES APB WANT AN ECHO?
         BC      8,NOECHO      * NO
*
* ECHO WAS REQUESTED
*
         MVI     OUTPUT01,X'82' * SET ECHO RESPONSE CODE
         XC      OUTPUT01+1(5),OUTPUT01+1 * CLEAR OTHER CONTROL
         CLC     LOGMSG,INPUT00+6 * USER WANT TO LOGOFF?
         BNE     CONTINUE       * NO
         MVC     OUTPUT01+13(35),OUTPUT02 * MOVE IN FINAL MSG
CONTINUE DS      0H
*
         SEND    OPTCD=(SYN,CA),CONTROL=DATA,STYPE=REQ,RTYPE=DFSYN,    C
               RECLEN=100,AREA=OUTPUT01,RPL=AUTHRPL,POST=SCHED,        C
               RESPOND=(NEX,NFME,NRRN) * SEND SOME DATA OUT
         MVI     POSTCODE+3,20  * SET UP MSG5, SEND VALIDITY FAILED
         LTR     R15,R15        * RC=0?
         BNZ     GOPOST         * NO, POST MAINLINE
         MVI     POSTCODE+3,24  * SET MSG6- RPL RETURN CODE
         CLI     RPLRTNCD,0     * RETURN CODE OF ZERO?
         BNE     GOPOST         * POST MAINLINE
         CLI     RPLFDB2,0      * CHECK FEEDBACK CODE
         BNE     GOPOST         * POST MAINLINE
AOK      XC      POSTCODE,POSTCODE * CLEAR POST CODE
*
*        BRANCH ENTRY TO POST REQUIRES LOCAL LOCK
*
GOPOST   DS      0H
LOCK     SETLOCK OBTAIN,TYPE=LOCAL,MODE=UNCOND,REGS=USE,               C
               RELATED=(LOCAL,AUTHPATH(UNLOCK))
         L       R10,POSTCODE   * GET POST CODE
         LA      R11,THRU       * POINT TO ECB
         L       R15,16         * CVT PTR
         L       R15,(CVT0PT02-CVT)(0,R15) * BRANCH ENTRY TO POST
         BALR    R14,R15        * POST ECB
UNLOCK   SETLOCK RELEASE,TYPE=LOCAL,REGS=USE,                          C
               RELATED=(LOCAL,AUTHPATH(LOCK))
AUTHEX0  L       R13,4(0,R13)   * PT TO SAVE AREA
         RETURN  (14,12),T,RC=0
NOECHO   DS      0H             * APB DOES NOT WANT AN ECHO
*  MUST ISSUE RESETSR TO SWAP MODE BACK TO CONTINUE ANY
         RESETSR RPL=AUTHRPL,OPTCD=(SYN,CA),RTYPE=DFSYN
         MVI     POSTCODE+3,28  * MESSAGE 7
         LTR     R15,R15        * RESET OK?
         BNZ     GOPOST         * NO
         B       AOK            * OK, CONTINUE
CLOSIT   DS      0H             * ERROR RESPONSE- CLOSE DEST
         MVI     POSTCODE+3,32  * SET SPECIAL CLOSE DEST CODE
         B       GOPOST         * POST MAINLINE
THRU     DC      F'0'           * ECB TO WAIT ON SRB
WTORECB  DC      F'0'           * ECB FOR WTOR
SYSDATE  DC      CL8'&SYSDATE'  * DATE OF ASSEMBLY
SAVE     DC      18F'0'
SAV2     DS      0F
         DC      2F'0'
         DC      A(SAV3)
         DC      15F'0'
SAV3     DS      0F
         DC      F'0'
         DC      A(SAV2)
         DC      16F'0'
```

```
REPLY    DC      C' '           * USER'S REPLY TO DUMP MSG
LOGMSG   DC      CL7'LOGOFF '   * USER WANTS TO LOGOFF
POSTCODE DC      F'0'           * POST CODE GOES HERE
OUTPUT01 DC      CL100' '
INPUT00  DC      CL100' '
OUTPUT02 DC      CL35' *** AUTHPATH WORKS, PASS IT ON ***'
APPL5ID  DC      X'05',CL5'APPL5' * ID AND PASSWORD
MSGS     DC      A(MSG0,MSG1,MSG2,MSG3,MSG4,MSG5,MSG6,MSG7,MSG8)
MSG0     WTO  'AUTHPATH ENDED',ROUTCDE=(1),DESC=(5),MF=L
MSG1     WTO  'OPEN ACB FAILED',ROUTCDE=(1),DESC=(5),MF=L
MSG2     WTO  'OPEN DESTINATION FAILED',ROUTCDE=(1),DESC=(5),MF=L
MSG3     WTO  'RECEIVE VALIDITY CHECK FAILED',ROUTCDE=(1),DESC=(5),MF=L
MSG4     WTO  'RECEIVE FAILED, EXIT ENTERED',ROUTCDE=(1),DESC=(5),MF=L
MSG5     WTO  'SEND VALIDITY CHECK FAILED',ROUTCDE=(1),DESC=(5),MF=L
MSG6     WTO  'SEND FAILED',ROUTCDE=(1),DESC=(5),MF=L
MSG7     WTO  'RESETSR FAILED',ROUTCDE=(1),DESC=(5),MF=L
MSG8     WTO  'APB REQUESTED CLOSE DEST',ROUTCDE=(1),DESC=(5),MF=L
         DS     0H
PATCH    DC     10CL8'PATCHES'
AUTHRPL  RPL    ACB=AUTHACB,AM=VTAM,NIB=AUTHNIB2
AUTHACB  ACB    AM=VTAM,APPLID=APPL5ID,PASSWD=APPL5ID,MACRF=NLOGON
AUTHNIB2 NIB    NAME=CTJ10LU1,USERFLD=C' NEW',LISTEND=YES
         IFGRPL   AM=VTAM
         IHASRB
         IHAPSA
         IHAFRRS
         CVT      DSECT=YES,LIST=YES
         END
```

Appendix A. Summary of control block field usage

This appendix serves as a reference for the experienced VTAM application programmer by showing the following information for each executable macroinstruction discussed in this book:

- The control block fields set by the application program when (or before) the macroinstruction is used.
- The control block fields and registers set by VTAM upon completion of the processing started by the macroinstruction.

Note: All of the control block fields that apply to each macroinstruction are shown, but remember that not all fields apply to every possible variation of a macroinstruction. See Chapter 13, "Conventions and descriptions of VTAM macroinstructions," on page 371, for details about each macroinstruction.

For declarative macroinstructions, the macroinstruction operands and defaults are shown.

Throughout this appendix, a pointer (→) indicates that a field contains the address of the given item, and an equal sign (=) indicates that a field contains the item itself.

Mutually exclusive fields are indicated as such when they occur in a macroinstruction. For example, NIB and ARG are mutually exclusive in the CLSDST macroinstruction; only one of the two can be used.

Indented fields are associated with the field preceding them. For example, the NAME, CID, USERFLD, LOGMODE, and LISTEND fields are all associated with the NIB field.

Fields that provide information supplied by your program rather than information provided by the macroinstruction are indicated by bold type. All information preceding the bold type concerns information that your application program supplies to the macroinstruction. All information shown in bold type is information that the macroinstruction passes back to your application program.

ACB

```
ACB     AM operand = VTAM
        APPLID field →      application program's symbolic name
        PASSWD field →      password
        EXLST field →       exit list
        MACRF field =       LOGON|NLOGON
        PARMS field =       (NIB subfield → NIB USERFLD
                            subfield = userdata)
        PARMS field =       (APPLVCTR → vector list address)
        PARMS=(KEEPFRR =    YES|NO)
        PARMS=(PERSIST =    YES|NO)
        PARMS=(FORCETKO =   YES|NO)
        PARMS=(SRBEXIT =    YES|NO)
        PARMS=(NQNAMES =    YES|NO)
        PARMS=(FDX =        YES|NO)
        PARMS=(PERFMON =    YES|NO)
```

5. Applies only to communication network management application program.

© Copyright IBM Corp. 2000, 2013

CHANGE

```
CHANGE →     RPL:     ACB field → ACB
                      NIB field → NIB:
                           NAME field = [symbolic name of LU whose
                                        association is terminated|*]
                           GNAME field = generic name
                           NETID field = symbolic network identifier of LU
                           LISTEND field = YES
                      Note: ECB and EXIT are mutually exclusive.
                      If ECB is used, only one is allowed.
                      ECB field → external ECB
                      ECB field = internal ECB
                      EXIT field →  RPL exit routine
                      BRANCH field = YES|NO
                      OPTCD field = (ENDAFFIN|ENDAFFNF,
                                     SYN|ASY)

     Registers 0 and 15 = return codes

     RPL:             RTNCD field = recovery action return code
                      FDB2 field = specific error return code
                      REQ field = request code (25)
```

CHECK

```
CHECK →              RPL being checked
                     RPL set inactive
                     ECB cleared

                     Register 0 and 15 = return codes
```

CLOSE

```
CLOSE →              ACB being closed
                     MF operand = list or execute form parameters

                     Register 15 = return code

                     ACB:     OFLAGS field = opened or not-opened
                                             indicator
                              ERROR field = specific error status
                                             information
```

CLSDST

```
CLSDST →     RPL:    ACB field → ACB
                     Note: ARG and NIB are mutually exclusive.
                     ARG field = CID of session terminated
                     NIB field → NIB:
                          Note: NAME and CID
                          are mutually exclusive.
                          NAME field = symbolic name of LU
                                       whose session
                                       is terminated
                          NETID field = symbolic network identifier of LU
                     Note: NETID and MODE
                     are mutually exclusive.
                          CID field = CID of session terminated
                          USERFLD field =correlator returned
                                                in NSEXIT exit routine
                          LOGMODE field = logon mode name
```

6. Applies only to CLSDST=PASS.

```
                              LISTEND field = YES
                         Note: MTSAREA and BNDAREA
                         are mutually exclusive.
                              MTSAREA field = 0|MTS area address
                              BNDAREA field → 0|BIND area
                         Note: ECB and EXIT are mutually exclusive.
                         If ECB is used, only one is allowed.
                         ECB field → external ECB
                         ECB field = internal ECB
                         EXIT field →  RPL exit routine
                         BRANCH field = YES|NO
                         AAREA field  → name of target PLU
                         AREA field  = user data for Initiate
                         RECLEN field  → length of user data
                         OPTCD field = (PASS|RELEASE|TERMQ,
                                        SYN|ASY,
                                        NSENSE|SENSE,
                                        MTS|NMTS)
                         PARMS field =(THRDPTY=NOTIFY|NONOTIFY)
                         PARMS = (SONCODE = code)
                         SSENSMO field = CPM|STATE|FI|RR|0
                         SSENSMO field = system-sense modifier value (or 0)
                         USENSEO field = user-sense value (or 0)
```

Registers 0 and 15 = return codes

RPL: **RTNCD field = recovery action return code**
FDB2 field = specific error return code
REQ field = request code (31)
SSENSEI field = CPM|STATE|FI|RR|PATH|0
SSEMSMI field = system-sense modifier value (or 0)
USENSEI field = user-sense value (or 0)

EXECRPL

```
EXECRPL →     RPL:     All fields appropriate for the request type
                       (indicated in the REQ field) are valid.
```

EXLST

```
EXLST     AM operand = VTAM
          LERAD field          → LERAD exit routine
          SYNAD field          → SYNAD exit routine
          DFASY field          → DFASY exit routine
          RESP field           → RESP exit routine
          SCIP field           → SCIP exit routine
          TPEND field          → TPEND exit routine
          RELREQ field         → RELREQ exit routine
          LOGON field          → LOGON exit routine
          LOSTERM field        → LOSTERM exit routine
          NSEXIT field         → NSEXIT exit routine
```

GENCB

```
GENCB     AM operand = VTAM
          BLK operand = control block type (ACB|EXLST|RPL|NIB)
          control block field name operand = value set in field
          COPIES operand = number of copies desired
          WAREA operand → work area where blocks are built
          LENGTH operand = length of work area
          MF operand = list, generate, or execute form parameters

          Register 0 = error return code (if register 15 indicates an error)
          or
          Register 0 = length of generated control blocks (if register 15
```

```
                    indicates successful completion)
               Register 1 → generated control blocks (if register 15
                    indicates successful completion)
               Register 15 = general return code
```

INQUIRE

```
INQUIRE →     RPL:     ACB field → ACB
                       Note: ECB and EXIT are mutually exclusive.
                       If ECB is used, only one is allowed.
                       ECB field → external ECB
                       ECB field = internal ECB
                       EXIT field → RPL exit routine
                       BRANCH field = YES|NO
                       OPTCD field = (LOGONMSG|DEVCHAR|COUNTS|
                                      TERMS|APPSTAT|NQN|CIDXLATE|
                                      TOPLOGON|SESSPARM|SESSKEY|
                                      USERVAR|PERSESS|SESSNAME|
                                      STATUS|SYN|ASY)

                            If OPTCD = LOGONMSG
                                 Note: ARG and NIB are mutually exclusive.
                                 ARG field = CID of pending active session
                                 NIB field → NIB:
                                      Note: NAME and CID
                                      are mutually exclusive.
                                      NAME field = symbolic name of LU
                                      NETID field = symbolic network identifier of LU
                                      CID field = CID of pending active session
                                      LISTEND field = YES
                                 AREA field → input area for user data from CINIT
                                 AREALEN field = length of input area

                       If OPTCD = DEVCHAR
                                 Note: ARG and NIB are mutually exclusive.
                                 ARG field = CID of session or pending
                                             active session
                                 NIB field → NIB:
                                      Note: NAME and CID
                                      are mutually exclusive.
                                      NAME field = symbolic name of
                                                   logical unit
                                      NETID field = symbolic network identifier of
                                                   logical unit
                                      CID field = CID of session or pending
                                                   active session
                                      LISTEND field = YES
                                 AREA FIELD → input are for LU characteristics
                                 AREALEN field = 8

                       If OPTCD = TERMS
                                 NIB field → NIB:
                                      NAME field = symbolic name of resource
                                                   in VTAM configuration tables
                                      NETID field = symbolic network identifier of resource
                                                   in VTAM configuration tables
                                      LISTEND field = YES
                                 AREA field → work area where NIBs are built
                                 AREALEN field = length of work area

                       If OPTCD = COUNTS
                                 AREA field → input area for data
                                 AREALEN field = 4

                       If OPTCD = APPSTAT
                                 NIB field → NIB:
                                      NAME field = symbolic name of
```

```
                         application program
                    NETID field = symbolic network identifier of
                         application program
                    LISTEND field = YES
               AREA field → network identifier and the real name
                    for specified application program
               AREALEN field = length of output area

If OPTCD = NQN
               NIB field → NIB:
                    NAME field = symbolic name
                    NETID field = symbolic network identifier
                    LISTEND field = YES
               AREA field → output name (based on type of
                    translation request)
               AREALEN field = length of output area

If OPTCD = CIDXLATE
               Note: ARG and NIB are mutually exclusive.
               ARG field = CID to be translated
               NIB field → NIB:
                    NAME field = symbolic name of LU
                    NETID field = symbolic network identifier of LU
                    LISTEND field = YES
               AREA field → input area of symbolic name
               AREALEN field = 8 for CID-to-name-translate
                             = 4 for name-to-CID translate

If OPTCD = TOPLOGON
               AREA field → input area for symbolic name
               AREALEN field = 8

If OPTCD = SESSNAME
               NIB field → NIB:
                    NAME field = symbolic name of LU
                    NETID field = symbolic network identifier of LU
                    GNAME field = generic name of application
                    LISTEND field = YES
               AREA field → area for application network name
               AREALEN field = 16

If OPTCD = SESSPARM
               Note: ARG and NIB are mutually exclusive.
               ARG field = CID of pending active session
               NIB field → NIB:
                    Note: NAME and CID
                    are mutually exclusive.
                    NAME field = symbolic name of LU
                    NETID field = symbolic network identifier of LU
                    CID field = CID of pending active session
                    LOGMODE field = |0 C'|'logon mode name
                    LISTEND field = YES
               AREA field → input area for session parameter
               AREALEN field = length of input area

If OPTCD = STATUS
               NIB field → NIB:
                    NAME field = symbolic name of LU
                    NETID field = symbolic network identifier of LU
                    LISTEND field = YES
               AREA field → network identifier and the real name
                         for specified LU
               AREALEN field = length of output area

If OPTCD = SESSKEY
               Note: ARG and NIB are mutually exclusive.
               ARG field = CID of session
```

```
                         NIB field → NIB:
                              Note: NAME and CID
                              are mutually exclusive.
                              NAME field = symbolic name of LU
                              NETID field = symbolic network identifier of LU
                              CID field = CID of session
                              LISTEND field = YES
                         AREA field → input area for session
                                   cryptography key and initial
                                   chaining value
                         AREALEN field = 16
                If OPTCD = PERSESS
                         AREA field → address of recovery data
                         AREALEN field = length of recovery data


     Registers 0 and 15 = return codes

     RPL:
          RTNCD field = recovery action return code
          FDB2 field = specific error return code
          REQ field = request code (26)
          SSENSEI field = CPM|STATE|FI|RR|PATH|0
          SSENSMI field = system-sense modifier value
                          (or 0)
          USENSEI field = user-sense value (or 0)
          RECLEN field = length of data received (if
                         (RTNCD,FDB2) = (X'00',X'05'),
                         RECLEN = total required length)
                         RECLEN = total required length
                         to identify one session
                         if OPTCD = PERSESS
          FDBK field = status information if
                         OPTCD = APPSTAT or OPTCD = STATUS
          ARG field = CID of oldest pending active
                         session if OPTCD = TOPLOGON
```

INTRPRET

```
INTRPRET →      RPL:     ACB field → ACB
                Note: ARG and NIB are mutually exclusive.
                ARG field = CID of session
                NIB field → NIB:
                     NAME field = symbolic name of LU
                     NETID field = symbolic network identifier of LU
                     LISTEND field = YES
                Note: ECB and EXIT are mutually exclusive.
                If ECB is used, only one is allowed.
                ECB field → external ECB
                ECB field = internal ECB
                EXIT field →  RPL exit routine
                BRANCH field = YES|NO
                AREA field → data to be interpreted
                RECLEN field = length of data to be interpreted
                AAREA field → work area for interpreted data
                AAREALN field = 8 or larger number
                OPTCD field = SYN|ASY

     Registers 0 and 15 = return codes

     RPL:
               RTNCD field = recovery action return code
               FDB2 field = specific error return code
               REQ field = request code (27)
               SSENSEI field = CPM|STATE|FI|RR|PATH|0
               SSENSMI field = system-sense modifier value (or 0)
               USENSEI field = user-sense value (or 0)
```

```
RECLEN field = length of data received (if
                (RTNCD,FDB2) = (X'00',X'05'),
                RECLEN = total required length)
```

ISTGLBAL

```
&ISTGLRL      = release-level macro global character variable
                  (product, version, release, modification code)
&ISTGLxy      = function-list macro global binary variables:
&ISTGL00      = NIB ENCR and RPL CRYPT (cryptography)
&ISTGL01      = ACB PARMS = (NIB=nib address)
                  (communication network management interface)
&ISTGL02      = Multiple-address-space application programs
&ISTGL03      = Authorized path for communication macros
&ISTGL04      = Authorized path for all RPL-based macros
&ISTGL05      = SRBEXIT (on APPL definition statement)
&ISTGL06      = SONSCIP (on APPL definition statement)
&ISTGL07      = VTAMFRR (on APPL definition statement)
&ISTGL10      = SSCP tracking of device-LU session capability by
                  means of NOTIFY (enabled/disabled/inhibited)
&ISTGL11      = RPL OPTCD=LMPEO
&ISTGL12      = RPL OPTCD=BUFFLST
&ISTGL13      = RPL OPTCD=USERRH
&ISTGL14      = ACB PARMS=(USERFLD=user data)
&ISTGL15      = RPL BRACKET=CEB
&ISTGL16      = Application program assignment of sequence
                  numbers for expedited DFC requests
&ISTGL17      = Resource-information vector list
&ISTGL20      = Access-method support vector list
&ISTGL21      = Return of system-response byte and extended-
                  response byte for BSC 3270 terminals attached
                  to NCP
&ISTGL22      = INTRPRET
&ISTGL23      = API is XRF-capable
&ISTGL24      = Sense included with a negative response to CINIT
                  using CLSDST OPTCD=(RELEASE,SENSE)
&ISTGL25      = Session outage notification code and sense included
                  with UNBIND
&ISTGL26      = Control the scheduling of the LOGON and SCIP
                exit routines to process session setup requests
&ISTGL27      = CINIT network addressed in vector key X'15'
&ISTGL30      = 31-bit application program
&ISTGL32      = LU 6.2 is supported
&ISTGL33      = Support for OPTCD=USERVAR for INQUIRE command
&ISTGL34      = Support for VCNS
&ISTGL35      = LAN support through the IBM 3172 Interconnect
                Controller for token-bus, token-ring,
                CSMA/CD, and FDDI
&ISTGL36      = Cross-memory API is supported
&ISTGL37      = VTAM maintains FRR stack for
                application-dispatchable unit of work
                while VTAM processes the work
                (KEEPFRR on ACB)
&ISTGL40      = Application program can use SRB processing
                in exit routines (SRBEXIT on ACB)
&ISTGL41      = Persistent LU-LU sessions supported
                (PERSIST on ACB)
&ISTGL43      = Enhanced data collection for NPM
&ISTGL44      = Persistent LU-LU sessions tracking
                supported for LU 6.1 and LU 6.2
&ISTGL45      = Reserved
&ISTGL46      = Reserved
&ISTGL47      = Network-qualified names are supported
&ISTGL50      = MS Transport is supported
&ISTGL51      = Performance monitor interface is supported
&ISTGL52      = QUEUED session termination supported (CLSDST
                or TERMSESS OPTCD=(TERMQ))
```

```
&ISTGL53      = CMIP services is supported
&ISTGL54      = Generic resources supported
&ISTGL55      = KEEPSRB supported
&ISTGL56      = Application vectors supported on ACB macroinstruction
&ISTGL57      = SETLOGON GNAME supported
&ISTGL60      = FORCETKO support
&ISTGL61      = User CV support
```

MODCB

```
MODCB     AM operand = VTAM
          control block type operand → control block
          control block field name operand = new value to be set
          MF operand = list, generate or execute form parameters

          Register 0 = error return code (if register 15 indicates an error)
          Register 15 = general return code
```

NIB

```
NIB       Name field = LU name
          AFFIN field = APPL|VTAM
          LUAFFIN field = APPL|NOTAPPL
          USERFLD field = user data
          LISTEND field = YES|NO
          NAME  field=[session parameter name|*]
          NETID field = LU network identifier
          SDT field = APPL|SYSTEM
          EXLST field → exit list
          ENCR field =REQD|SEL|NONE
          RESPLIM field = 1|response limit
          LOGMODE field = 0|C''|logon mode name
          ASDAREA field → dial parameter list address
          BNDAREA field  → 0|BIND area
          MTSAREA field  → 0|MTS area address
          PROC field = (CA|CS|CONDCS|RPLC,
                        NDFASYX|DFASYX,
                        NRESPX|RESPX,
                        NCONFTXT|CONFTXT,
                        KEEP|TRUNC,
                        SYSRESP|APPLRESP,
                        STOKEN ,
                        ORDRESP|NORDRESP,
                        NEGBIND|NNEGBIND)
          GNAME field = generic name of application
```

OPEN

```
OPEN →    ACB being opened
          MF operand = list or execute form parameters

          Register 15 =return code

          ACB:     OFLAGS field = opened or not-opened indicator
                   ERROR field = specific error status information
                   ACBAMSVL field [10]→access-method-support vector list
                   ACBPSINS field = persistent-instance bit
```

7. The NAME and PROC=STOKEN specifications are mutually exclusive.

8. Applies to the encryption facility only.

9. The MTSAREA and BNDAREA parameters are mutually exclusive.

10. There is no ACB operand for this field.

ACBPLUSC field = persistent capable application bit
ACBFRCTO field = MNPS forced takeover capable bit
ACBRIVL field [10]=resource information vector list

OPNDST

```
OPNDST →     RPL:     ACB field → ACB
                      NIB field → NIB|NIB list
                           Note: NAME and CID
                           are mutually exclusive.
                           NAME field: see following discussion of OPTCD
                           NETID field: see following discussion of OPTCD
                           CID field: see following discussion of OPTCD
                           LUAFFIN field: see following discussion of OPTCD
                      USERFLD field = data to be returned at the
                           completion of subsequent
                           RPL-based requests
                      EXLST field → exit routine list
                      ENCR field = level of cryptography required
                           (REQD|SEL|NONE)
                      SDT field = sender of SDT requests
                      PROC field = processing options
                           Note: LOGMODE and BNDAREA
                           are mutually exclusive.
                      LOGMODE field = 0|C''|logon mode name
                      BNDAREA field → 0|BIND area containing
                            session parameter
                      LISTEND field: see following discussion of OPTCD
                      Note: ECB and EXIT are mutually exclusive.
                      If ECB is used, only one is allowed.
                      ECB field → external ECB
                      ECB field = internal ECB
                      EXIT field → RPL exit routine
                      BRANCH field = YES|NO
                      OPTCD field = (SPEC|ANY,
                             SYN|ASY,
                             CS|CA,
                             Q|NQ,
                             CONANY|CONALL,
                             ACQUIRE|ACCEPT|RESTORE,
                             BACKUP|NBACKUP)

                 If OPTCD = ACQUIRE, then for NIB:
                       NAME field = symbolic name of LU
                       NETID field = symbolic network identifier of LU
                       LUAFFIN field = APPL|NOTAPPL
                       LISTEND field = YES|NO

                 If OPTCD = (ACCEPT,SPEC), then for NIB:
                       Note: NAME and CID
                       are mutually exclusive.
                       NAME field = symbolic name of SLU in
                              pending active session
                       NETID field = symbolic network identifier of SLU in
                              pending active session
                       CID field = CID of pending active session
                       LUAFFIN field = APPL|NOTAPPL
                       LISTEND field = YES

                 If OPTCD = (ACCEPT,ANY), then for NIB:
                       NAME field not examined
                       LUAFFIN field = APPL|NOTAPPL
                       LISTEND field = YES

                 If OPTCD = RESTORE, then for NIB:
                       NAME field = symbolic name of partner LU in the
                              session pending recovery
```

```
                    NETID field = symbolic network identifier of partner LU in the
                          session pending recovery
                    CID field = CID of the session pending recovery
                    LISTEND field = YES|NO
                    AAREA field → address of recovery data
                    AAREALN field = length of recovery data

              If NIB PROC = NEGBIND, then for RPL:
                    AAREA field → area to place negotiable
                          BIND response
                    AAREALN field = length of AAREA

              Registers 0 and 15 = return codes

             RPL:   RTNCD field = recovery action return code
                    FDB2 field = specific error return code
                    REQ field = request code (23)
                    SSENSEI field = CPM|STATE|FI|RR|PATH|0
                    SSENSMI field = system-sense modifier value (or 0)
                    USENSEI field = user-sense value (or 0)
                    ARG field = CID of session established in effect
                          value is unpredictable if CONALL is in effect
                          and if more than one session is established)
                    AREA field → NIB:
                          CID field = CID of session established
                          NAME field = symbolic name of LU in
                           session (when ACCEPT in effect)
                          NETID field = symbolic network identifier of LU in
                           session (when ACCEPT in effect)
                          DEVCHAR field = LU characteristics
                          NIBNACLQ field = statue of queues CINIT
                          CON field = YES|NO (session established or not)
                          NIBRPARM field = pointer to restore parameter list
                          (if OPTCD=RESTORE)
                          NIBPSPLU field = application type (1=PLU, 0=SLU)
                          (if OPTCD = RESTORE)
                          NIBPSRSP field = RESP Data Mode (0=Continue Any,
                          1=Continue Specific) (if OPTCD = RESTORE)
                          NIBPSDFS field = DFSYN Data Mode (0=Continue Any,
                          1=Continue Specific) (if OPTCD = RESTORE)
                          NIBPSDFA field = DFASY Data Mode (0=Continue Any,
                          1=Continue Specific) (if OPTCD = RESTORE)
                          NIBAFFIN field = affinity owner (0=VTAM, 1=APPL)
                          (if NIBLAFFN = 1)
                    ARECLEN field = length of received BIND response
                           (if NIB PROC = NEGBIND)
                    ARECLEN field = total length of the session recovery data
                           (if OPTCD = RESTORE)
```

OPNSEC

```
OPNSEC →     RPL:     ACB field → ACB
                      NIB field → NIB:
                           Note: NAME and CID
                           are mutually exclusive.
                           NAME field = symbolic name of PLU in
                                        pending active session
                           NETID field = symbolic network identifier of PLU in
                                         pending active session
                           LUAFFIN field = APPL|NOTAPPL
                           CID field = CID of pending active session
                           USERFLD field = data to be returned at the
                                           completion of subsequent
```

11. Applies to the encryption facility only.

```
                                   RPL-based requests
                              EXLST field → exit routine list
                              ENCR field =level of cryptography required
                                          (REQD|SEL|NONE)
                              RESPLIM field = responded-output-limit
                              SDT field = responder to SDT requests
                              PROC field = processing options
                              BNDAREA field (if PROC = NEGBIND) →
                                             0|BIND area containing
                                             session parameter
                              LISTEND field = YES
                         Note: ECB and EXIT are mutually exclusive.
                         If ECB is used, only one is allowed.
                         ECB field → external ECB
                         ECB field = internal ECB
                         EXIT field → RPL exit routine
                         BRANCH field = YES|NO
                         OPTCD field = (SYN|ASY,CA|CA)

     Register 0 and 15 = return codes

     RPL:          RTNCD field = recovery action return code
                   FDB2 field = specific error return code
                   REQ field = request code (42)
                   SSENSEI field = CPM|STATE|FI|RR|PATH|0
                   SSENSMI field = system-sense modifier value (or 0)
                   USENSEI field = user-sense value (or 0)
                   ARG field = CID of the established session
                   AREA field → NIB:
                        NAME field = symbolic name of the PLU
                        of established session
                        NETID field = symbolic network identifier of the PLU
                        of established session
                        CID field = CID of the established session
                        CON field = YES|NO (session established
                                            or not)
                        NIBAFFIN field = affinity owner (0=VTAM, 1=APPL)
                        (if NIBLAFFN = 1)
```

RCVCMD

```
RCVCMD →     RPL:     ACB field→  ACB
                      Note: ECB and EXIT are mutually exclusive.
                      If ECB is used, only one is allowed.
                      ECB field → external ECB
                      ECB field = internal ECB
                      EXIT field → RPL exit routine
                      BRANCH field = YES|NO
                      AREA field → input area for message
                      AREALEN field = length of input area
                      OPTCD field = (SYN|ASY,TRUNC,Q|NQ)

     Registers 0 and 15 = return codes

     RPL:     RTNCD field = recovery action return code
              FDB2 field = specific error return code
              REQ field = request code (40)
              AAREALN field = maximum length of reply
               allowed (if requested)
              RECLEN field = length of input message
```

RECEIVE

```
RECEIVE →     RPL:     ACB field → ACB
                       ARG field = CID session
                       Note: ECB and EXIT are mutually exclusive.
                       If ECB is used, only one is allowed.
                       ECB field → external ECB
                       ECB field = internal ECB
                       EXIT field → RPL exit routine
                       BRANCH field = YES|NO
                       AREA field → input data area
                       AREALEN field = length of input data area
                       RTYPE field = (DFSYN|NDFSYN,DFASY|NDFASY,
                            RESP|NRESP)
                       OPTCD field = (SYN|ASY,
                            CA|CS|CONDCS
                            SPEC|ANY,
                            TRUNC|KEEP|NIBTK,
                            Q|NQ)

    Registers 0 and 15 = return codes

    RPL:     RTNCD field = recovery action return code
             FDB2 field = specific error return code
             REQ field = request code (35)
             SSENSEI field = CPM|STATE|FI|RR|PATH|0
             SSENSMI field = system-sense modifier value (or 0)
             USENSEI field = user-sense value (or 0)
             USER field = data from USERFLD in NIB used
                  to establish session
             SEQNO field = sequence number of input
             RESPOND field = (EX|NEX,FME|NFME,RRN|
                  NRRN,QRESP|NQRESP)
             ARG field = CID of session completing the
                  RECEIVE
             RTYPE field = type of input received
             RECLEN field = length of input data
             CONTROL field = DATA|BID|BIS|CANCEL|
                  CHASE|LUS|QC|RTR|QEC|
                  RELQ|RSHUTD|SBI|SHUTC|
                  SHUTD|SIGNAL
             SIGDATA field = signal data (if CONTROL =
                  SIGNAL)
             CHNGDIR field = (CMD|NCMD,|REQ|NREQ)
             BRACKET field = (BB|NBB,EB|NEB,CEB|NCEB)
        CHAIN field = FIRST|MIDDLE|LAST|ONLY
        CRYPT field = YES|NO
        CODESEL field = STANDARD|ALT
        OPTCD field = FMHDR|NFMHDR
        RPLURH field =3-byte user RH
```

REQSESS

```
REQSESS →     RPL:     ACB field → ACB
                       NIB field→  NIB:
                            NAME field = symbolic name of PLU
                            NETID field = symbolic network identifier of PLU
                            LOGMODE field = logon mode name
                            USERFLD field = correlator to be returned in
                                              SCIP or NSEXIT exit routine
                            LISTEND = YES
                            Note: MTSAREA and BNDAREA are
                            mutually exclusive.
```

12. There is no RPL operand for this field.

```
                                   MTSAREA field = 0|MTS area address
                                   BNDAREA field → 0|BIND area
                                   Note: ECB and EXIT are mutually exclusive.
                              If ECB is used, only one is allowed.
                              ECB field → external ECB
                              ECB field = internal ECB
                              EXIT field → RPL exit routine
                              BRANCH field = YES|NO
                              AREA field → user data for initiate
                              RECLEN field = user data length
                              AAREA field = 0
                              OPTCD field = (SYN|ASY, NQ, MTS|NMTS)
```

Registers 0 and 15 = return codes

```
RPL:     RTNCD field = recovery action return code
         FDB2 field = specific error return code
         REQ field = request code (41)
         SSENSEI field = CPM|STATE|FI|RR|PATH|0
         SSENSMI field = system-sense modifier value (or 0)
         USENSEI field = user-sense value (or 0)
```

RESETSR

```
RESETSR →     RPL:     ACB field→  ACB
                       ARG field = CID of session
                       Note: ECB and EXIT are mutually exclusive.
                       If ECB is used, only one is allowed.
                       ECB field → external ECB
                       ECB field = internal ECB
                       EXIT field → RPL exit routine
                       BRANCH field = YES|NO
                       RTYPE field = (DFSYN|NDFSYN,DFASY|
                            NDFASY,RESP|NRESP)
                       OPTCD field = (SYN|ASY,CA|CS)
```

Register 0 and 15 = return codes

```
RPL:           RTNCD field = recovery action return code
               FDB2 field = specific error return code
               REQ field = request code (36)
               USER field = data from USERFLD in NIB used
                                    to establish session
```

RPL

```
RPL  AM operand = VTAM
     ACB field → ACB
     NIB field → NIB
     AREA field → data area
     AREALEN field = data area length
     RECLEN field = data length
     AREA = data area address
     AAREA field → alternate data area
     AREALEN → data area length
     AAREALEN field = alternate data length
     Note: ECB and EXIT are mutually exclusive.
     If ECB is used, only one is allowed.
     ECB field → external ECB
     ECB field = INTERNAL
     EXIT field → RPL exit routine
```

13. Applies only to the encryption facility.

```
BRANCH field = YES|NO
SEQNO field = sequence number
POST field = SCHED|RESP
RESPOND field = (EX|NEX,FME|NFME,RRN|NRRN,
             QRESP|NQRESP)
CONTROL field = DATA|BID|BIS|CANCEL|CHASE|LUS|QC
             RTR|QEC|RELQ|RSHUTD|SBI|SHUTC|
             SHUTD|SIGNAL|BIND|CLEAR|RQR|SDT|
             STSN|UNBIND|SWITCH
CHAIN field = FIRST|MIDDLE|LAST|ONLY
CHNGDIR field = (CMD|NCMD,REQ|NREQ)
CRYPT field =YES|NO
BRACKET field = (BB|NBB,EB|NEB,CEB|NCEB)
RTYPE field = (DFSYN|NDFSYN,DFASY|NDFASY,RESP|
             NRESP)
STYPE field = REQ|RESP
IBSQAC field = SET|TESTSET|INVALID|IGNORE|TESTPOS|
             TESTNEG|RESET
OBSQAC field = SET|TESTSET|INVALID|IGNORE|TESTPOS|
             TESTNEG|RESET
IBSQVAL field = inbound sequence number
OBSQVAL field = outbound sequence number
SIGDATA field = signal data
CODESEL field = STANDARD|ALT
PARMS field = (THRDPTY = NOTIFY|NONOTIFY)
PARMS = SONCODE = code
SSENSEO field = CPM|STATE|FI|RR|0
SSENSMO field = system-sense modifier value (or 0)
USENSEO field = user-sense value (or 0)
OPTCD field = (NIBTK|TRUNC|KEEP,
             NFMHDR|FMHDR,
             CONALL|CONANY,
             SYN|ASY,
             CA|CS,
             COND|UNCOND|UNBIND,
             SPEC|ANY,
             ACCEPT|ACQUIRE,
             QUIESCE|STOP|START|HOLD,
             RELEASE|PASS,
             RELRQ|NRELREQ,
             LOGONMSG|DEVCHAR|COUNTS|TERMS|STATUS|
             APPSTAT|CIDXLATE|TOPLOGON|
             USERVAR|SESSPARM|SESSKEY,
             LMPEO|NLMPEO,
             CONTCHN|NCONTCHN,
             BUFFLST|NBUFFLST,
             USERRH|NUSERRH,
             SENSE|NSENSE,
             SONCODE|NSONCODE,RSPQUED| NRSPQUED,
             NBACKUP|BACKUP,
             Q|NQ,
             QALL|QSESSLIM|QNOTENAB,
             MTS|NMTS,
             KEEPSRB|NKEEPSRB)
```

SEND

```
SEND →     RPL:     ACB field → ACB
                    ARG field = CID of session
                    Note: ECB and EXIT are mutually exclusive.
                    If ECB is used, only one is allowed.
                    ECB field → external ECB
                    ECB field = internal ECB
```

14. Applies only to the encryption facility.

```
                    EXIT field → RPL exit routine
                    BRANCH field = YES|NO
                    POST field = SCHED|RESP
                    AREA field → data to be sent or buffer list
                    RECLEN field = length of data to be sent or length
                         of buffer list
                    RESPOND field = (EX|NEX,FME|NFME,RRN|
                         NRRN,QRESP|NQRESP)
                    RTYPE field = (DFSYN|NDFSYN,DFASY|NDFASY,
                         RESP|NRESP)
                    CONTROL field = DATA|BID|BIS|CANCEL|
                         CHASE|LUS|QC|RTR|QEC|
                         RELQ|RSHUTD|SBI|SHUTC|
                         SHUTD|SIGNAL
                    SIGDATA field = signal data (for STYPE=REQ and
                         CONTROL=SIGNAL)
                    CHNGDIR field = (CMD|NCMD,REQ|NREQ)
                    BRACKET field = (BB|NBB,EB|NEB,CEB|NCEB)
                    CHAIN field = FIRST|MIDDLE|LAST|ONLY
                    CRYPT field =YES|NO
                    CODESEL field = STANDARD|ALT
                    STYPE field = REQ|RESP

                    If STYPE = RESP,
                         SEQNO field = sequence number

                    If STYPE = RESP and RESPOND = EX
                         SSENSEO field = CPM|STATE|FI|RR|0
                         SSENSMO field = system-sense modifier field
                         USENSEO field = user-sense value

                    OPTCD field = (SYN|ASY,
                         FMHDR|NFMHDR,
                         CA|CS,
                         LMPEO|NLMPEO,
                         CONTCHN|NCONTCHN,
                         BUFFLST|NBUFFLST,
                         USERRH|NUSERRH,
                         RSPQUED|NRSPQUED)

Register 0 and 15 = return codes

RPL:        RTNCD field = recovery action return code
            FDB2 field = specific error return code
            REQ field = request code (34)
            USER field = data from USERFLD in NIB used
                                    to establish session
            SEQNO field = sequence number (last sequence
                          number for LMPEO)
            OBSQVAL field = first sequence number for LMPEO

            If POST = RESP (and it was not overridden)
                 CONTROL field = SNA request code returned
                 on response
                 RESPOND field = (EX|NEX|FME|NFME,RRN|NRRN)
                 CHNGDIR field = REQ indicator returned on
                 response
                 SSENSMI field = CPM|STATE|GI|RR|PATH|0
                 USENSEI field = user-sense value (or 0)
```

SENDCMD

```
SENDCMD →     RPL: ACB field → ACB
              AREA field → header and command to be sent
              RECLEN field = length of header and command to be sent
              Note: ECB and EXIT are mutually exclusive.
              If ECB is used, only one is allowed.
              ECB field → external ECB
              ECB field = internal ECB
              EXIT field → RPL exit routine
              BRANCH field = YES|NO
              OPTCD field = SYN|ASY

     Register 0 and 15 = return codes

     RPL:   RTNCD field = recovery action return code
            FDB2 field = specific error return code
            REQ field = request code (39)
```

SESSIONC

```
SESSIONC →    RPL:    ACB field → ACB
                      Note: ARG and NIB are mutually exclusive.
                      ARG field = CID of session or pending active session
                      NIB field → NIB: (only to send negative response
                           to BIND)
                           Note: NAME and CID
                           are mutually exclusive.
                           NAME is required if
                           the NIB field is used.
                           NAME field = symbolic name of PLU
                           NETID field = symbolic network identifier of PLU
                           CID field = CID of pending active session
                           LISTEND field = YES
                      Note: ECB and EXIT are mutually exclusive.
                      If ECB is used, only one is allowed.
                      ECB field → external ECB
                      ECB field = internal ECB
                      EXIT field → RPL exit routine
                      BRANCH field = YES|NO
                      STYPE field = REQ|RESP

                      If RESPONSE = NEX and CONTROL = STSN,
                      IBSQVAL field = inbound sequence
                           number
                      IBSQAC field = RESET|TSTPOS|
                           INVALID|TESTNEG
                      OBSQVAL field = outbound sequence
                           number
                      OBSQAC field = RESET|TESTPOS|
                           INVALID|TESTNEG
```

SETLOGON

```
SETLOGON →    RPL:    ACB field → ACB
                      Note: ECB and EXIT are mutually exclusive.
                      If ECB is used, only one is allowed.
                      ECB field → external ECB
                      ECB field = internal ECB
                      EXIT field → RPL exit routine
                      BRANCH field = YES|NO
                      OPTCD field = (SYN|ASY,
                           START|STOP|QUIESCE|HOLD|
                           PERSIST|NPERSIST|GNAMEADD|
                           GNAMEDEL|GNAMESUB)
                      PARMS = (MAXSESS = number of sessions,
                               FORCETKO= YES|NO,
                           PSTIMER = time limit)
```

```
                              NIB field → NIB
                                   GNAME field = generic name of application
                                   AFFIN = VTAM|APPL
                                   NIBUCVA = address of storage containing a vector list
                                   LISTEND = YES

          Register 0 and 15 = return codes

          RPL: RTNCD field = recovery action return code
               FDB2 field = specific error return code
               REQ field = request code (21)
               SSENSEI field = CPM|STATE|FI|RR|PATH|0
               SSENSMI field = system-sense modifier value (or 0)
               USENSEI field = user-sense value (or 0)
               RECLEN field = number of queued CINIT requests
                         (if OPTCD = QUIESCE)
```

SHOWCB

```
SHOWCB     AM operand = VTAM
           control block type operand → control block
           FIELDS operand = control block field to be extracted
           AREA operand → work area where contacts of blocks
                     will be placed
           LENGTH operand = length of the work area
           MF operand = list, generate, or execute form parameters

           Register 0 = error return code (if register 15 indicates an error)
           or
           Register 0 = length of the control block field extracted
                     (if register 15 indicated successful completion)
           Register 15 = general return code
```

SIMLOGON

```
SIMLOGON →     RPL:     ACB field → ACB
                        NIB field → NIB:
                             NAME field = symbolic name of LU
                             NETID field = symbolic network identifier of LU
                             USERFLD field = correlator to be returned in
                                  LOGON or NSEXIT exit
                                  routine
                             LOGMODE field = logon mode name
                             LISTEND field = YES|NO
                        Note: ECB and EXIT are mutually exclusive.
                        If ECB is used, only one is allowed.
                        ECB field → external ECB
                        ECB field = internal ECB
                        EXIT field → RPL exit routine
               BRANCH field = YES|NO
               AREA field → user data for Initiate
               RECLEN field = user data length
               OPTCD field = (SYN|ASY,Q|NQ,CONANY|
                    CONALL,RELQ|NRELRQ,
                    BACKUP|NBACKUP
                    QALL|QSESSLIM|QNOTENAB)

          Registers 0 and 15 = return codes

          RPL:     RTNCD field = recovery action return code
               FDB2 field = specific error return code
               REQ field = request code (22)
               SSENSEI field = CPM|STATE|FI|RR|PATH|0
               SSENSMI field = system-sense modifier value (or 0)
               USENSEI field = user-sense value (or 0)
```

```
NIB:     NIBSLWRK = on if SIMLOGON successful for this NIB
         NIBNNAMS = on if the association established with a partner
         LU is known by the application's network name when it has
         a generic name.
```

TERMSESS

```
TERMSESS →     RPL:     ACB field → ACB
                        Note: ARG and NIB are mutually exclusive.
                        ARG field = CID of session or pending active
                                    session to be terminated
                        NIB field → NIB:
                             Note: NAME and CID
                             are mutually exclusive.
                             NAME is required if
                             the NIB field is used.
                             NAME field = symbolic name of PLU
                             NETID field = symbolic network identifier of PLU
                             CID field = CID of session or pending active
                                         session to be terminated
                             LISTEND field = YES
                        Note: ECB and EXIT are mutually exclusive.
                        If ECB is used, only one is allowed.
                        ECB field → external ECB
                        ECB field = internal ECB
                        EXIT field → RPL exit routine
                        BRANCH field = YES|NO
                        OPTCD = (SYN|ASY,COND|UNCOND|
                                      UNBIND|TERMQ,SONCODE|NSONCODE)
                        PARMS = (SONCODE = code)
                        SSENSEO field = CPM|STATE|FI|RR|0
                        SSENSMO field = system-sense modifier value (or 0)
                        USENSEO field = user-sense value (or 0)
```

```
Registers 0 and 15 = return codes

RPL:     RTNCD field = recovery action return code
         FDB2 field = specific error return code
         REQ field = request code (44)
         SSENSEI field = CPM|STATE|FI|RR|PATH|0
         SSENSMI field = system-sense modifier value (or 0)
         USENSEI field = user-sense value (or 0)
```

TESTCB

```
TESTCB     AM operand = VTAM
           control block operand → control block
           field name operand = test value
           ERET operand → error exit routine
           MF operand = list, generate, or execute
                        form parameters
```

```
Register 0 = error return code (if register 15
     indicates an error)
Register 15 = general return code
PSW condition code = test result
```

Appendix B. Return codes and sense fields for RPL-based macroinstructions

This appendix provides information about return code posting and explains what the different return code and feedback field values mean. It also provides information about SNA sense fields.

Return code posting

VTAM posts return code information in registers 0 and 15, and in certain fields of the request's RPL. These fields are referred to as feedback fields. The manner in which registers 0 and 15 and the feedback fields are posted depends on whether synchronous request handling, asynchronous request handling with an ECB, or asynchronous request handling with an RPL exit routine is used.

Chapter 3, "Organizing an application program," on page 33, provides an overview of these modes of operation. Chapter 9, "Handling errors and special conditions," on page 277 describes the way that VTAM and the application program handle RPL-based macroinstruction errors and special conditions. This appendix lists the RTNCD and FDB2 feedback field values that are set when those conditions arise.

Note: RTNCD and FDB2 designate RPL DSECT fields (and are not RPL operands). The RPL DSECT label for RTNCD is RPLRTNCD, and the label for FDB2 is RPLFDB2. Because of the similarity in spelling between these fields and label designations and others found in the RPL, refer to Table 112 on page 751, Figure 169 on page 767, and Figure 170 on page 768 to avoid misinterpreting or incorrectly designating field names. The FDBK2 parameter on the SHOWCB and TESTCB macroinstructions represents the RPLFDB2 field.

Figure 104 on page 652, Figure 105 on page 653, and Figure 106 on page 654 show the RTNCD,FDB2 combinations that are valid for each macroinstruction. Only the RPL-based macroinstructions are included because feedback posting applies only to RPL-based macroinstructions. CHECK and EXECRPL are not shown because *all* of the indicated RTNCD,FDB2 combinations are possible upon return from them.

Although specific error return codes apply only when RTNCD contains a recovery action code that is not 0, Figure 104 on page 652, Figure 105 on page 653, and Figure 106 on page 654 include some FDB2 values for RTNCD=0. These are additional information codes that apply to certain normally completing requests. These codes are explained in the section following Figure 104 on page 652, Figure 105 on page 653, and Figure 106 on page 654.

After you have used Figure 104 on page 652, Figure 105 on page 653, and Figure 106 on page 654 to determine which RTNCD,FDB2 combinations are possible for a particular macroinstruction, refer to the return code descriptions for an explanation of each RTNCD,FDB2 combination.

© Copyright IBM Corp. 2000, 2013

RTNCD	FDB2	APPCCMD	CHANGE	CLSDST	INQUIRE	INTRPRET	OPNDST	OPNSEC	RCVCMD	RECEIVE	REQSESS	RESETSR	SEND	SENDCMD	SESSIONC	SETLOGON	SIMLOGON	TERMSESS
	X'00'(0)			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'05'(5)				X	X	X											
	X'06'(6)								X	X								
	X'07'(7)				X													
X'00'(0)	X'08'(8)			X			X										X	
	X'09'(9)						X											
	X'0A'(10)				X													
	X'0B'(11)	X																
	X'0D'(13)				X													
	X'03'(3)									X								
X'04'(4)	X'04'(4)									X			X		X			
	X'05'(5)			X	X	X	X	X			X				X		X	X
X'08'(8)	X'00'(0)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'0A'(10)									X								
	X'0B'(11)			X			X			X		X	X		X			X
X'0C'(12)	X'0C'(12)									X		X	X					
	X'0D'(13)												X					
	X'0E'(14)													X				
	X'00'(0)			X			X	X			X						X	
	X'01'(1)						X											
	X'02'(2)			X			X				X						X	
	X'03'(3)						X	X		X	X	X	X		X	X	X	
	X'05'(5)						X	X					X					
	X'07'(7)									X		X	X		X			
	X'09'(9)									X		X	X		X			
	X'0A'(10)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'0D'(13)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'0E'(14)			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'0F'(15)									X			X	X				
X'10'(16)	X'11'(17)						X											
	X'12'(18)							X			X							X
	X'13'(19)						X											
	X'14'(20)						X	X										
	X'15'(21)			X			X											
	X'16'(22)						X										X	
	X'17'(23)												X					
	X'18'(24)		X		X		X	X								X		
	X'19'(25)		X		X											X		
	X'1A'(26)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'1B'(27)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'1C'(28)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 104. RTNCD,FDB2 combinations possible for each macroinstruction (Part 1 of 3)

RTNCD	FDB2	APPCCMD	CHANGE	CLSDST	INQUIRE	INTRPRET	OPNDST	OPNSEC	RCVCMD	RECEIVE	REQSESS	RESETSR	SEND	SENDCMD	SESSIONC	SETLOGON	SIMLOGON	TERMSESS
	X'00'(0)			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'02'(2)			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'03'(3)			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'04'(4)	This code applies only to CHECK																
	X'10'(16)			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	X'11'(17)										X							
	X'12'(18)										X		X	X		X		
	X'13'(19)			X	X	X	X	X	X	X	X	X	X	X	X		X	X
	X'1E'(30)			X	X	X	X	X	X	X	X		X	X		X	X	X
	X'23'(35)			X	X	X		X		X		X	X		X			X
	X'24'(36)									X		X	X		X			
	X'3B'(59)												X					
	X'3C'(60)												X		X			
	X'40'(64)												X		X			
	X'41'(65)									X			X		X			
	X'42'(66)														X			
	X'44'(68)												X					
	X'47'(71)												X					
	X'48'(72)														X			
	X'49'(73)														X			
X'14'(20)	X'4A'(74)			X	X	X	X				X						X	
	X'4B'(75)				X	X												
	X'4C'(76)					X												
	X'4D'(77)					X	X											
	X'4E'(78)				X		X											
	X'4F'(79)			X			X				X						X	
	X'50'(80)																X	
	X'51'(81)			X	X		X				X						X	
	X'52'(82)				X	X	X	X			X				X	X	X	X
	X'53'(83)		X	X	X	X	X	X			X					X	X	X
	X'55'(85)					X	X	X	X				X	X	X			X
	X'57'(87)			X				X										
	X'5E'(94)			X														
	X'60'(96)							X							X			
	X'61'(97)															X		
	X'6C'(108)								X				X	X				
	X'6D'(109)								X					X				
	X'6E'(110)													X				
	X'6F'(111)								X					X				
	X'70'(112)								X					X				
	X'71'(113)												X					
	X'73'(115)																	
	X'74'(116)							X										

Figure 105. RTNCD,FDB2 combinations possible for each macroinstruction (Part 2 of 3)

RTNCD	FDB2	APPCCMD	CHANGE	CLSDST	INQUIRE	INTRPRET	OPNDST	OPNSEC	RCVCMD	RECEIVE	REQSESS	RESETSR	SEND	SENDCMD	SESSIONC	SETLOGON	SIMLOGON	TERMSESS
X'14'(20)	X'75'(117)							X										
	X'76'(118)							X										
	X'77'(119)												X					
	X'78'(120)												X					
	X'79'(121)												X					
	X'7B'(123)												X					
	X'7C'(124)														X			
	X'7D'(125)						X										X	
	X'7E'(126)																X	X
	X'7F'(127)									X		X		X	X			
	X'80'(128)															X		
	X'81'(129)																	X
	X'82'(130)						X										X	
	X'83'(131)						X										X	
	X'84'(132)						X										X	
	X'85'(133)									X		X	X		X			
	X'86'(134)		X													X		
	X'87'(135)				X											X		
	X'88'(136)		X															
	X'89'(137)															X		
	X'8A'(138)															X		
	X'8B'(139)				X											X		
	X'8C'(140)		X													X		
	X'8D'(141)															X		
	X'8E'(142)															X		
	X'8F'(143)		X															
	X'90'(144)															X		
	X'91'(145)															X		
	X'92'(146)															X		
	X'93'(147)															X		
	X'94'(148)										X					X		
	X'95'(149)										X					X		
	X'96'(150)										X					X		

Figure 106. RTNCD,FDB2 combinations possible for each macroinstruction (Part 3 of 3)

Should you detect a return code during program execution other than one described in this appendix, stop session communication. You can use SHOWCB macroinstructions to extract the contents of the RPL fields. You should then obtain a program dump. Save your source listings and any program execution output for IBM program service representatives.

RPL return code (RTNCD,FDB2) combinations

This section describes all the RTNCD,FDB2 combinations that can be set in an RPL when it is posted complete. The information posted in registers 0 and 15 is specified in Chapter 9, "Handling errors and special conditions," on page 277. The macroinstruction description in "RPL—Create a request parameter list" on page 487 lists all the RPL fields which can be set when each macroinstruction is posted complete.

RTNCD	FDB2	Explanation
0	0	Normal completion or request accepted

The operation has been completed normally or the request has been accepted.

RTNCD	FDB2	Explanation
0	5	Input area too small

You issued INQUIRE, INTRPRET, or OPNDST OPTCD=RESTORE and specified an input work area that is too small. VTAM has placed the required length (in bytes) in the RPL's RECLEN field (for INQUIRE) or ARECLEN (for INTRPRET). No data has been placed in the work area.

Obtain a work area that is at least as long as the value set in RECLEN or ARECLEN, place the length in the AREALEN field (for INQUIRE) or AAREALN (for INTRPRET), and reissue INQUIRE or INTRPRET.

RTNCD	FDB2	Explanation
0	6	No input available

A RECEIVE OPTCD=NQ was issued and there was no input of the specified RTYPE available to satisfy the macroinstruction, or a RCVCMD OPTCD=NQ was issued and there was no input available to satisfy the macroinstruction.

RTNCD	FDB2	Explanation
0	7	INQUIRE information not available

One of the following has occurred:

- You issued INQUIRE OPTCD=LOGONMSG to obtain user data (a logon message) from a queued CINIT and there is no queued CINIT.
- You issued INQUIRE OPTCD=SESSPARM to obtain session parameters from a queued CINIT and there is no queued CINIT.
- You issued INQUIRE OPTCD=SESSKEY to obtain the session cryptography key, and there is no session cryptography key.
- You issued INQUIRE OPTCD=DEVCHAR for a cross-domain resource.
- You issued INQUIRE OPTCD=TOPLOGON for queued CINITs, and there are no queued CINITs.
- You issued INQUIRE OPTCD=CIDXLATE for a session that has not been established.
- You issued an INQUIRE OPTCD=USERVAR and no USERVAR was defined.

- You issued an INQUIRE OPTCD=PERSESS, and no record application program interface sessions are pending recovery.

The problem might be due to an incorrectly set NAME field in the NIB, an CID that is not valid in the NIB or RPL, a failure on the part of the system programmer to create the appropriate entry during VTAM definition, or a VARY command issued by the VTAM operator that deactivated the entry.

RTNCD	FDB2	Explanation
0	8	OPNDST OPTCD=ACQUIRE, SIMLOGON, or CLSDST OPTCD=PASS failed

An OPNDST OPTCD=ACQUIRE or SIMLOGON OPTCD=NQ failed for one of the following reasons: the requested logical unit is at its session limit or is not enabled for sessions in which it is to be the SLU. See Chapter 5, “Establishing and terminating sessions with logical units,” on page 81z/OS Communications Server: SNA Programming for a description of OPNDST and SIMLOGON.

A SIMLOGON OPTCD=Q failed because the requested logical unit is at its session limit and at least one of its current sessions is with the application program that issued the SIMLOGON.

A CLSDST OPTCD=PASS failed for one of two reasons. There is already a queued session between the logical unit being passed and the target primary logical unit, or you attempted to initiate or pass the session to the same PLU APPL.

RTNCD	FDB2	Explanation
0	9	OPNDST OPTCD=ACCEPT denied (no queued CINITs) or OPNDST OPTCD=RESTORE denied (no sessions restored)

You attempted to accept a session and indicated that your request should be rejected if no pending active session is waiting to be accepted (OPTCD=NQ). The request is rejected because no CINIT is queued for your application program.

An OPNDST OPTCD=RESTORE failed because the sessions that are requested are not pending recovery. None of the sessions specified by the NIBLIST are restored.

RTNCD	FDB2	Explanation
0	10(X'0A')	Application program not connectable

You issued INQUIRE OPTCD=APPSTAT to check an application program's ability to establish sessions. The application program is in an inactive, non-connectable state because the VTAM operator deactivated it. Therefore, the application program is not available for sessions.

RTNCD	FDB2	Explanation
0	11(X'0B')	Conditional Completion for APPCCMD

Some type of error might have occurred on an APPCCMD macroinstruction. For further problem determination, refer to the primary and secondary return codes in the RPL extension. See the z/OS Communications Server: SNA Programmer's LU 6.2 Reference for further information.

RTNCD	FDB2	Explanation
0	13(X'0D')	Additional sessions pending recovery

You have issued INQUIRE PERSESS and specified an input work area that is too small. VTAM fills the work area with as much information as possible and places the length used in the RPL's RECLEN. The INQUIRE must be reissued to recover the remainder of the information.

RTNCD	FDB2	Explanation
4	3	Exception request received

An exception request has been received. The reason for the exception is contained in the RPL's SSENSEI, SSENSMI, and USENSEI fields. If a negative response has not been sent to a request of this chain and if this request (the exception request) requires a response, move the input sense fields to the output sense fields and send a negative response. All requests in the current chain that have been received by the application program should be discarded. If the current request did not end the chain, issue RECEIVE macroinstructions with OPTCD=TRUNC and AREALEN=0 until CHAIN=LAST or CONTROL=CANCEL is received. No responses should be sent for any request in the rest of the chain.

RTNCD	FDB2	Explanation
4	4	Negative response received

The logical unit (or some other node in the network) has sent a response indicating that an exception condition was detected for one of the requests that the application program sent on this session. The SEQNO field indicates the sequence number of the request to which the negative response applies. The SSENSEI, SSENSMI, and USENSEI fields indicate the reason for the exception condition. See “RPL fields set by VTAM” on page 505 for more information on the SEQNO field, and “SNA sense fields” on page 679 for more information on the SSENSEI, SSENSMI, and USENSEI fields.

If the request with which the negative response is associated is part of an incomplete chain currently being transmitted to the logical unit, the application program should terminate the chain by issuing a SEND STYPE=REQ, CONTROL=DATA, CHAIN=LAST or a SEND STYPE=REQ, CONTROL=CANCEL to indicate that the logical unit can stop discarding the requests it is receiving. Refer to z/OS Communications Server: SNA ProgrammingChapter 6, “Communicating with logical units,” on page 151 for information about the use of STSN and CLEAR to alter sequence numbers. Also see the discussion of (RTNCD,FDB2)=(12,13) in this section.

RTNCD	FDB2	Explanation
4	5	Symbolic name known in this SSCP by its network-qualified name only

A real-to-symbolic translation request is made, and NIBNET is filled in with a network identifier, but VTAM cannot provide a symbolic name. VTAM knows this resource only by its network-qualified name; there is no symbolic name that represents this resource. Do one of the following:

- Use the network-qualified name
- Define a symbolic name to represent this resource.

RTNCD	FDB2	Explanation
8	0	Temporary storage shortage

VTAM is temporarily unable to secure enough storage to process the request. The request can usually be reissued (with EXECRPL, for example). For applications running at a priority near to or higher than VTAM's priority, the application should wait a brief time before retrying this.

In certain cases, the macroinstruction processing has not gotten far enough to have done significant work, and the request can be reissued. In other cases, the processing might have gone beyond some irreversible point before failing; as a result, the request cannot simply be reissued. For example, if the LOGON exit routine has been scheduled with a CINIT request and OPNDST OPTCD=ACCEPT is issued, the OPNDST operation can fail before responding to the CINIT, in which case the OPNDST can simply be reissued. If the response to CINIT had been sent, however, and then storage could not be obtained, the OPNDST request could not be reissued as there would no longer be a CINIT to accept. In this case, the application program might wish to initiate another session between itself and the LU, perhaps by using SIMLOGON. These two cases can be distinguished by a bit in the NIB; when the OPNDST OPTCD=ACCEPT is posted, NIBNACLQ is 1 if the response to CINIT is sent; otherwise it is 0.

RTNCD	FDB2	Explanation
12(X'0C')	10(X'0A')	Request canceled by RESETSR

This RECEIVE operation has been canceled by a RESETSR macroinstruction issued by another part of your application program.

RTNCD	FDB2	Explanation
12(X'0C')	11(X'0B')	Request canceled because the session has been terminated

The request has been canceled because the session was terminated. Session termination always cancels any pending requests for the session, and returns this return code in the RPL. See z/OS Communications Server: SNA Programming"Session outage notification" on page 110 for a list of the possible causes of session termination.

This return code is also used when an OPNDST OPTCD=(ACCEPT,SPEC,Q) is canceled by CLSDST.

RTNCD	FDB2	Explanation
12(X'0C')	12(X'0C')	Request canceled by CLEAR request

While the RPL-based request was being processed, a CLEAR request was sent or received on the session. This stops all data flow and cancels all pending communication requests on the session. The CLEAR request might have been sent by your application program (SESSIONC macroinstruction), or the request might have been sent on behalf of your application program by VTAM. The CLEAR request might also have been sent from the other end of the session.

RTNCD	FDB2	Explanation
12(X'0C')	13(X'0D')	Prior exception in chain detected

A series of chained requests was being sent to the logical unit and a negative response was returned for one of them. All subsequent SEND macroinstructions for that chain are posted complete with this return code; however, for each such SEND, the associated request unit is sent on the session to the session partner where it should be discarded.

RTNCD	FDB2	Explanation
12(X'0C')	14(X'0E')	Request cancelled - POA queue limit exceeded

The POA issued a SENDCMD after it reached its queue limit (POAQLIM on the APPL definition statement). Subsequent SENDCMDs complete with this return code until you receive all of the messages in the queue. You can empty the message queue by issuing RCVCMD OPTCD=NQ (no queue) until an RCVCMD completes with a return code and feedback of X'0006'. A SENDCMD now returns successfully.

RTNCD	FDB2	Explanation
16(X'10')	0	Logical unit not available, application program status not available, queued BIND not available, or incorrect dial parameters

This code is set for one of the following reasons:

- You are attempting to establish a session with a logical unit that is not active.
- You are attempting to pass a logical unit to a primary logical unit that is not active (or is in the process of being deactivated).
- You are attempting to issue an OPNSEC macroinstruction and there is no queued BIND request to respond to.
- You are attempting to determine the status of an application program that is in another domain, the status is not available, and your application program has to proceed without it.
- You issued a SIMLOGON macroinstruction that specifies dial parameters for a nonswitched PU.
- The dial parameters specified in the SIMLOGON macroinstruction do not match the original dial parameters.
- You issued a macroinstuction and a resource, such as a network address or storage, was not available. A sense code is returned in the RPL containing specific information.

The RPL system-sense (SSENSEI), the system-sense modifier (SSENSMI), and the user-sense (USENSEI) can contain a more detailed explanation of the failure.

RTNCD	FDB2	Explanation
16(X'10')	1	OPNDST failed

OPNDST failed; if a session had been established by the OPNDST, it has now been terminated. Some reasons for OPNDST failure are as follows:

- No network path could be obtained. For example, there might have been a failure of the virtual route or route extension, or the operator might have deactivated a network component along the path.
- A dial connection was not completed.
- A negative response to a CRV request was received.
- A request rejected response to a BIND request was received.
- The logical unit does not exist.
- A BIND response that is not valid was received; for example, a negotiable BIND response was received for a non-negotiable BIND request.
- OPNDST OPTCD=ACQUIRE specifies dial parameters for a nonswitched PU.
- The dial parameters specified in the OPNDST OPTCD=ACQUIRE do not match the original dial parameters.

The SSENSEI, SSENSMI, and USENSEI fields are set; these fields are described in "SNA sense fields" on page 679.

RTNCD	FDB2	Explanation
16(X'10')	2	Logical unit inhibited for sessions

You attempted to initiate a session and one of the logical units in the requested session is inhibited. For example, a VTAM application program is inhibited for sessions if it issues SETLOGON OPTCD=QUIESCE or has never issued SETLOGON OPTCD=START. Refer to Chapter 5, "Establishing and terminating sessions with logical units," on page 81z/OS Communications Server: SNA Programming for more information.

RTNCD	FDB2	Explanation
16(X'10')	3	HALT issued

The VTAM operator has issued a HALT command. Depending on the type of HALT, certain macroinstructions can no longer be issued by your application program. Refer to "TPEND exit routine is entered" on page 74z/OS Communications Server: SNA Programming for more information.

RTNCD	FDB2	Explanation
16(X'10')	5	Request or response encryption failure

Encryption has failed while:

- Sending an FM data request
- Sending the BIND response during OPNSEC processing
- Sending the CRV request during OPNDST processing.

RTNCD	FDB2	Explanation
16(X'10')	7	Request canceled by VARY command

The communication operation has been canceled because the VTAM operator deactivated a necessary portion of the path while the macroinstruction was being processed. If a LOSTERM exit routine is available, it has been scheduled. You can no longer communicate with the LU, and you should issue CLSDST to terminate its session with your application program.

RTNCD	FDB2	Explanation
16(X'10')	9	Unconditional Terminate or character-coded logoff received

The logical unit has sent an unconditional Terminate request or a character-coded logoff that is a request for unconditional session-termination. No further communication on the session is possible. CLSDST must be issued.

RTNCD	FDB2	Explanation
16(X'10')	10(X'0A')	VTAM error

An error occurred in VTAM itself. No further attempts to establish or terminate a session with the logical unit should be made.

RTNCD	FDB2	Explanation
16(X'10')	13(X'0D')	VTAM inactive for your ACB

The association between VTAM and your application program (ACB) that was established with OPEN has been broken; the ACB is in the process of being closed. This might have occurred because you have elsewhere issued a CLOSE that has not yet completed, or it might have occurred because VTAM has become inactive, or a VARY INACT was issued for your application program.

RTNCD	FDB2	Explanation
16(X'10')	14(X'0E')	Request abnormally terminated

VTAM has abnormally terminated a request because of an error detected while processing the request or because of an error in the associated session, task, or address space (for example, an abend). See “Isolation of errors” on page 322 for more information about error isolation and recovery.

RTNCD	FDB2	Explanation
16(X'10')	15(X'0F')	Buffers filled

Previously VTAM had received an RU; the application program did not have an appropriate EXLST exit routine or outstanding RECEIVE for the RU and there was no buffer space left for VTAM to queue the RU. Under these circumstances, VTAM discards that RU and any other RUs queued for the session and schedules the LOSTERM exit routine (if there is one) with reason code 36. If appropriate for the TS Profile for this session, a Clear is sent to the session partner. In all cases, the end of the session that experienced the buffer shortage is put into data-traffic-reset

state (at least momentarily). Any SEND or RECEIVE issued while the session is in this state is rejected with (RTNCD,FDB2)=(X'10',X'0F'). This mode of operation continues until a Start Data Traffic response is processed (or until the Clear function completes, if SDT is not appropriate for the TS profile).

RTNCD	FDB2	Explanation
16(X'10')	17(X'11')	SDT failure on OPNDST

A negative response was sent by a logical unit in reply to a Start Data Traffic (SDT) request. The OPNDST was not completed successfully. The SSENSEI, SSENSMI, and USENSEI fields are set; these fields are described in "SNA sense fields" on page 679.

RTNCD	FDB2	Explanation
16(X'10')	18(X'12')	Macroinstruction failure, sense included

A REQSESS, TERMSESS, or OPNSEC has failed. A sense code (SSENSEI, SSENSMI, and USENSEI field) is returned in the RPL for the failing macroinstruction.

RTNCD	FDB2	Explanation
16(X'10')	19(X'13')	Attempt to start LU 6.2 session request rejected

An LU 6.2 application has tried to start an LU 6.2 session independent of VTAM. No pending sessions have been disturbed. This occurs when an OPNDST is issued with an LU 6.2 user-specified BIND.

RTNCD	FDB2	Explanation
16(X'10')	20(X'14')	Attempt to start LU 6.2 session pending session terminated

An LU 6.2 application has tried to start an LU 6.2 session independent of VTAM. The pending session has been terminated. This occurs when the LOGMODE specified on an OPNDST resolves to an LU 6.2 BIND or when OPNSEC is issued for an LU 6.2 BIND.

RTNCD	FDB2	Explanation
16(X'10')	21(X'15')	An APPCCMD must be issued

An OPNDST or CLSDST has been issued for a pending LU 6.2 session. An APPCCMD CONTROL=OPRCNTL, QUALIFY=ACTSESS or QUALIFY=DACTSESS macroinstruction must be issued for this session. See the z/OS Communications Server: SNA Programmer's LU 6.2 Reference for more information.

RTNCD	FDB2	Explanation
16(X'10')	22(X'16')	Specified LU is nonswitched

The application issues a SIMLOGON or OPNDST OPTCD=ACQUIRE macroinstruction using the application supplied dial-out function. The specified LU is nonswitched and the request failed.

RTNCD	FDB2	Explanation
16(X'10')	23(X'17')	Encryption not allowed

You attempted to request encryption on a send, but session does not support encryption.

RTNCD	FDB2	Explanation
16(X'10')	24(X'18')	Sysplex is inaccessible

You attempted to use either the INQUIRE OPTCD=SESSNAME, SETLOGON OPTCD=GNAMEADD, SETLOGON OPTCD=GNAMEDEL, SETLOGON OPTCD=GNAMESUB, OPNDST, OPNSEC, or the CHANGE OPTCD=ENDAFFIN macroinstruction, but the coupling facility for this host is inaccessible.

RTNCD	FDB2	Explanation
16(X'10')	25(X'19')	Host is not member of Sysplex

The application issued either the INQUIRE OPTCD=SESSNAME, the CHANGE OPTCD=ENDAFFIN, or the SETLOGON OPTCD=GNAMEADD|GNAMEDEL|GNAMESUB macroinstruction, but the coupling facility for this host is inaccessible. The coupling facility might be inaccessible because:

- A coupling facility does not exist.
- A CFRM policy for the required coupling facility structure was not active.
- VTAM is not defined as an APPN node.
- VTAM has lost connectivity to the required coupling facility structure.

RTNCD	FDB2	Explanation
16(X'10')	26(X'1A')	SUSPEND failed

VTAM attempted to SUSPEND an RPL request issued in either cross-memory mode or in synchronous SRB mode with OPTCD=KEEPSRB specified. The attempt failed.

RTNCD	FDB2	Explanation
16(X'10')	27(X'1B')	RESUME failed

VTAM attempted to RESUME an RPL request issued in either cross-memory mode or in synchronous SRB mode with OPTCD=KEEPSRB specified. The attempt failed. VTAM is unable to post the request complete. If the application has a LOSTERM exit, it will be scheduled with a reason code of 44. For more information about the LOSTERM exit, see “LOSTERM exit routine” on page 241. The RPL is now available for reuse.

RTNCD	FDB2	Explanation
16(X'10')	28(X'1C')	OS level does not support requested function

A macroinstruction request required the use of an operating system service which is not supported by the active operating system level.

RTNCD	FDB2	Explanation
16(X'10')	29(X'1D')	Security Manager Error

An error was encountered when attempting to invoke the security management program. The APPL definition statement for this application specifies VERIFY=REQD or VERIFY=OPTIONAL, indicating that the use of an installed security manager was required for APPC sessions by this application program. However, VTAM was unable to successfully invoke the security manager. The SETLOGON START macroinstruction is rejected.

RTNCD	FDB2	Explanation
20(X'14')	0	VSAM request

The RPL contains a VSAM or other non-VTAM request code. No ECB has been posted and no RPL exit routine has been scheduled.

RTNCD	FDB2	Explanation
20(X'14')	2	Zero EXIT field

The RPL indicates that the ECB-EXIT field is being used as an EXIT field, but the RPL exit routine address in it is 0. No RPL exit routine has been scheduled.

RTNCD	FDB2	Explanation
20(X'14')	3	Zero ECB field

The RPL indicates that the ECB-EXIT field is being used to point to an external ECB, but the address in the field is 0. No ECB has been posted.

RTNCD	FDB2	Explanation
20(X'14')	4	Inactive RPL checked

CHECK was issued for an inactive RPL (an RPL that had been posted complete and for which CHECK has already been issued successfully). All RPL-based macroinstructions must use an inactive RPL. All CHECK macroinstructions, however, must use an active RPL; an RPL cannot be checked twice.

RTNCD	FDB2	Explanation
20(X'14')	16(X'10')	Control block not valid

The RPL's ACB field does not contain the address of a valid ACB or the ACB is closed. This can mean that the ACB field of the RPL was incorrectly set or the ACB has been destroyed.

RTNCD	FDB2	Explanation
20(X'14')	17(X'11')	RTYPE not valid

A RECEIVE has been issued with the RTYPE field set to NDFSYN, NDFASY, and RESP.

RTNCD	FDB2	Explanation
20(X'14')	18(X'12')	CLSDST in progress

At the time this macroinstruction was executed, a CLSDST request was pending for the session. The CLSDST request takes priority, and the request that received this return code cannot be honored.

RTNCD	FDB2	Explanation
20(X'14')	19(X'13')	CID not valid

The RPLARG field or the NIBCID field does not contain a valid CID, or a valid CID was issued with the wrong ACB, or INTRPRET is being used for a cross-domain LU.

You might have inadvertently modified the field, initially failed to set it, or used the CID of a session that no longer exists.

Another possibility is that you violated the following rule: when placing a CID into the RPLARG field, always use the ARG keyword (ARG=(6), for example), and when placing an NIB address into the RPL's NIB field, always use the NIB keyword (for example, NIB=(6)). Because these two fields occupy the same 4 bytes in the RPL, VTAM can distinguish between an NIB address and a CID only through your use of the ARG or NIB keyword. Thus, the presence of this return code could mean that you placed an NIB address in the RPL with the ARG keyword, and VTAM has rejected your CID as not valid.

This feedback information is also used when a CID is specified for INTRPRET, and the LU implied by the CID is in another domain.

RTNCD	FDB2	Explanation
20(X'14')	30(X'1E')	Data address or length not valid

A request was issued that specified a work area address that is beyond the addressable range of your application program. Here a work area is defined to be any storage area addressed by an RPL operand, for example, the areas referenced by AREA and AAREA.

Check the work area address and work area length fields in the RPL for an incorrect setting. See the RPL macroinstruction description to determine which fields must point to valid work areas for each macroinstruction.

If your application program resides in an authorized library, check for correct load module characteristics.

RTNCD	FDB2	Explanation
20(X'14')	35(X'23')	Request type not valid

When an RPL-based macroinstruction is issued, VTAM sets the REQ field in the RPL to indicate the type of macroinstruction that is using the RPL. The presence of

this return code indicates that you modified that code before the requested operation completed. To avoid this and other related errors, never modify an RPL while it is in use. Compare with VSAM request, (RTNCD,FDB2)=(X'14',X'00').

RTNCD	FDB2	Explanation
20(X'14')	36(X'24')	Request for address space not valid

You attempted to issue one of the following macroinstructions in other than the session address space: RECEIVE OPTCD=SPEC, RESETSR, SEND, or SESSIONC (except request rejected response to BIND).

RTNCD	FDB2	Explanation
20(X'14')	59(X'3B')	NFME-NRRN response

You attempted to send a response with the RESPOND field set to NFME and NRRN. A response must be identified as FME, RRN, or both; in effect, you have identified the response as neither.

RTNCD	FDB2	Explanation
20(X'14')	60(X'3C')	Previous macroinstruction outstanding

You issued a SEND POST=SCHED, a SEND for an expedited data-flow-control request, or a SESSIONC macroinstruction before a previous macroinstruction of the same type had been completed. Only one macroinstruction of the three preceding types can be outstanding on a session at a time. After the previous macroinstruction has been completed, this macroinstruction can be reissued.

RTNCD	FDB2	Explanation
20(X'14')	64(X'40')	CONTROL not valid

You modified the bits in the CONTROL field, or you used a CONTROL value for a SESSIONC macroinstruction that was not BIND, RQR, SDT, CLEAR, STSN, or SWITCH.

RTNCD	FDB2	Explanation
20(X'14')	65(X'41')	Data traffic not allowed

You attempted to communicate on a session for which no SDT request had been sent or for which a CLEAR is in progress. For certain TS profiles, until an SDT request/response exchange has occurred on the session, no traffic flow is possible; only SDT, Set and Test Sequence Numbers (STSN), Request Recovery (RQR), and Clear requests can be exchanged. Every time a Clear request is sent on a session, a new SDT request might be required before traffic flow can resume (this depends upon the transmission services profile used). For further information, refer to z/OS Communications Server: SNA Programming"Controlling flow" on page 164.

RTNCD	FDB2	Explanation
20(X'14')	66(X'42')	STYPE for SESSIONC not valid

STYPE=RESP has been specified for a SESSIONC CONTROL=CLEAR or a SESSIONC CONTROL=RQR macroinstruction. Only STYPE=REQ is valid. Also, if the NIB used to establish the session specified SDT=SYSTEM, then STYPE=RESP is not valid for SESSIONC CONTROL=SDT.

RTNCD	FDB2	Explanation
20(X'14')	68(X'44')	RESPLIM exceeded

The number of outstanding SEND POST=RESP macroinstructions for a session exceeds the RESPLIM value set in the NIB used to establish the session.

RTNCD	FDB2	Explanation
20(X'14')	71(X'47')	3270 SEND option not valid

The RPL specified by your LU type 0 3270 SEND macroinstruction had one or more of the following fields not valid: STYPE, RESPOND, CHAIN, or CONTROL. See “Exception conditions and sense information” on page 332z/OS Communications Server: SNA Programming for more information about exception conditions.

If the RPL was last used for a RECEIVE for the 3270, check the RESPOND field first; you might have failed to reset the field following the RECEIVE (RECEIVE sets the RESPOND field to (NEX,NFME,NRRN) in this case).

RTNCD	FDB2	Explanation
20(X'14')	72(X'48')	Session-control protocol violation

Protocol violations indicated are as follows:

- The PLU sent an SDT request while not in data-traffic-reset state, or the SDT sent was not allowed by the TS profile.
- The PLU sent a Clear request, and a previous Clear request has been sent and has not completed, or the Clear request was not allowed by the TS profile.
- The PLU sent an STSN request while not in data-traffic-reset state, or the STSN request was not allowed by the TS profile.
- The PLU sent an RQR request, and the RQR request was not allowed by the TS profile.
- The SLU sent an SDT response and any previously received SDT request had already been responded to, or an SDT request had not been received.

For more information, refer to “Controlling flow” on page 164z/OS Communications Server: SNA Programming.

RTNCD	FDB2	Explanation
20(X'14')	73(X'49')	STSN action/result code not valid

One of the following applies:

- You attempted to send a Set and Test Sequence Numbers (STSN) request and set the IBSQAC or OBSQAC fields (or both) to some value other than SET, TESTSET, IGNORE, or INVALID.

- You attempted to send an STSN response and set the IBSQAC or OBSQAC field (or both), to some value other than TESTPOS, TESTNEG, INVALID, or RESET.
- You attempted to send a result code that is not a valid response to the action code.

See "SESSIONC—Send a session-control request or response" on page 532 macroinstruction description for more information.

RTNCD	FDB2	Explanation
20(X'14')	74(X'4A')	Installation-wide exit routine was not available

You issued an INTRPRET macroinstruction; VTAM has located the appropriate entry in the interpret table, and found that the system programmer has specified a logon-interpret exit routine to do the interpret function. That routine, however, has not been loaded.

RTNCD	FDB2	Explanation
20(X'14')	75(X'4B')	INTRPRET sequence or LOGMODE not valid, or cryptographic incompatibility

You issued an INTRPRET macroinstruction—one of the following might apply:

- VTAM cannot locate an entry in the interpret table that corresponds to the sequence you provided.
- You might have inadvertently modified the sequence or the address in the RPL's AREA field that points to the sequence.
- The system programmer might have failed to properly define the entry in the interpret table.

After your application program has been tested and debugged and you have eliminated the possibility of the three situations listed above, you can assume that the terminal operator or program that initiated the logon must have passed an invalid logon sequence to your application program.

You issued an INQUIRE, OPNDST, SIMLOGON, REQSESS, or CLSDST OPTCD=PASS macroinstruction. Either the NIB for this request specified a logon mode name that could not be found in the logon mode table for the logical unit named in that NIB, or the SSCP discovered that cryptography had been specified for the requested session, but at least one of the logical units in the requested session did not support cryptography.

RTNCD	FDB2	Explanation
20(X'14')	76(X'4C')	Search argument for INQUIRE or INTRPRET not valid

You issued INQUIRE or INTRPRET, and failed to properly provide VTAM with the identity of the pending active session, logical unit, or application program:

- INTRPRET was issued and the name in the NIB was not that of a logical unit.
- INQUIRE (OPTCD=APPSTAT) was issued and one of the following conditions exists:
  - The name is not that of an application program.
  - The application program is a cross-domain resource, and the SSCP that owns the resource does not support INQUIRE (OPTCD=APPSTAT).

  - The application program is a cross-domain resource, and no active route exists to the host that owns the application program.
- INQUIRE OPTCD=TERMS was issued and the name was not that of a resource (such as an LU, PU, CLUSTER, or CDRSC) in the VTAM configuration tables.
- INQUIRE OPTCD=DEVCHAR was issued and the device characteristics were not available (perhaps because the logical unit was in another domain and there was no appropriate CINIT queued for the application program).
- INQUIRE OPTCD=LOGONMSG was issued and there was no appropriate CINIT queued for the application program.
- INQUIRE OPTCD=SESSPARM was issued with LOGMODE=0 in the NIB, and there was no appropriate CINIT queued for the application program.
- INQUIRE OPTCD=NQN was issued and one of the following applies:
  - The resource does not exist.
  - The resource is cross-domain an there is no active route to it.

Refer to "INQUIRE—Obtain logical unit information or application program status" on page 412z/OS Communications Server: SNA Programming for a description of the INQUIRE macroinstruction.

Assuming that the system programmer properly defined the entry in the VTAM configuration tables for the logical unit, you have probably: (1) failed to set a valid symbolic name in the NIB's NAME field or (2) correctly issued INQUIRE OPTCD=SESSPARM or INQUIRE OPTCD=DEVCHAR but the session has been terminated.

RTNCD	FDB2	Explanation
20(X'14')	77(X'4D')	No interpret table

You issued an INTRPRET macroinstruction, but there is no interpret table for the logical unit. The system programmer might have failed to include an interpret table for this logical unit during the VTAM definition process or the logical unit might be in another domain.

RTNCD	FDB2	Explanation
20(X'14')	78(X'4E')	Use of an NIB list not valid

You issued OPNDST OPTCD=ACCEPT without setting the NIB's LISTEND field to YES, or you specified a NIB list in which more than one NIB indicated PROC=NEGBIND.

RTNCD	FDB2	Explanation
20(X'14')	79(X'4F')	OPTCD setting not valid

The OPNDST or INQUIRE request fails because bits in the OPTCD field have been incorrectly set. From the OPNDST and the INQUIRE option code settings, you must specify only one value for the mutually exclusive sets of option codes. Because you cannot cause the field to be incorrectly set by using VTAM macroinstructions, you might have inadvertently modified the OPTCD field with assembler instructions.

RTNCD	FDB2	Explanation
20(X'14')	80(X'50')	RPL field not valid

The OPNDST, CLSDST, SIMLOGON, or REQSESS failed because the bits in the RPL's OPTCD or AAREA field were found to be not valid.

If an OPNDST or SIMLOGON failed, the particular bits that have been incorrectly set are those that form the CONANY-CONALL option code. This return code does not mean that the CONANY option was erroneously used in place of CONALL, or vice versa; it means that neither CONALL nor CONANY is indicated in the OPTCD field. Because you cannot cause the field to be incorrectly set in this manner by using VTAM macroinstructions, you might have inadvertently modified the OPTCD field with assembler instructions.

If a REQSESS failed, either OPTCD=NQ was not specified or the AAREA field of the RPL was not set to zero.

If a CLSDST failed, OPTCD=SENSE was specified and a zero sense was provided in the SSENSEO, SSENSMO, USENSEO fields of the RPL. A zero sense is not permitted for CLSDST OPTCD=SENSE.

RTNCD	FDB2	Explanation
20(X'14')	81(X'51')	OPNDST OPTCD=ACCEPT and SIMLOGON not allowed

You attempted to issue OPNDST OPTCD=ACCEPT to accept a CINIT for a session with a logical unit, or to issue SIMLOGON to initiate a session. However, these operations cannot be performed because of one of the following:

- The ACB was opened with MACRF=NLOGON.
- SETLOGON OPTCD=QUIESCE was issued and CINITs are pending.
- SETLOGON OPTCD=QUIESCE was issued and no matching CINIT was found.

RTNCD	FDB2	Explanation
20(X'14')	82(X'52')	NIB not valid

The request failed because there is no NIB at the location indicated in the RPL's NIB field.

RTNCD	FDB2	Explanation
20(X'14')	83(X'53')	Logical unit not found

The symbolic name you supplied in the NIB's NAME field or indicated by the RPL's AAREA field does not have a corresponding entry in the VTAM configuration tables. This can occur for one of the following reasons:

- You failed to set the NAME field correctly.
- The system programmer did not include the entry in the VTAM configuration tables during VTAM definition.
- The VTAM operator has not activated the major node containing the application program that issued the macroinstruction.

- The VTAM operator has not activated the major node containing the resource named in the NIB (in a cross-domain environment).
- A dynamically created definition for a cross-domain LU has been deleted after lack of use for a defined period of time.
- Contact with the resource was lost and the definition of the resource was subsequently deleted from the VTAM configuration tables.
- You issued either SETLOGON OPTCD=GNAMEADD, SETLOGON OPTCD=GNAMESUB, SETLOGON OPTCD=GNAMEDEL, INQUIRE OPTCD=SESSNAME, or CHANGE OPTCD=ENDAFFIN and one of the names you supplied is not valid.

If you were using an NIB list, no sessions have been established.

RTNCD	FDB2	Explanation
20(X'14')	85(X'55')	One of the following is true: • Application program is not authorized. • Application program name is not available. • Task association is not specified. • Application is not authorized to supply dial parameters. • PU is not authorized to accept dial parameters. • You must issue a send RPL.

- You attempted to acquire a logical unit (SIMLOGON or OPNDST), but the installation has denied you authorization to do so. The system programmer might have specified during VTAM definition that your application program is not authorized to acquire any logical units. If you are authorized to acquire logical units and you still receive this return code, this means that an authorization exit routine has been invoked and has determined that you cannot acquire the specific logical unit indicated in your request.
- You attempted to initiate a session, but the authorization exit routine has denied you authorization.
- You issued an INTRPRET macroinstruction; VTAM located the appropriate entry in the interpret table and found that the installation has specified an exit routine to convert the input sequence into an output sequence. That routine was loaded, but it failed to do the conversion.
- You issued one of the following macroinstructions in SRB mode without specifying the required task association: CLSDST, INQUIRE, INTRPRET, OPNDST, OPNSEC, REQSESS, RCVCMD, SENDCMD, SETLOGON, SIMLOGON, TERMSESS.

  Refer to Chapter 13, “Conventions and descriptions of VTAM macroinstructions,” on page 371z/OS Communications Server: SNA Programming for more information.
- An application that is not authorized to supply dial parameters attempted to supply dial parameters, or a PU that is not authorized to accept dial parameters attempted to accept dial parameters.

Refer to the z/OS Communications Server: SNA Resource Definition Reference for information about coding an application program major node (the AUTH operand of the APPL definition statement).

RTNCD	FDB2	Explanation
20(X'14')	87(X'57')	MODE field not valid

You issued an OPNDST or OPNSEC macroinstruction and failed to set the NIB's MODE field to RECORD.

RTNCD	FDB2	Explanation
20(X'14')	94(X'5E')	CLSDST OPTCD=PASS not authorized

CLSDST OPTCD=PASS is a function whose use is authorized by the installation. You attempted to use this function, but the installation has not authorized you to pass logical units to other primary logical units. This CLSDST macroinstruction should have been issued with RELEASE in effect, not PASS.

Refer to the z/OS Communications Server: SNA Resource Definition Reference for a description of the AUTH operand of the APPL definition statement.

RTNCD	FDB2	Explanation
20(X'14')	96(X'60')	LU name for CLSDST, SESSIONC, or OPNSEC not valid

You attempted to terminate a session with a logical unit that is not in session with your application program, or had no CINIT queued for your application program. This return code applies to CLSDST used with a logical unit's symbolic name.

You issued a SESSIONC macroinstruction to send a request rejected response to BIND, but the LU name field in the NIB does not match any BIND currently queued for the application program.

You issued an OPNSEC macroinstruction and a queued BIND could not be found for the LU name passed in the NIB.

RTNCD	FDB2	Explanation
20(X'14')	97(X'61')	SETLOGON not valid

Either you opened the ACB with its MACRF field set to NLOGON, or you issued SETLOGON OPTCD=QUIESCE and permanently closed the CINIT queue. Because you attempted to either open a CINIT queue that cannot be opened or close a CINIT queue that is closed, SETLOGON START, STOP, and QUIESCE are not valid.

You might have issued a SETLOGON OPTCD=PERSIST or NPERSIST with a PSTIMER value that is greater than the allowed value (86400 seconds).

Note: You can successfully issue SETLOGON OPTCD=PERSIST or SETLOGON OPTCD=NPERSIST with the MACRF field set to NLOGON or after a QUIESCE.

RTNCD	FDB2	Explanation
20(X'14')	108(X'6C')	Exceeded limit on outstanding RCVCMD requests

You attempted to issue an RCVCMD macroinstruction while a previous RCVCMD was outstanding. The limit on outstanding RCVCMD requests is one.

RTNCD	FDB2	Explanation
20(X'14')	109(X'6D')	Application program not authorized

Your application program is not authorized to issue the SENDCMD and RCVCMD macroinstructions, or your CNM application program attempted to send something other than a formatted Forward RU to the SSCP.

Refer to the z/OS Communications Server: SNA Resource Definition Reference for a description of the AUTH operand of the APPL definition statement.

RTNCD	FDB2	Explanation
20(X'14')	110(X'6E')	Syntax error in reply to VTAM operator message

In reply to a VTAM operator message, you issued a SENDCMD macroinstruction that contained a syntax error in the REPLY command.

RTNCD	FDB2	Explanation
20(X'14')	111(X'6F')	SENDCMD/RCVCMD processor inactive

The portion of VTAM that processes SENDCMD and RCVCMD macroinstructions is currently inactive for your application program, and the application program issued a SENDCMD or RCVCMD macroinstruction. The request cannot be processed because an ACB has not been opened for the portion of the application program that issued the SENDCMD or RCVCMD, or because a final CLOSE has been issued for this ACB but has not yet completed.

RTNCD	FDB2	Explanation
20(X'14')	112(X'70')	Program operator closing ACB with requests outstanding

Your application program is in the process of closing its ACB, and you (1) issued a SENDCMD macroinstruction for a command other than REPLY or (2) issued a RCVCMD OPTCD=Q and there were no VTAM messages available to satisfy the request.

RTNCD	FDB2	Explanation
20(X'14')	113(X'71')	Operator command not valid

You attempted to send a VTAM operator command to VTAM using the SENDCMD macroinstruction; however, the command was not recognized by VTAM, or it was a command (START or HALT) that cannot be sent by the application program.

RTNCD	FDB2	Explanation
20(X'14')	115(X'73')	SEND parameters for CNM not valid

You issued a SEND macroinstruction when using a CNM application program and you have specified a parameter that is not valid.

RTNCD	FDB2	Explanation
20(X'14')	116(X'74')	Negotiable response to non-negotiable BIND

You attempted to issue an OPNSEC PROC=NEGBIND to a non-negotiable BIND request. A request-rejected response to the BIND is sent with a sense code indicating resource unavailable (X'08010000').

RTNCD	FDB2	Explanation
20(X'14')	117(X'75')	Negotiable BIND response parameters not valid

You specified negotiable BIND parameters on an OPNSEC macroinstruction that are not valid. A request rejected response to the BIND is sent with a sense code indicating resource unavailable (X'0801000').

RTNCD	FDB2	Explanation
20(X'14')	118(X'76')	Negotiable BIND response size not valid

You specified a negotiable BIND response on OPNSEC that was greater than 256 bytes. A request rejected response to the BIND is sent with a sense code indicating resource unavailable (X'08010000').

RTNCD	FDB2	Explanation
20(X'14')	119(X'77')	FMD request unit required

You issued a SEND OPTCD=BUFFLST or a SEND OPTCD=LMPEO and the RU specified was not an FMD request unit.

RTNCD	FDB2	Explanation
20(X'14')	120(X'78')	Chain specification not valid

You issued a SEND OPTCD=(BUFFLST,USERRH) in which multiple chains or multiple partial chains were specified in the buffer list. Only requests from a single chain might be specified in a buffer list.

RTNCD	FDB2	Explanation
20(X'14')	121(X'79')	Buffer list length not valid

You issued a SEND OPTCD=BUFFLST, and RECLEN did not contain a nonzero multiple of 16.

RTNCD	FDB2	Explanation
20(X'14')	123(X'7B')	User RH not valid

One of the following conditions was detected for a SEND OPTCD=USERRH:

- The settings of the CONTROL operand and of the RU category field in the user RH were inconsistent. If CONTROL=DATA, then the RU category must be FMD. If CONTROL is not DATA, then the RU category must be DFC. See also (RTNCD,FDB2)=(X'14',X'77').
- A sense indicator in the user RH field was found to be on with zero sense provided. For a non LUO session, zero sense is architecturally incorrect.

RTNCD	FDB2	Explanation
20(X'14')	124(X'7C')	OPTCD=USERRH for SESSIONC not valid

You specified a SESSIONC macroinstruction with OPTCD=USERRH.

RTNCD	FDB2	Explanation
20(X'14')	125(X'7D')	XRF protocol error

A protocol error has occurred during the processing of a SIMLOGON or OPNDST macroinstruction.

SIMLOGON for a backup XRF request is processed by setting the "backup XRF session request" indicator in the INITIATE RU. This indicator is set based on the setting of the RPL bit indicating OPTCD=BACKUP (RPLBCKUP). If an Initiate is received specifying a backup XRF session and queue, it is rejected.

The RPL system-sense (SSENSEI), the system-sense modifier (SSENSMI), and the user-sense (USENSEI) can contain a more detailed explanation of the failure.

RTNCD	FDB2	Explanation
20(X'14')	126(X'7E')	Conflicting OPTCD on a macroinstruction request

One of the following conditions was detected:

- A TERMSESS macroinstruction has been issued with none or more than one of the following OPTCDs specified: COND, UNCOND, and UNBIND.
- A SETLOGON request has been issued with none or more than one of the following OPTCDs specified: HOLD, NPERSIST, PERSIST, QUIESCE, GNAMEADD, GNAMEDEL, GNAMESUB, START, and STOP.
- A SIMLOGON request has been issued with more than one of the following OPTCDs specified: QALL, QSESSLIM, and QNOTENAB.

RTNCD	FDB2	Explanation
20(X'14')	127(X'7F')	Policing error - non-APPC macroinstruction

An application program issued a non-APPCCMD macroinstruction to establish an LU 6.2 session, or issued a non-APPCCMD macroinstruction against a current LU 6.2 session.

RTNCD	FDB2	Explanation
20(X'14')	128(X'80')	SETLOGON not valid

You specified SETLOGON OPTCD=NPERSIST or PERSIST for an application that is not capable of persistence.

RTNCD	FDB2	Explanation
20(X'14')	129(X'81')	TERMSESS without OPTCD=UNBIND with session in a pending state

A TERMSESS macroinstruction is issued for a pending active session without specifying OPTCD=UNBIND.

RTNCD	FDB2	Explanation
20(X'14')	130(X'82')	Parameter length not valid

The length of an application-supplied dial parameter is not valid. Refer to "Application-supplied dial parameter control block (ASDP)" on page 119z/OS Communications Server: SNA Programming for a description of the valid lengths.

RTNCD	FDB2	Explanation
20(X'14')	131(X'83')	Subfield error

Either a subfield is not supported, or a combination of subfields that is not valid is specified. Refer to "Application-supplied dial parameter control block (ASDP)" on page 119z/OS Communications Server: SNA Programming for information about the valid subfields that can be specified.

RTNCD	FDB2	Explanation
20(X'14')	132(X'84')	NIBASDPA = 0

The value of NIBASDPA is 0. The NIBASDP indicator was on, indicating that the application is providing dial parameters; however, no address for the control block was given. This probably resulted from the application program passing an address that is not valid to the NIB.

RTNCD	FDB2	Explanation
20(X'14')	133(X'85')	Session must be restored

A SEND, RECEIVE, RESETSR, or SESSIONC request is rejected because it is issued for a session that is pending recovery. Use OPNDST OPTCD=RESTORE to restore the session and reissue the request.

RTNCD	FDB2	Explanation
20(X'14')	134(X'86')	Existing session prevents successful completion of this operation

One of the following applies:

- You issued CHANGE OPTCD=ENDAFFIN to terminate the association between your application program and the specified LU. At least one session exists between the specified LU and the application program; all sessions with the partner LU must be ended before the association can be terminated.
- You issued SETLOGON OPTCD=GNAMEADD to register your application as a generic resource, but a session exists already.

RTNCD	FDB2	Explanation
20(X'14')	135(X'87')	Resource name and generic name are the same

You attempted to issue either SETLOGON OPTCD=GNAMEADD, SETLOGON OPTCD=GNAMESUB, or SETLOGON OPTCD=GNAMEDEL using a generic name that was the same as the application network name; they must be different.

RTNCD	FDB2	Explanation
20(X'14')	136(X'88')	No association matching the given criteria exists.

You issued either CHANGE OPTCD=ENDAFFIN or INQUIRE OPTCD=SESSNAME, but the values specified in the NIB do not correspond to any known association.

RTNCD	FDB2	Explanation
20(X'14')	137(X'89')	Generic name not authorized

The generic name has not been authorized using a security management product such as RACF.

RTNCD	FDB2	Explanation
20(X'14')	138(X'8A')	Application program already registered

The application program is registered already as a generic resource, but with a different name.

RTNCD	FDB2	Explanation
20(X'14')	139(X'8B')	SETLOGON OPTCD=GNAMEDEL not valid

You used SETLOGON OPTCD=GNAMEDEL to deregister generic resources but VTAM determined that generic mapping does not exist; no VTAM message is issued.

RTNCD	FDB2	Explanation
20(X'14')	140(X'8C')	Network identifiers conflict for this generic resource.

This generic resource exists already with another network identifier.

RTNCD	FDB2	Explanation
20(X'14')	141(X'8D')	Simultaneous generic resource registration in progress

Two applications with the same application network name are simultaneously attempting to register a generic name.

RTNCD	FDB2	Explanation
20(X'14')	142(X'8E')	APPC capabilities conflict

All applications registering as generic resources must have the same APPC capabilities specified on their APPL statements.

RTNCD	FDB2	Explanation
20(X'14')	143(X'8F')	Deletion of VTAM affinity rejected

VTAM owns the affinity. Your application cannot delete it.

RTNCD	FDB2	Explanation
20(X'14')	144(X'90')	USERVAR conflict while registering generic resources

You issued SETLOGON OPTCD=GNAMEADD to register generic resources. VTAM detected a conflict (the generic resource exists already as a USERVAR name).

RTNCD	FDB2	Explanation
20(X'14')	145(X'91')	TSO GENERIC NAME CONFLICT

Either a non-TSO application is attempting to use the generic name already being used by TSO, or TSO is attempting to use the generic name already being used by a non-TSO application.

RTNCD	FDB2	Explanation
20(X'14')	146(X'92')	SETLOGON GNAMESUB FAILURE

A SETLOGON OPTCD=GNAMESUB macroinstruction failed for one of the following reasons:

- SETLOGON OPTCD=GNAMEADD was previously issued for this ACB.
- SETLOGON OPTCD=GNAMESUB was previously issued for this ACB.
- The application program network name specified in the VTAM node identification block (NIB) either was not found or was not an instance of the generic name specified in the NIB.

RTNCD	FDB2	Explanation
20(X'14')	147(X'93')	STOKEN not valid.

PROC=STOKEN is specified and the NIBSTKN field contains an invalid STOKEN.

RTNCD	FDB2	Explanation
20(X'14')	148(X'94')	No LU name passed.

No LU name was passed on the SETLOGON OPTCD=START or on the REQSESS.

RTNCD	FDB2	Explanation
20(X'14')	149(X'95')	No applicable RDTE found.

No RDTE was found that matched the LU name passed on the SETLOGON OPTCD=START or on the REQSESS.

RTNCD	FDB2	Explanation
20(X'14')	150(X'96')	Conflict with found RDTE.

An RDTE was found that matched the LU name passed on the SETLOGON OPTCD=START or on the REQSESS, but its characteristics or state was not appropriate.

SNA sense fields

When the application program or a logical unit receives an exception request, a negative response, or a Logical Unit Status (LUSTAT) request, the associated sense data includes information regarding the reason for the exception condition. The following three types of information describe the exception condition:

- System-sense information
- System-sense modifier information
- User-sense information.

System sense information indicates one of the five major classes of system-defined errors. The five major classes are described in Table 96 on page 680.

System-sense modifier information indicates one of many specific causes of the error indicated by the system-sense information. Like RTNCD and FDB2, the system-sense and its modifier information together form a specific type of error condition within a general class of error conditions. See the *SNA Formats* and z/OS Communications Server: IP and SNA Codes for more information.

User-sense information is generally used when the error condition is detected by the user-written program itself. In general, no particular codes or values are defined by IBM to indicate types of errors. The logical unit must generate its own user-sense information that is understood by other logical units.

The SNA defined values for the sense fields can be found in *SNA Formats* Additional information is contained in *SNA Format and Protocol Reference Manual: Architectural Logic,* and *SNA Sessions between Logical Units*

These three types of sense information—system, system modifier, and user—are set in RPL fields. Three fields (one for each type of sense information) are set by the application program when it sends a negative response or LUSTAT request to the logical unit. Three other fields are set by VTAM when the application program receives an exception request, a negative response, or LUSTAT request from the logical unit. These are the names of the six fields, as they would be used on a manipulative or RPL macroinstruction:

Sense information	Received by the application program	Sent from the application program
System-sense information	SSENSEI	SSENSEO
System-sense modifier information	SSENSMI	SSENSMO
User-sense information	USENSEI	USENSEO

System-sense information

The values that are set in the system-sense field are predefined by IBM. These values are as follows (the operands shown here are those used with a MODCB or TESTCB macroinstruction; the corresponding hexadecimal value is also shown in parentheses):

Table 96. Sense field values

System-sense values	Meaning
SSENSEI=PATH (X'80')	A path error occurred. The RU could not be delivered to the intended receiver because of a physical problem in the network path or an error in the system-supplied transmission header that accompanied the RU. If no recovery action is possible, terminate the session with the logical unit.
SSENSEI=CPM (X'40')	An unrecoverable request header error occurred.
SSENSEO=CPM (X'40')	The sender did not correctly enforce the current session protocols. Terminate the session with the logical unit.
SSENSEI=STATE (X'20') SSENSEO=STATE (X'20')	A state error occurred in the application program's or logical unit's use of sequence numbers, chaining indicators, bracket indicators, or change-direction indicators. A state error can also occur when a data-flow-control request is issued, data is sent after a Clear request, or when a session-control request is issued before a Clear request. This type of error is recoverable; use Clear, STSN, and SDT requests.
SSENSEI=FI (X'10') SSENSEO=FI (X'10')	A request error occurred. The application program or logical unit cannot handle the request because the request itself is not valid. This error might be recoverable.
SSENSEI=RR (X'08') SSENSEO=RR (X'08')	A request reject occurred. The request was delivered to the intended receiver; it was correctly interpreted, but not handled by the receiver. This might be a recoverable condition.

Appendix C. Summary of control requests and indicators

This appendix contains tables (Table 97 through Table 104 on page 691) that summarize the SNA control requests and indicators sent and received by VTAM application programs. The tables summarize:

- Normal-flow data-flow-control requests
- Expedited-flow data-flow-control requests
- Session-control requests
- Change-direction indicator
- Bracket indicators.

For normal-flow data-flow-control requests, expedited-flow data-flow-control requests, and session-control requests, the following information for sending each request is provided:

- The purpose of each request
- Who can send it
- The macroinstruction used by an application program to send it
- The RU type of the request
- The next action to be taken by the sender

The following information for receiving each request is also provided:

- Who can receive it
- How it is received by the application program
- Who sends the response to the request
- The next action to be taken by the receiver

For the change-direction and bracket indicators, the information is summarized with the entry for each indicator providing information on both sending and receiving the indicator.

The ability to send and receive the indicators and control requests described in this appendix is determined by the session parameter agreed on by the application program and the logical unit (LU) when the session is established. For detailed information on a session parameter, see Appendix F, "Specifying a session parameter," on page 793.

Table 97. Summary of sending normal-flow data-flow-control requests

Request sent	Function	Who can send	Macroinstruction used by application program to send	RU type	Next action by sender
Bid	Asks receiver for permission to begin a bracket.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=BID	DFSYN	Expects response from receiver. Response indicates whether the sender can begin a bracket.
Bracket Initiation Stopped (BIS)	Tells the receiver that the sender will not begin any new brackets.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=BIS	DFSYN	Expects response from receiver. Refrains from beginning any new brackets.

© Copyright IBM Corp. 2000, 2013

Table 97. Summary of sending normal-flow data-flow-control requests (continued)

Request sent	Function	Who can send	Macroinstruction used by application program to send	RU type	Next action by sender
Cancel	Tells receiver to purge request of incomplete chain it is receiving.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=BIS	DFSYN	Expects response from receiver. Positive response indicates that chain requests have been purged.
CHASE	Tells receiver to send response to any data request or normal-flow request it has not yet responded to.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=CANCEL	DFSYN	Expects response from receiver. When response to CHASE request is received, the sender of the request knows that all normal-flow responses are accounted for.
Logical Unit Status (LUSTAT)	Informs receiver of a condition encountered at the sender's end of the session. Codes indicating reason for sending the request are placed in the SSENSEO, SSENSMO and USENSEO fields of the RPL.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=LUS	DFSYN	Expects response from receiver. Subsequent action depends on the sense information sent in LUSTAT.
Quiesce Complete (QC)	Tells receiver that the sender has quiesced itself (as the result of receipt of a Quiesce at End of Chain request) and will not send any normal-flow requests until released.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=QC	DFSYN	Expects response from receiver. Refrains from sending any normal-flow requests until a Release Quiesce request is received.
Ready to Receive (RTR)	Tells the receiver that the receiver can now send a request to begin a bracket.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=RTR	DFSYN	Expects response from receiver. After receiving the response, an application program issues RECEIVE RTYPE=DFSYN to receive a normal-flow request with the begin-bracket indicator set on.

Notes:

An application program can receive the response in one of the following ways, depending on how the program is coded:

- By specifying POST=RESP in SEND (SEND is not completed until response is received)
- By issuing RECEIVE RTYPE=RESP (a RESP response)
- In a RESP exit routine
- By issuing RECEIVE RTYPE=DFSYN (a DFSYN response).

(See "How requests and responses are exchanged" on page 157 for further details about controlling the handling of normal-flow requests.)

Table 98. Summary of receiving normal-flow data-flow-control requests

Request received	Who can receive	How received by application program	Who sends response if receiver is a VTAM application program	Next action by receiver
Bid	Primary or secondary logical unit	RECEIVE RTYPE=DFSYN CONTROL field in RPL will contain BID.	Application program	Sends positive response to indicate bidder can start a bracket. Sends a negative response to deny permission to start a bracket. Application program sends response with SEND ....,STYPE=RESP, CONTROL=BID, RESPOND=(*response operands*).
Bracket Initiation Stopped (BIS)	Primary or secondary logical unit	RECEIVE RTYPE=DFSYN CONTROL field in RPL will contain BIS.	Application program	Sends response to Bracket Initiation chains that have been received. Then sends positive response. Application SEND ...,STYPE=RESP, CONTROL=BIS, RESPOND=(*response operands*).
Cancel	Primary or secondary logical unit	RECEIVE RTYPE=DFSYN CONTROL field in RPL will contain CANCEL.	Application program	Purges any requests of incomplete chains that have been received. Then sends positive response. Application program sends response with SEND ...,STYPE=RESP, CONTROL=CANCEL, RESPOND=(*response operands*).
Chase	Primary or secondary logical unit	RECEIVE RTYPE=DFSYN CONTROL field in RPL will contain CHASE.	Application program	If any responses to previously received data requests or normal-flow requests have not been sent, sends those responses. Then sends response to Chase request. Application program sends response to Chase request with SEND ...,STYPE=RESP, CONTROL=CHASE, RESPOND=(*response operands*).

Table 98. Summary of receiving normal-flow data-flow-control requests (continued)

Request received	Who can receive	How received by application program	Who sends response if receiver is a VTAM application program	Next action by receiver
Logical Unit Status (LUSTAT)	Primary or secondary logical unit	RECEIVE RTYPE=DFSYN CONTROL field in RPL will contain LUS. Codes indicating reason for the request are in the SSENSEI, SSENSMI, and USENSEI fields of the RPL.	Application program	Examines codes in SSENSEI, SSENSMI, and USENSEI fields of RPL and takes action based on those codes. Then sends response to LUSTAT request. Application program sends response with SEND ...,STYPE=RESP, CONTROL=LUS, RESPOND=(*response operands*).
Quiesce Complete (QC)	Primary or secondary logical unit	RECEIVE RTYPE=DFSYN CONTROL field in RPL will contain QC.	Application program	Sends response to Quiesce Complete request. Application program sends response with SEND ...,STYPE=RESP, CONTROL=LUS, RESPOND=(*response operands*).
Ready to Receive (RTR)	Primary or secondary logical unit	RECEIVE RTYPE=DFSYN CONTROL field in RPL will contain RTR.	Application program	Sends response to Ready to Receive request by using SEND ....,STYPE=RESP, CONTROL=RTR, RESPOND=(*response operands*). Then sends a request that includes BRACKET=BB.

Note:

If the application program sends a negative response, the SSENSEO, SSENSMO, and USENSEO fields are used.

Table 99. Summary of sending expedited-flow-control requests

Request sent	Function	Who can send	Macroinstruction used by application program	RU type	Next action by sender
Quiesce at End of Chain (QEC)	Tells the receiver to quit sending normal-flow requests now, or, if chaining, at the end of the chain being sent.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=QEC	DFASY	Expects response from receiver. After receiving positive response, awaits Quiesce Complete.
Release Quiesce	Tells the receiver that it can now resume sending normal-flow requests.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=RELQ	DFASY	Expects response from receiver. After receiving positive response, prepares to receive normal-flow input. An application program issues RECEIVE RTYPE=DFSYN.

Table 99. Summary of sending expedited-flow-control requests (continued)

Request sent	Function	Who can send	Macroinstruction used by application program	RU type	Next action by sender
Request Shutdown (RSHUTD)	Asks the primary logical unit to terminate this session with the secondary logical unit.	Secondary logical unit only	SEND STYPE=REQ, CONTROL=RSHUTD	DFASY	Expects response from receiver. Response indicates that request has been properly received. Prepares for session termination.
Shutdown Complete (SHUTC)	Tells the primary logical unit that shutdown operations (requested previously in a Shutdown request from the primary logical unit) have been completed.	Secondary logical unit only	SEND STYPE=REQ, CONTROL=SHUTC	DFASY	Expects response from receiver. Response indicates that request has been properly received.
Shutdown (SHUTD)	Tells the secondary logical unit to quiesce itself and to perform all preparations for Shutdown.	Primary logical unit only	SEND STYPE=REQ, CONTROL=SHUTD	DFASY	Expects response from receiver. Response indicates that request has been properly received. Then, expects to receive Shutdown Complete request from the secondary logical unit.
Signal	Passes 4 bytes of Signal information with an agreed-upon meaning. Signal information is placed in the SIGDATA field of the RPL.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=SIGNAL	DFASY	Expects response from receiver. Response indicates that request has been properly received.
Stop Bracket Initiation (SBI)	Tells receiver not to begin any new brackets.	Primary or secondary logical unit	SEND STYPE=REQ, CONTROL=SBI	DFASY	Expects response from receiver. Response indicates that request has been properly received.

Note:

An application program receives the response in SEND (SEND is not completed unit response is received).

Table 100. Summary of receiving expedited-flow data-flow-control requests

Request received	Who can receive	How received by application program	Who sends response if receiver is a VTAM application program	Next action by receiver
Quiesce at End of Chain (QEC)	Primary or secondary logical unit	Either RECEIVE RTYPE=DFASY or in DFASY exit routine. CONTROL field in RPL will contain QEC.	Either VTAM or the program sends the response.	Halts sending of normal-flow requests immediately or at end of chain. Then sends Quiesce Complete (QC) request to sender of QEC request.

Table 100. Summary of receiving expedited-flow data-flow-control requests (continued)

Request received	Who can receive	How received by application program	Who sends response if receiver is a VTAM application program	Next action by receiver
Release Quiesce (RELQ)	Primary or secondary logical unit	Either RECEIVE RTYPE=DFASY or in DFASY exit routine. CONTROL field in RPL contains RELQ.	Either VTAM or the program sends the response.	Sends a normal-flow request to sender of RELQ request, if desired.
Request Shutdown (RSHUTD)	Primary logical unit only	Either RECEIVE RTYPE=DFASY or in DFASY exit routine. CONTROL field in RPL contains RSHUTD.	Either VTAM or the program sends the response.	Terminates the session with the secondary logical unit.
Shutdown Complete (SHUTC)	Primary logical unit only	Either RECEIVE RTYPE=DFASY or in DFASY exit routine. CONTROL field in RPL will contain SHUTC.	Either VTAM or the program sends the response.	Issues a Chase request (if permitted by FM profile) to ensure that all responses have been received. Then ends the session with the secondary logical unit.
Shutdown (SHUTD)	Secondary logical unit only	Either RECEIVE RTYPE=DFASY or in DFASY exit routine. CONTROL field in RPL will contain SHUTD.	Either VTAM or the program sends the response.	If necessary, stops normal-flow transmission to the primary logical unit. Performs all preparations for shutdown. Then sends the Shutdown Complete request to the primary logical unit.
Signal	Primary or secondary logical unit	Either RECEIVE RTYPE=DFASY or in DFASY exit routine. CONTROL field in RPL contains SIGNAL. Four byes of information are in the SIGDATA field of the RPL.	Either VTAM or the program sends the response.	Depends on the contents of the Signal information.
Stop Bracket Initiation (SBI)	Primary or secondary logical unit	Either RECEIVE RTYPE=DFASY or in DFASY exit routine. CONTROL field in RPL will contain SBI.	Either VTAM or the program sends the response.	Sends a Bracket Initiation Stopped (BIS) request to SBI sender and then refrains from initiating any new brackets.

Table 100. Summary of receiving expedited-flow data-flow-control requests (continued)

Request received	Who can receive	How received by application program	Who sends response if receiver is a VTAM application program	Next action by receiver

Notes:

The responder to expedited-flow data-flow-control request is determined by the setting of a PROC option in the NIB when the session is established:

- If PROC=APPLRESP was specified in the NIB, the application program sends the response using SEND...,STYPE=RESP,CONTROL=*request code* or *received request*, RESPOND=(*response operands*). If a negative response is sent the SSENSEO, SSENSMO, and USENSEO fields are used.
- If PROC=SYSRESP was specified in the NIB, VTAM automatically sends the response before presenting the request to the application program.

The session-control requests, described in the following table, control session-related functions and are sent separately from normal- and expedited-flow-control requests and their responses.

Table 101. Summary of sending session-control requests

Request sent	Function	Who can send	Macroinstruction used by application program to send	Next action by sender
Bind Session (BIND)	Informs the receiver that the sender wants to go into session with the receiver. A session parameter is sent as part of the Bind Session request.	Primary logical unit	Indirectly, by issuing the OPNDST macroinstruction	VTAM handles response and does not complete the OPNDST until response is received. Positive response causes VTAM to complete setting up the session. Negative response negates the session. After positive response, either VTAM or the PLU application program sends the Start Data Traffic request (if required by TS profile).
Clear	For certain TS profiles, tells VTAM and the receiver to stop sending data and data-flow-control requests and responses. Causes VTAM to discard any requests and responses still in the network and not yet delivered. Resets outbound and inbound sequence numbers at both ends of the session to 0.	Primary logical unit	SESSIONC STYPE=REQ, CONTROL=CLEAR	Response reflected in RPL on completion of SESSIONC macroinstruction. VTAM handles response.

Table 101. Summary of sending session-control requests (continued)

Request sent	Function	Who can send	Macroinstruction used by application program to send	Next action by sender
Request Recovery (RQR)	Informs primary logical unit that recovery action is needed.	Secondary logical unit	SESSIONC STYPE=REQ, CONTROL=RQR	In an SLU application program, response reflected in RPL on completion of SESSIONC macroinstruction. After receiving positive response, awaits next request from the primary logical unit (usually the Clear request).
Set and Test Sequence Numbers (STSN)	Exchanges information with secondary logical unit to allow sequence numbers to be determined or set or both.	Primary logical unit	SESSIONC STYPE=REQ, CONTROL=STSN and settings in IBSQAC, OBSQAC, IBSQVAL, and OBSQVAL fields	Response reflected in RPL on completion of SESSIONC macroinstruction. Tests IBSQAC, OBSQAC, IBSQVAL, and OBSQVAL fields to determine answers to action codes and values sent in the request.
Start Data Traffic (SDT)	Informs secondary logical unit that session setup or recovery is complete and flow of data and data-flow-control requests and responses can begin.	Primary logical unit	SESSIONC STYPE=REQ, CONTROL=SDT. VTAM sends command at beginning of session if SDT=SYSTEM was set in NIB when the session was established.	Depending on the session parameter, you may send first requested or wait for secondary logical unit to send a request.
Unbind Session (UNBIND)	Informs VTAM and the receiver that the session is being terminated.	Primary logical unit, secondary logical unit, or other network component	PLU application program can send indirectly by issuing CLSDST or TERMSESS.	Continues communications on sessions with other logical units, or closes program.

Table 102. Summary of receiving session-control requests

Request received	Who can receive	How received by application program	Who sends response if receiver is a VTAM application program	Next action by receiver
Bind Session (BIND)	Secondary logical unit	In SCIP exit	Application program [1]	Examines a session parameter in BIND request and determines whether the complete set of parameters is acceptable. If acceptable, sends positive response. (For SLU application program, positive response results from issuance of the OPNSEC macroinstruction.) If not acceptable, sends negative response. (SLU application program sends negative response with SESSIONC ...,STYPE=RESP, CONTROL=BIND, RESPOND=(*response operands*).)
Clear	Secondary logical unit	In SCIP exit	VTAM	Stops sending requests and responses, and awaits next request from the primary logical unit.
Request Recovery (RQR)	Primary logical unit	In SCIP exit	VTAM	Initiates recovery action, usually by sending the Clear request followed by a Set and Test Sequence Numbers request and then a Start Data Traffic request.
Set and Test Sequence Numbers (STSN)	Secondary logical unit	In SCIP exit	Application program [1]	Examines action codes and sequence number values provided with the request. Prepares answering action codes and values and puts them in IBSQAC, OBSQAC, IBSQVAL, and OBSQVAL fields. Then, sends response with SESSIONC ...,STYPE=RESP, CONTROL=STSN.
Start Data Traffic (SDT)	Secondary logical unit	In SCIP exit	Depending on the SDT field in the NIB used during OPNSEC processing, either the secondary application program [1] or VTAM may respond [2]	After response is sent, depending on session parameter, you may send first request or wait for primary logical unit to send request.
Unbind Session (UNBIND)	Primary logical unit or secondary logical unit	In SCIP exit [3]	VTAM	Continues communication on sessions with other logical units, or closes program.

Table 102. Summary of receiving session-control requests (continued)

Request received	Who can receive	How received by application program	Who sends response if receiver is a VTAM application program	Next action by receiver

Notes:

1. If the application program sends a negative response, the SSENSEO, SSENSMO, and USENSEO fields are used.
2. If the secondary application program specified SDT=APPL for the NIB used in OPNSEC processing, the secondary application program responds. Otherwise (SDT=SYSTEM), VTAM responds.
3. Alternatively, for a PLU application program, the NSEXIT or LOSTERM exit routines may be invoked. See "Session outage notification (SON) codes on UNBIND" on page 93.

Change-direction indicator

The change-direction indicator can be set on (designated CD) in a normal-flow data request or in a Cancel, Chase, Quiesce Complete, or LU Status request. The request containing the CD must be a single-request chain (CHAIN=ONLY) or the last request in a chain.

Table 103. Summary of change-direction indicator

Indicator	Function	PLU or SLU application program can send/receive	Macroinstruction used or RPL field set	RU type	Next action expected
Change Direction (CD)	Tells the receiver that it may now send.	Send	SEND CHNGDIR=CMD	DFSYN	Start receiving from the opposite end of the session.
		Receive	CHNGDIR field in RPL contains CMD	DFSYN	Start sending from the opposite end of the session.

Bracket indicators

The normal-flow control requests Bid and Ready to Receive are used by the VTAM application program to determine whether it can send a request with the begin-bracket indicator set on (designated BB).

BB and CEB can be sent in a data request or an LU Status request. The end-bracket indicator can be set on (designated EB) in a data request or in a Cancel, Chase, Quiesce Complete, or LU Status request. The request containing the BB or EB must be a single-request chain (CHAIN=ONLY) or the first request of a chain. The request containing the CEB must be a single-request chain or the last request in a chain.

Table 104. Summary of bracket indicators

Indicator	Function	PLU or SLU application program can send/receive	Macroinstruction used or RPL field set	RU type	Next action expected
Begin Bracket (BB)	Indicates first chain in a bracket.	Send	SEND Bracket=BB	DFSYN	Continues to send or waits to receive, according to user conventions.
		Receive	BRACKET field in RPL contains BB	DFSYN	Accept or reject the request to start a bracket.
End Bracket (EB)	Indicates last chain in a bracket.	Send	SEND BRACKET=EB	DFSYN	Attempts to start a new bracket, or waits for other LU to start a bracket, according to user conventions.
Receive	BRACKET field in RPL contains EB	DFSYN	Attempts to start a new bracket, or waits for other LU to start a bracket, according to user conventions.		
Conditional End Bracket (CEB)	Indicates last chain in a bracket.	Send	SEND BRACKET=CEB	DFSYN	Attempts to start a new bracket, or waits for other LU to start a bracket, according to user conventions.
Receive	BRACKET field in RPL contains CEB	DFSYN	Attempts to start a new bracket, or waits for other LU to start a bracket, according to user conventions.		

Appendix D. Request and response exchanges for typical communication operations

This appendix contains diagrams that show the sequences in which requests and responses are exchanged to perform typical data communication operations using VTAM. The diagrams can be useful in coding application programs that perform the operations.

Note: A ***PLU application program*** is a VTAM application program that acts as the PLU in the session, and an ***SLU application program*** is a VTAM application program that acts as the SLU in the session.

Figure 107 on page 694 through Figure 127 on page 714 are oriented primarily toward communication between a PLU application program and a device-type logical unit, although some of these diagrams apply also when the SLU is a SLU application program. In Figure 107 on page 694 through Figure 127 on page 714, the “reads” and “writes” shown in the “Logical Unit” column represent logic that can be performed by a control program in the logical unit, a user-written program that operates in the logical unit, or both. It is a general representation of the input and output from the logical unit. The PLU application program's side of the exchange is shown in more detail.

Figure 128 on page 715 through Figure 142 on page 728 are oriented toward operations between a PLU application program and a SLU application program.

Figure 143 on page 729 and Figure 144 on page 730 illustrate a session between the VTAM SSCP and an application program over the CNM interface.

In any diagram showing a negative response being sent to an application program, a SYNAD exit routine might be scheduled with an exception condition return code. SYNAD exit routines are not illustrated in this appendix. See “Coding LERAD and SYNAD exit routines” on page 290 for details.

© Copyright IBM Corp. 2000, 2013

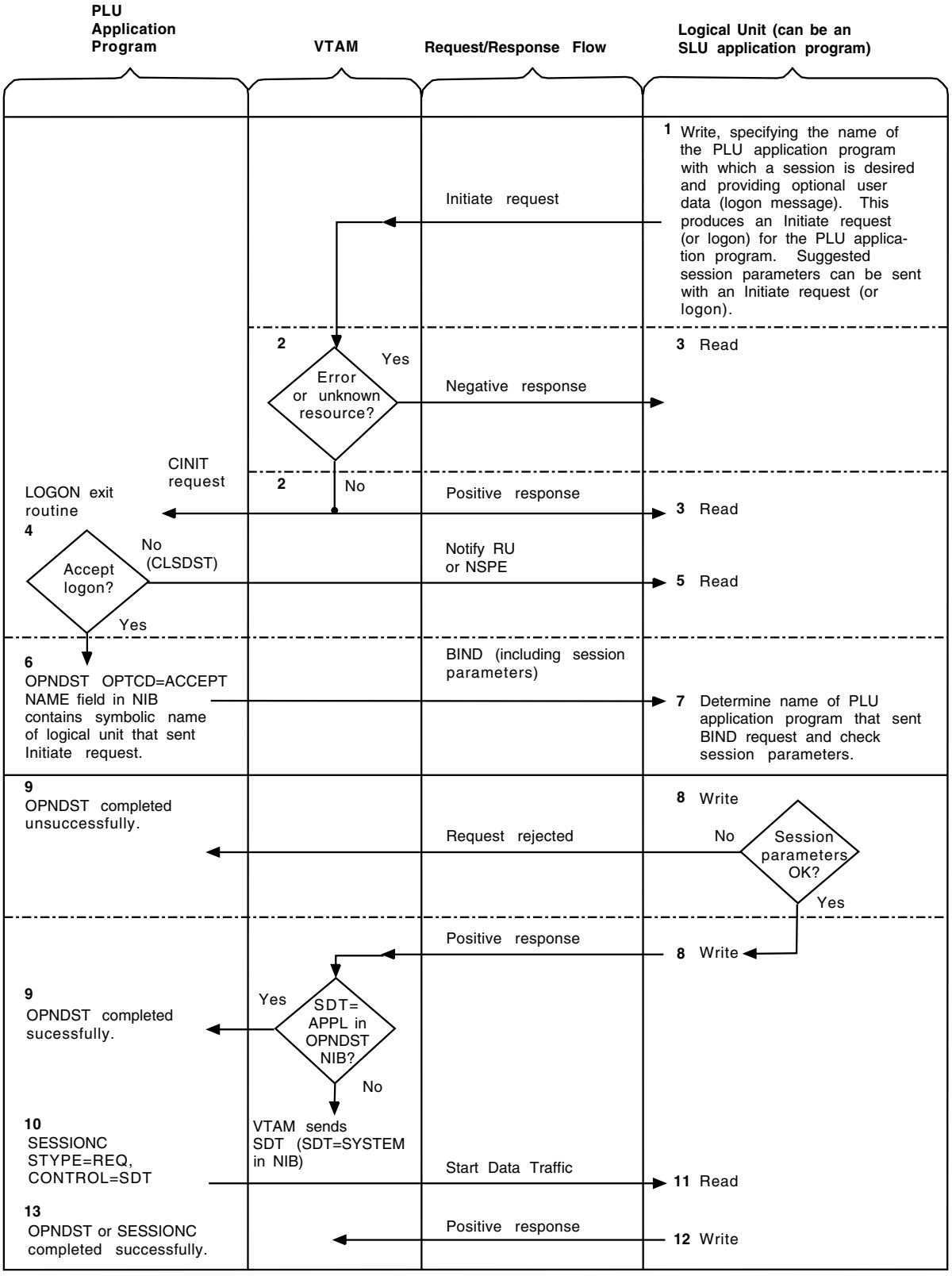


Figure 107. Device-type LU initiates and establishes a session with a PLU application program.

For initiating and establishing a session from a SLU application program, see Figure 128 on page 715 and Figure 129 on page 716.

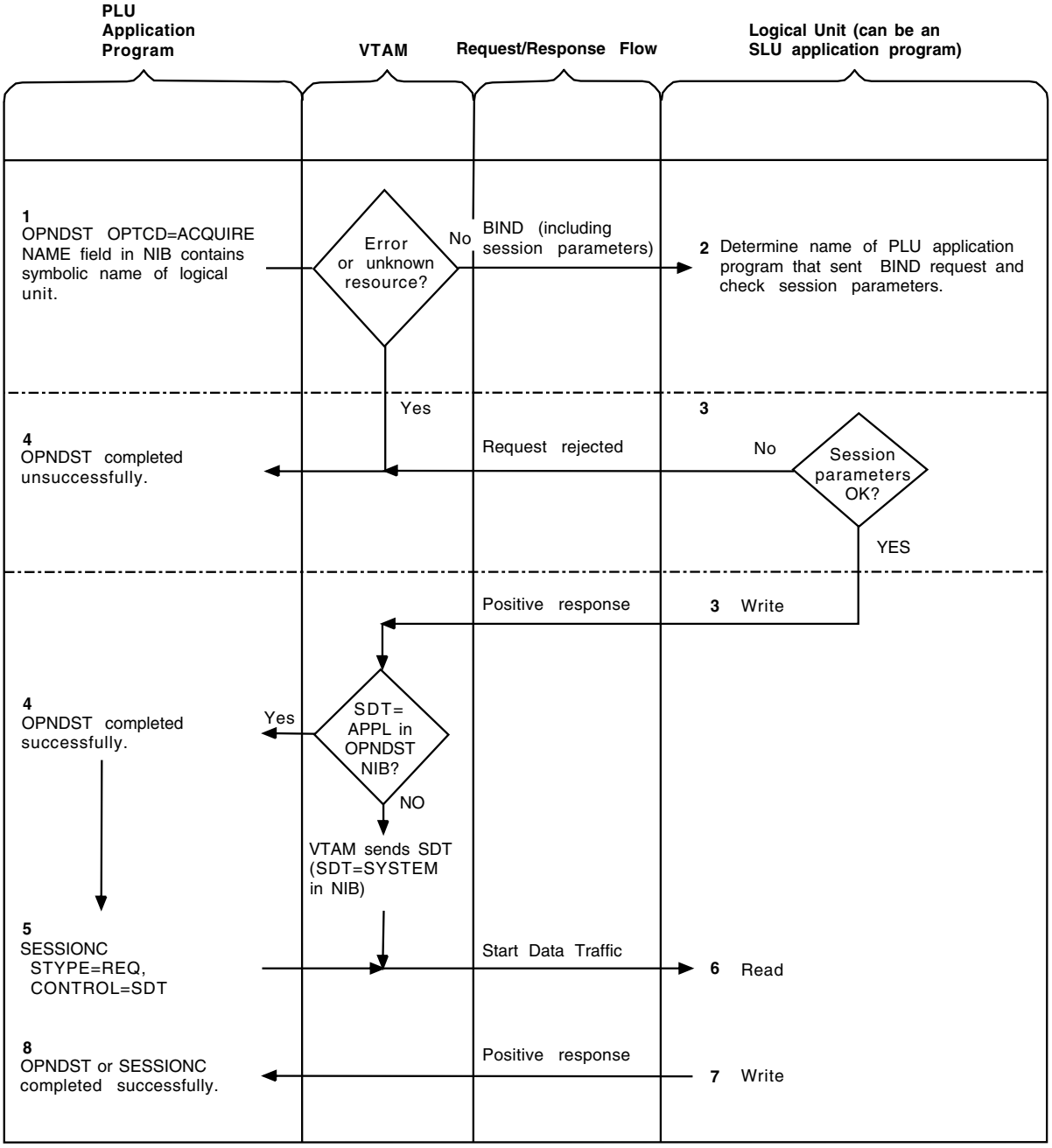


Figure 108. PLU application program acquires (initiates and establishes) a session with a device-type LU.

For initiating a session from a PLU to a SLU application program, see Figure 130 on page 717 and Figure 131 on page 718.

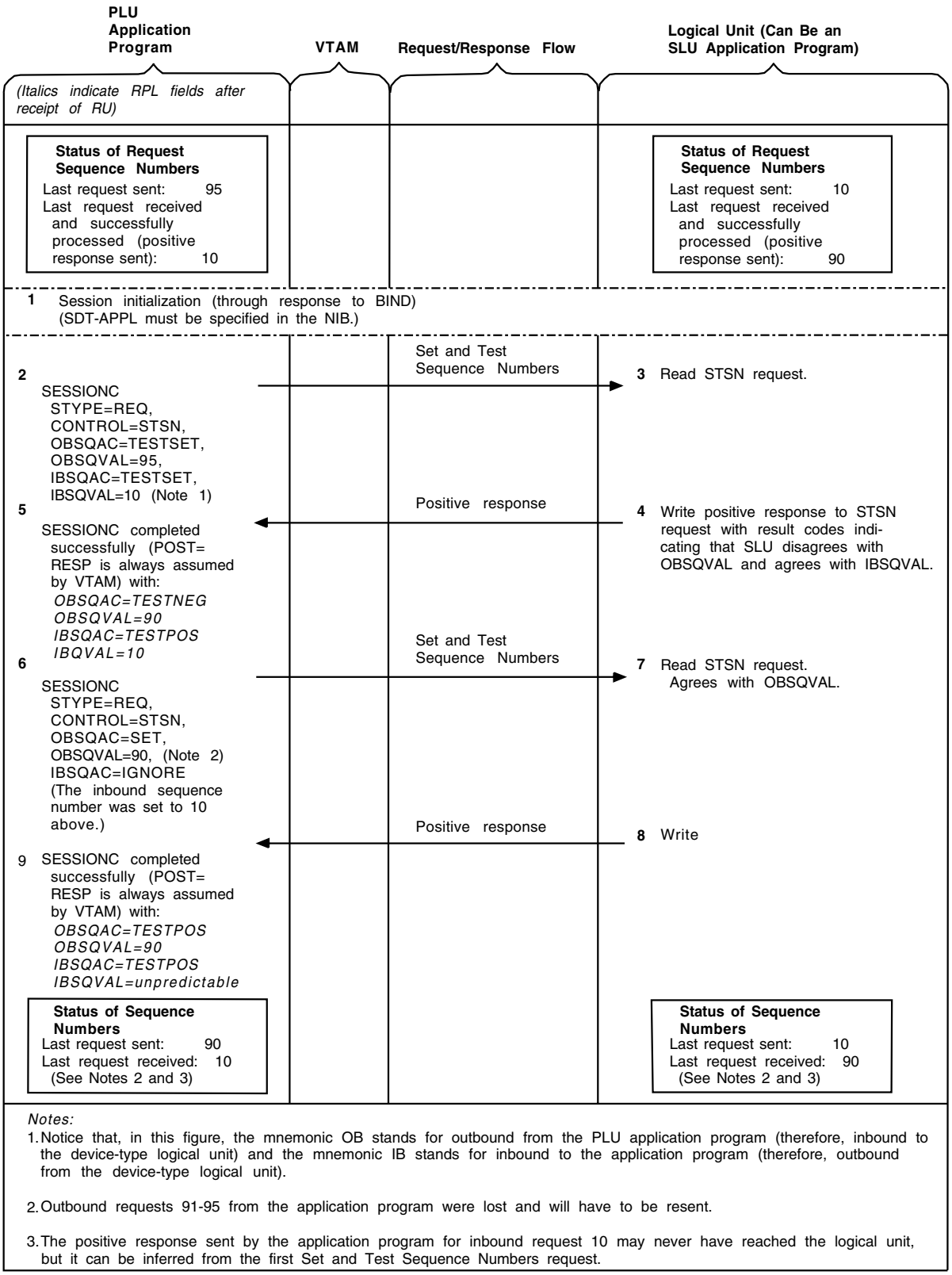
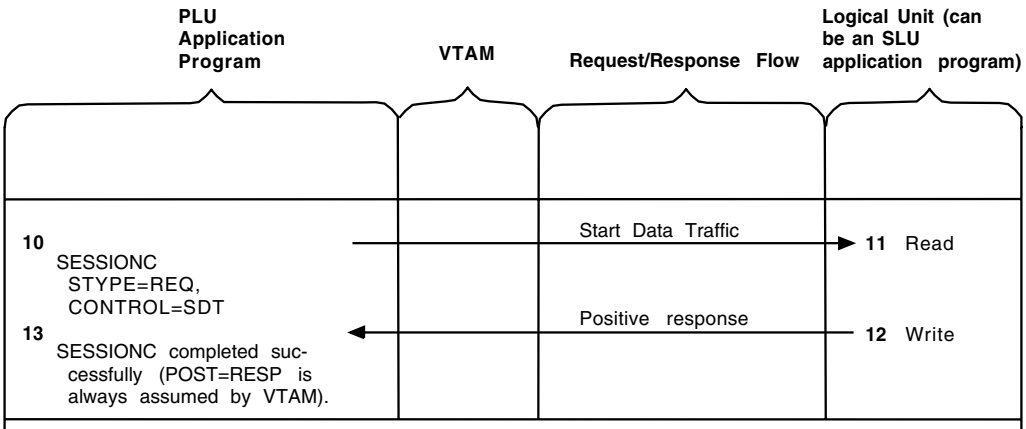


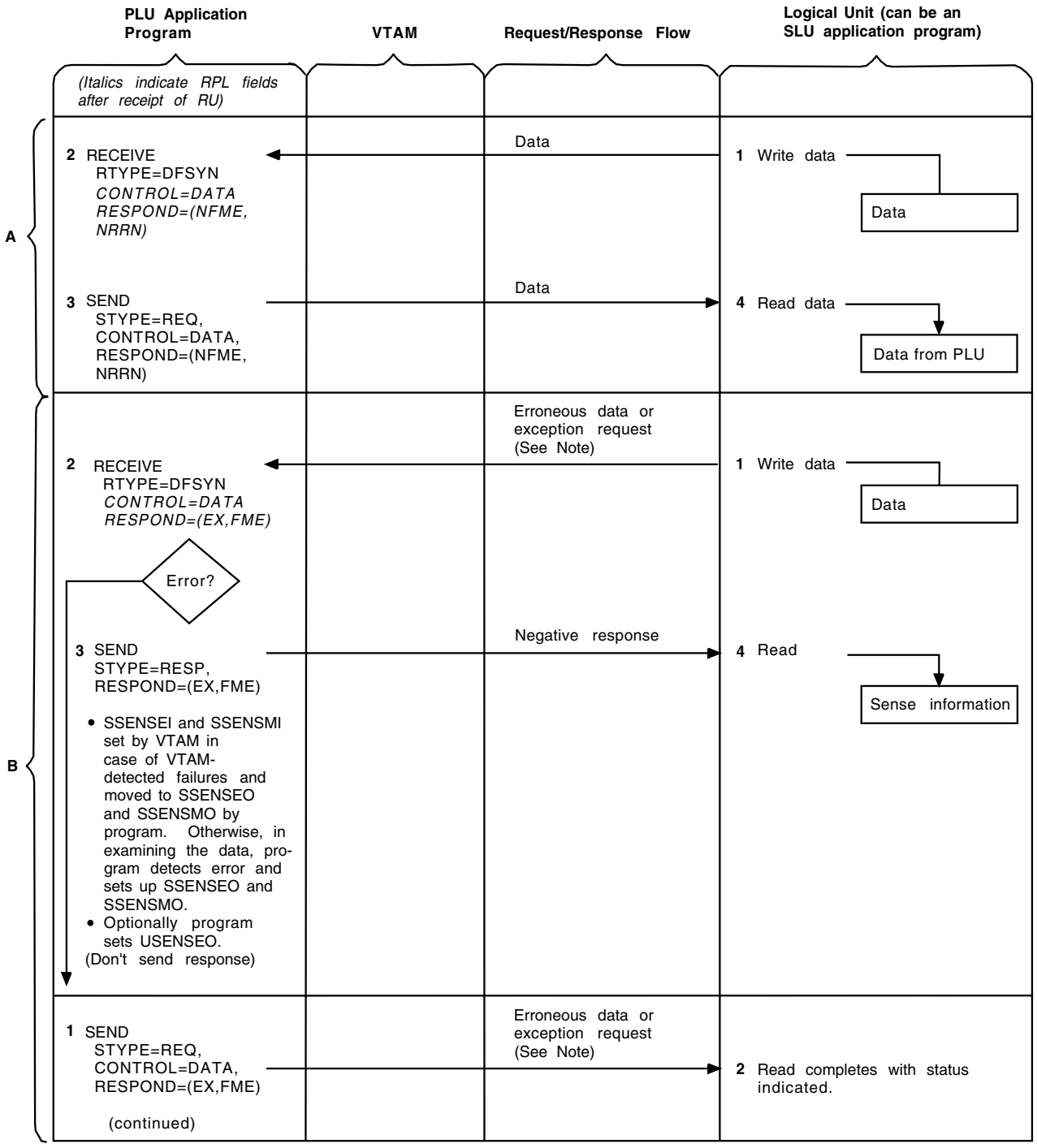
Figure 109. After a warm start, a PLU application program reestablishes a session and resynchronizes sequence numbers (Part 1 of 2)



Notes:

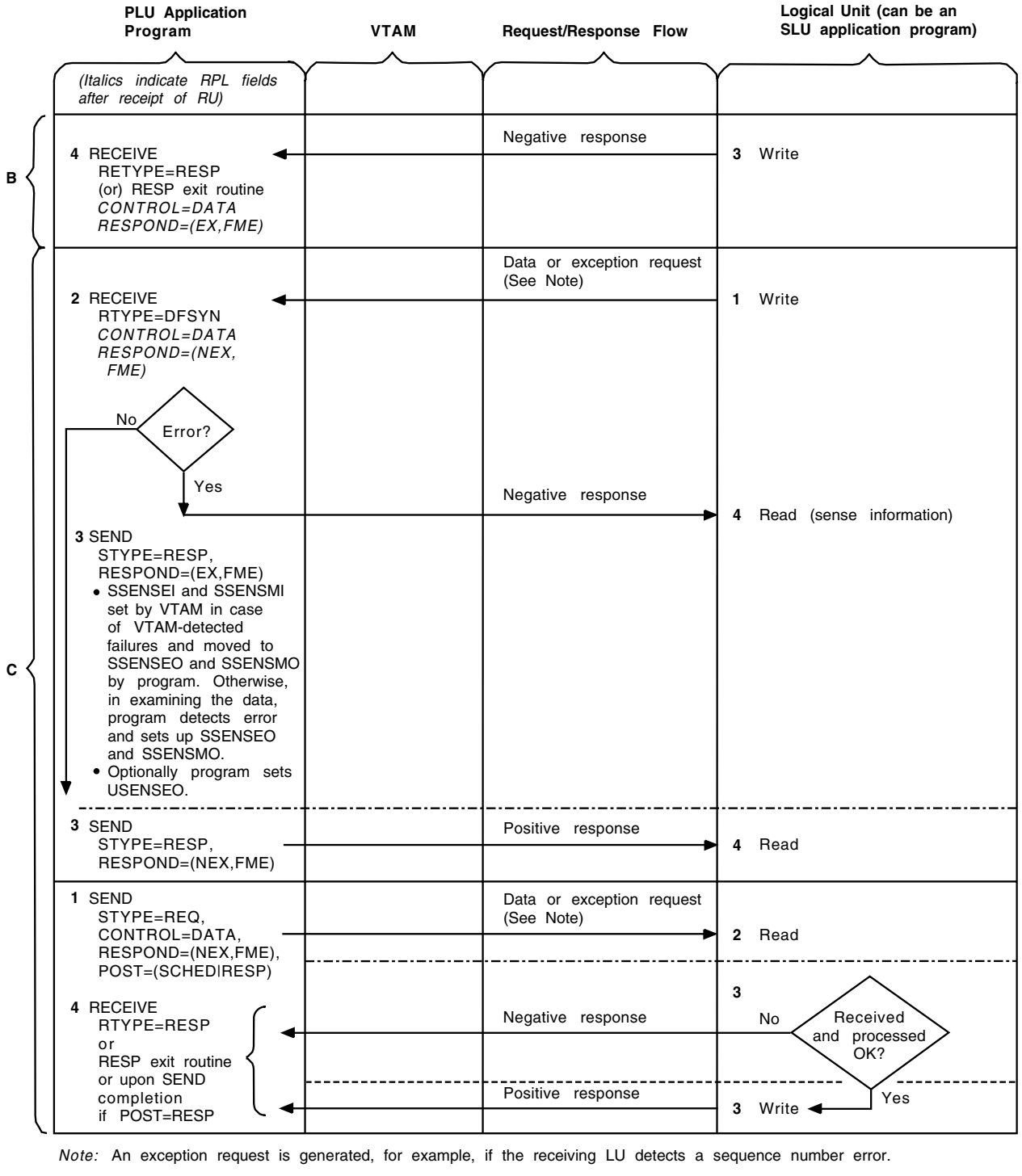
1. Notice that, in this figure, the mnemonic OB stands for outbound from the PLU application program (therefore, inbound to the device-type logical unit) and the mnemonic IB stands for inbound to the application program (therefore, outbound from the device-type logical unit).
2. Outbound requests 91-95 from the application program were lost and will have to be resent.
3. The positive response sent by the application program for inbound request 10 may never have reached the logical unit, but it can be inferred from the first Set and Test Sequence Numbers request.

Figure 110. After a warm start, a PLU application program reestablishes a session and resynchronizes sequence numbers (Part 2 of 2)



Note: An exception request is generated, for example, if the receiving LU detects a sequence number error.

Figure 111. PLU application program and a secondary logical unit exchange data (Part 1 of 3)



Note: An exception request is generated, for example, if the receiving LU detects a sequence number error.

Figure 112. PLU application program and a secondary logical unit exchange data (Part 2 of 3)

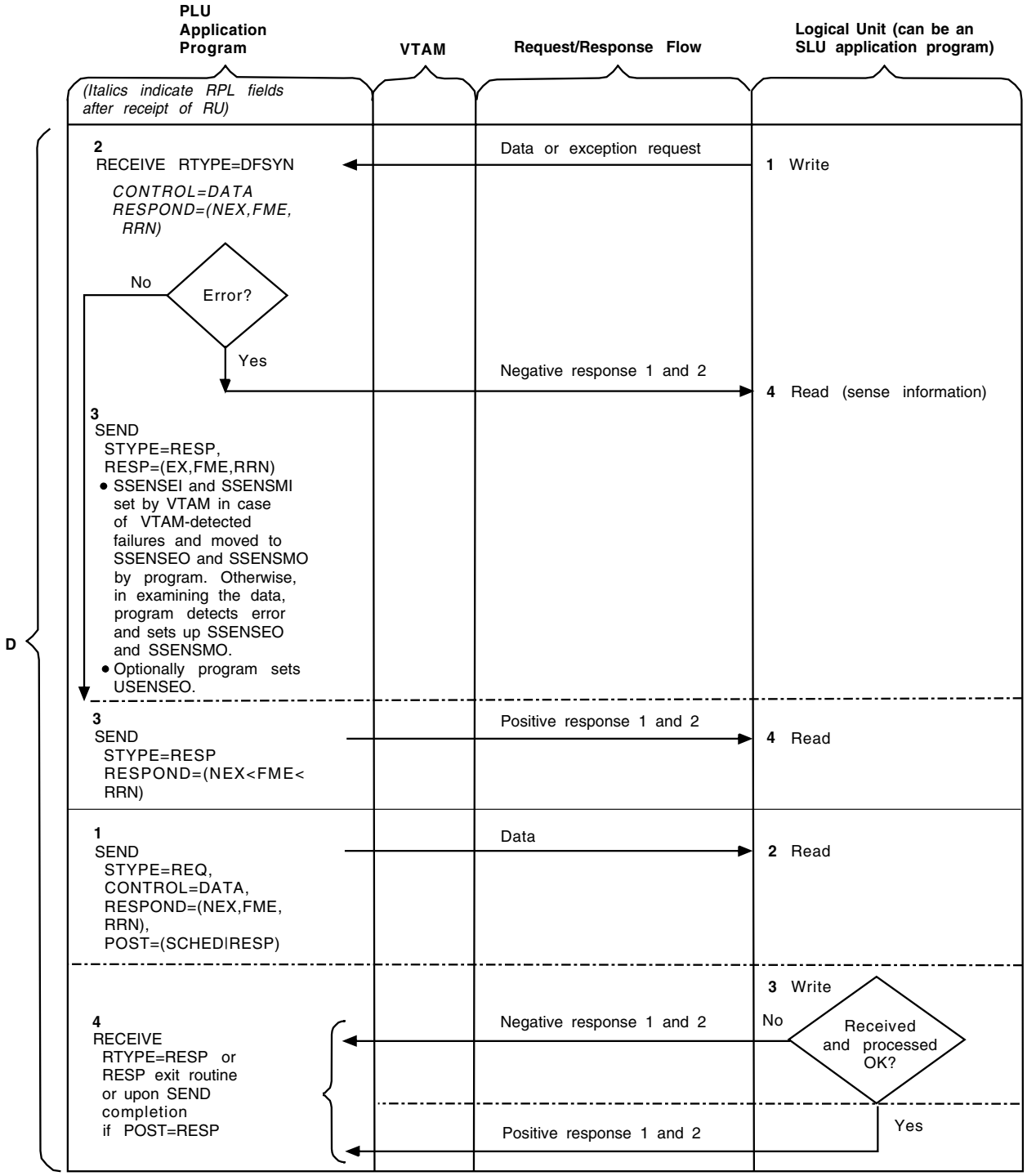


Figure 113. PLU application program and a secondary logical unit exchange data (Part 3 of 3 about sections A, B, C and D)

A With no responses

B With negative responses only if an exception occurs

C With definite response 1 (positive or negative)

D With definite responses 1 and 2 sent at the same time.

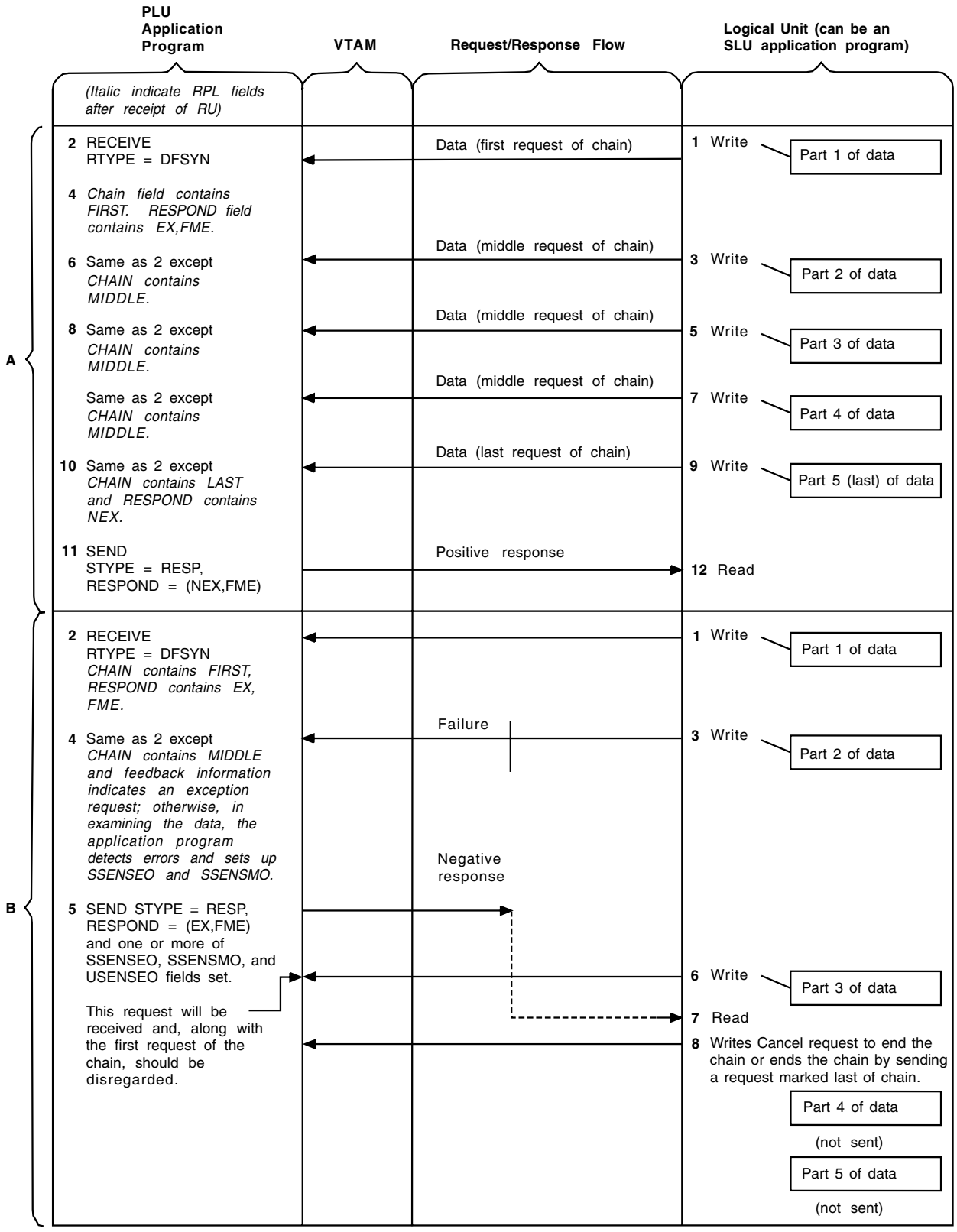


Figure 114. Logical unit sends a chain of data to the PLU application program

A Without a negative response

B With a negative response.

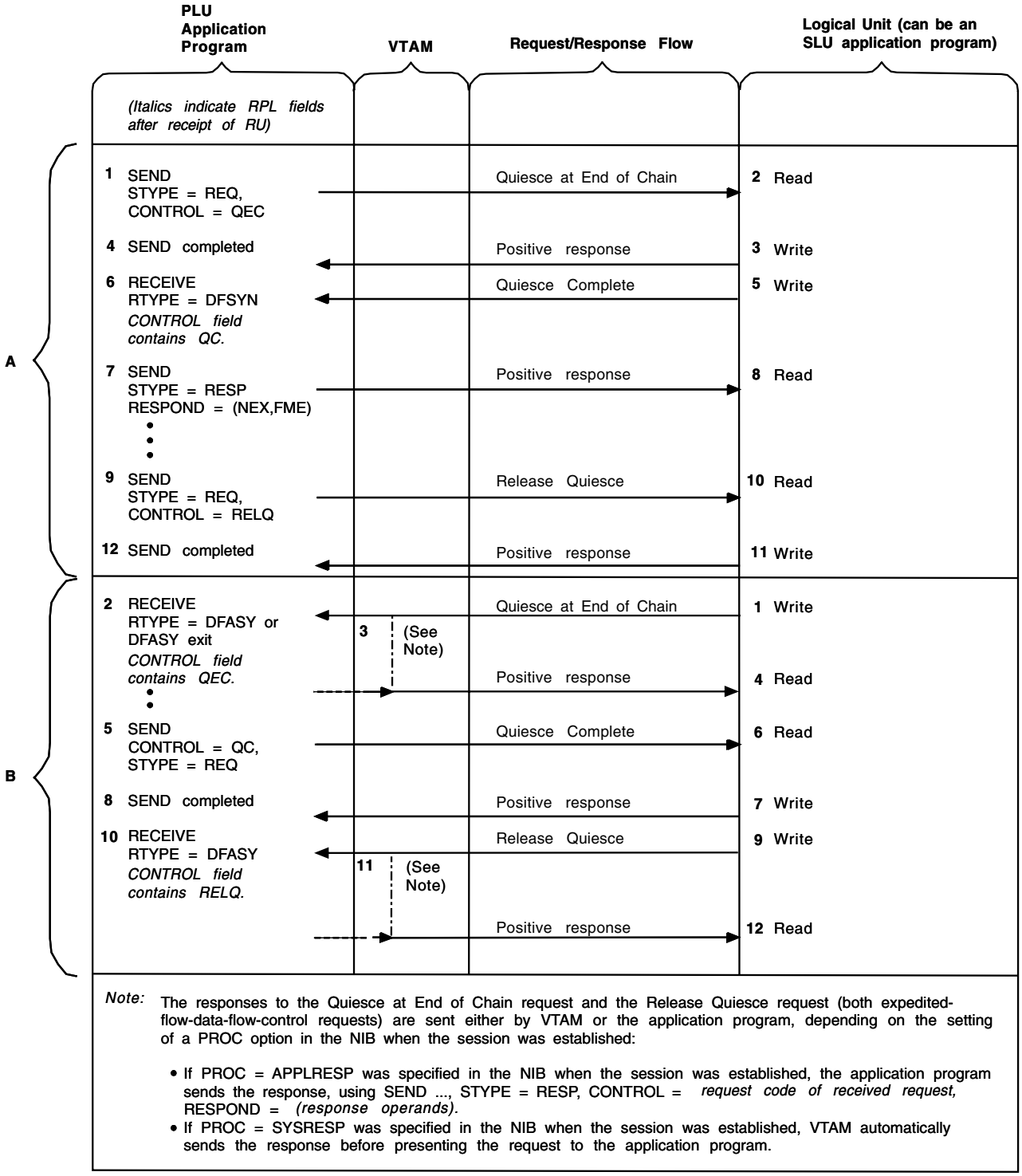


Figure 115. Application program and logical unit use quiesce protocol

A The application program quiesces the logical unit

B The logical unit quiesces the application program.

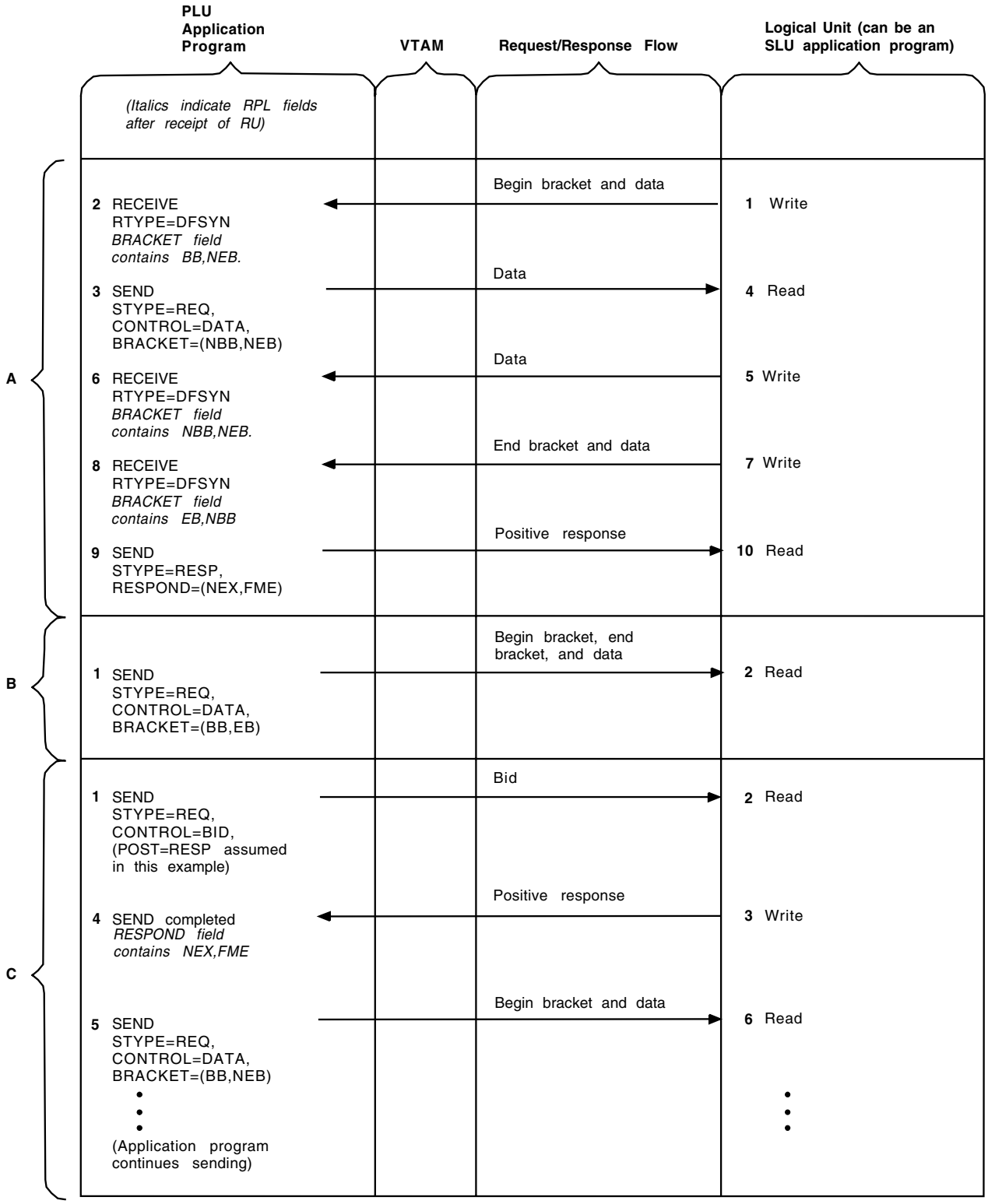


Figure 116. Application program and logical unit use bracket protocol (Part 1 of 2)

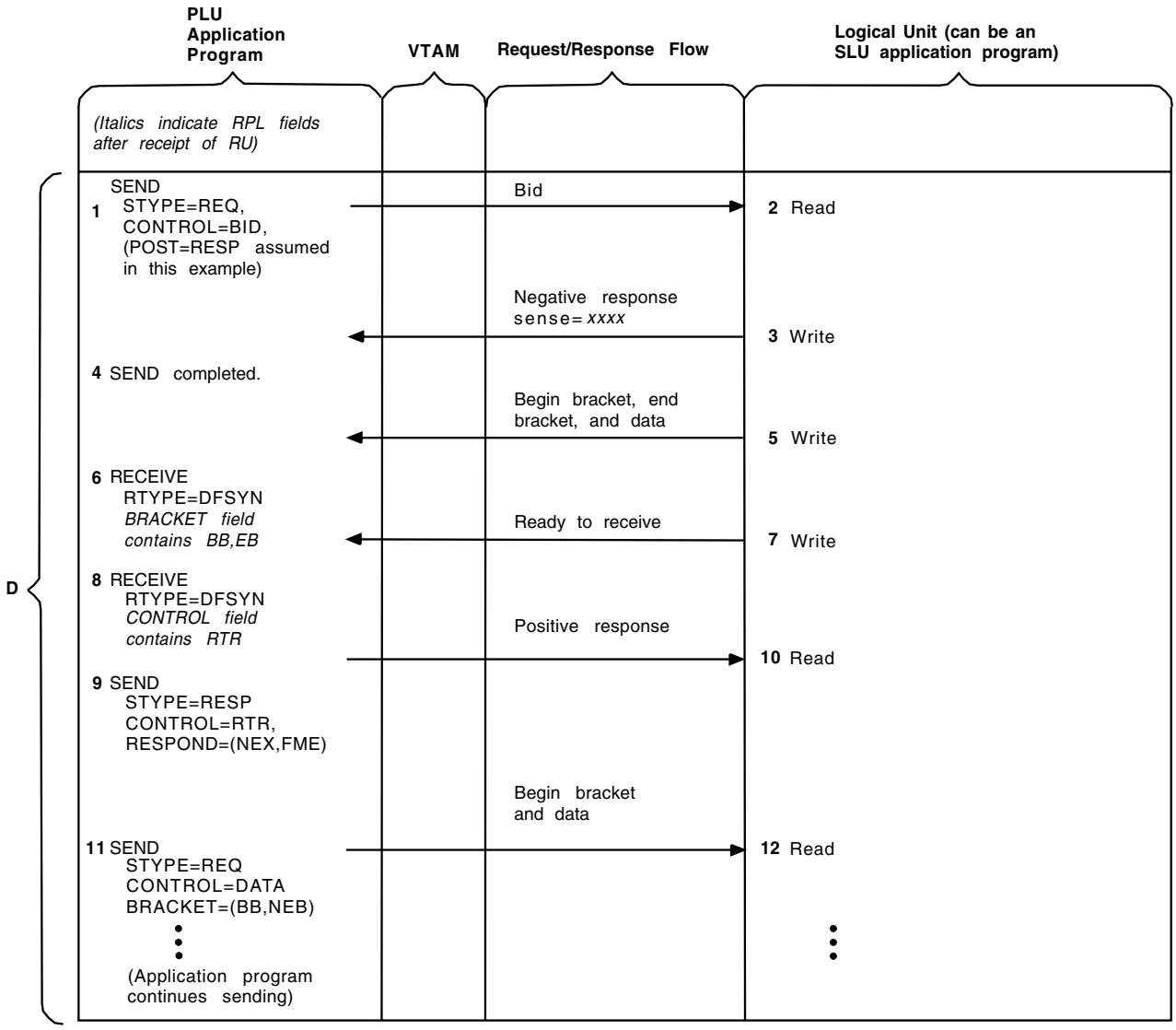


Figure 117. Application program and logical unit use bracket protocol (Part 2 of 2)

A Where the logical unit begins the bracket

B Where the PLU application program begins the bracket

C Where the PLU application program gets a positive response to its BID and begins the bracket

D Where BID produces a later Ready-to-Receive request.

For more details on using brackets with a SLU application program, see Figure 134 on page 721 and Figure 135 on page 722.

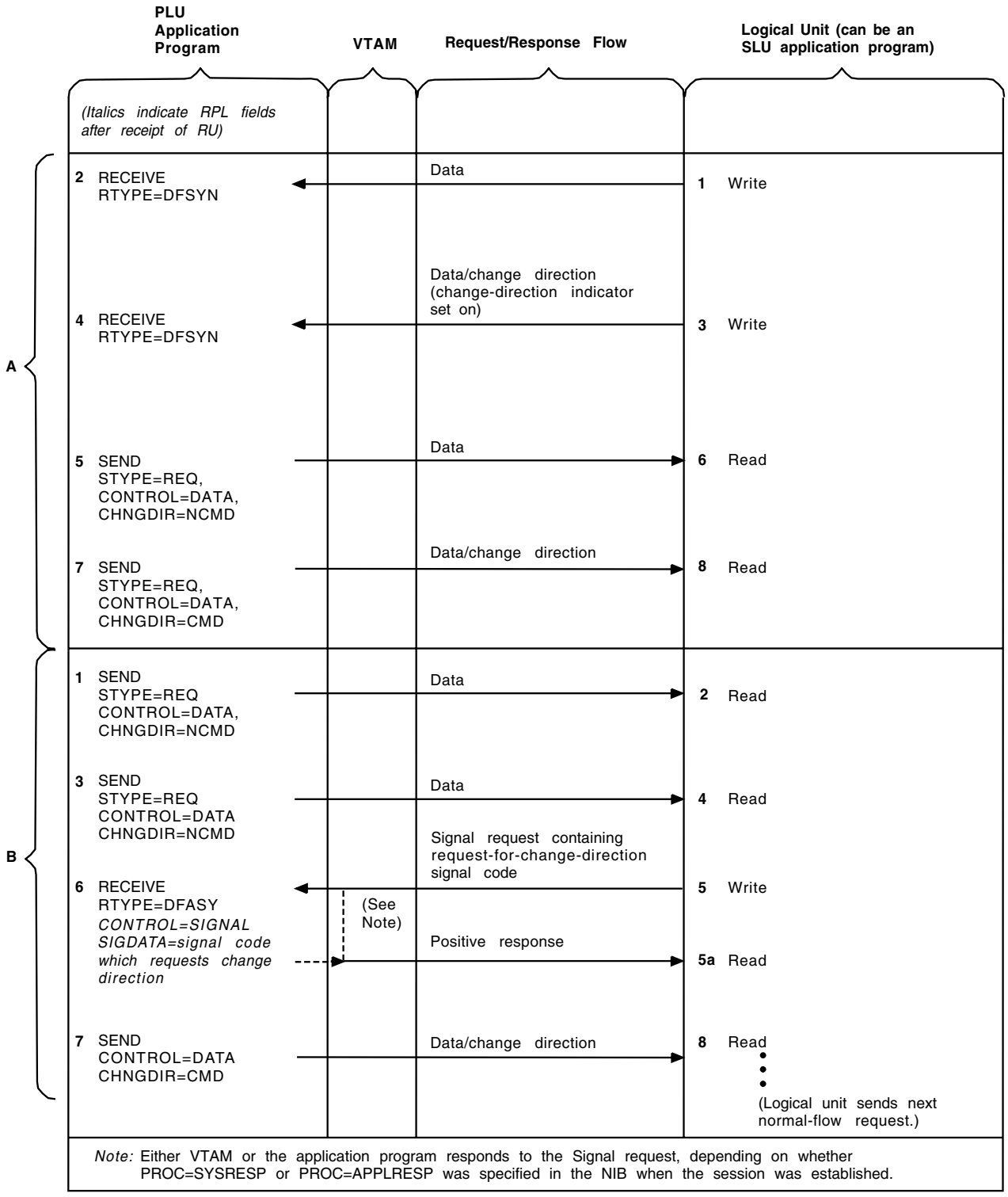


Figure 118. Application program and logical unit use change-direction protocol

A Where only change-direction indicators are used

B Where in addition, the Signal request (requesting change direction) is used.

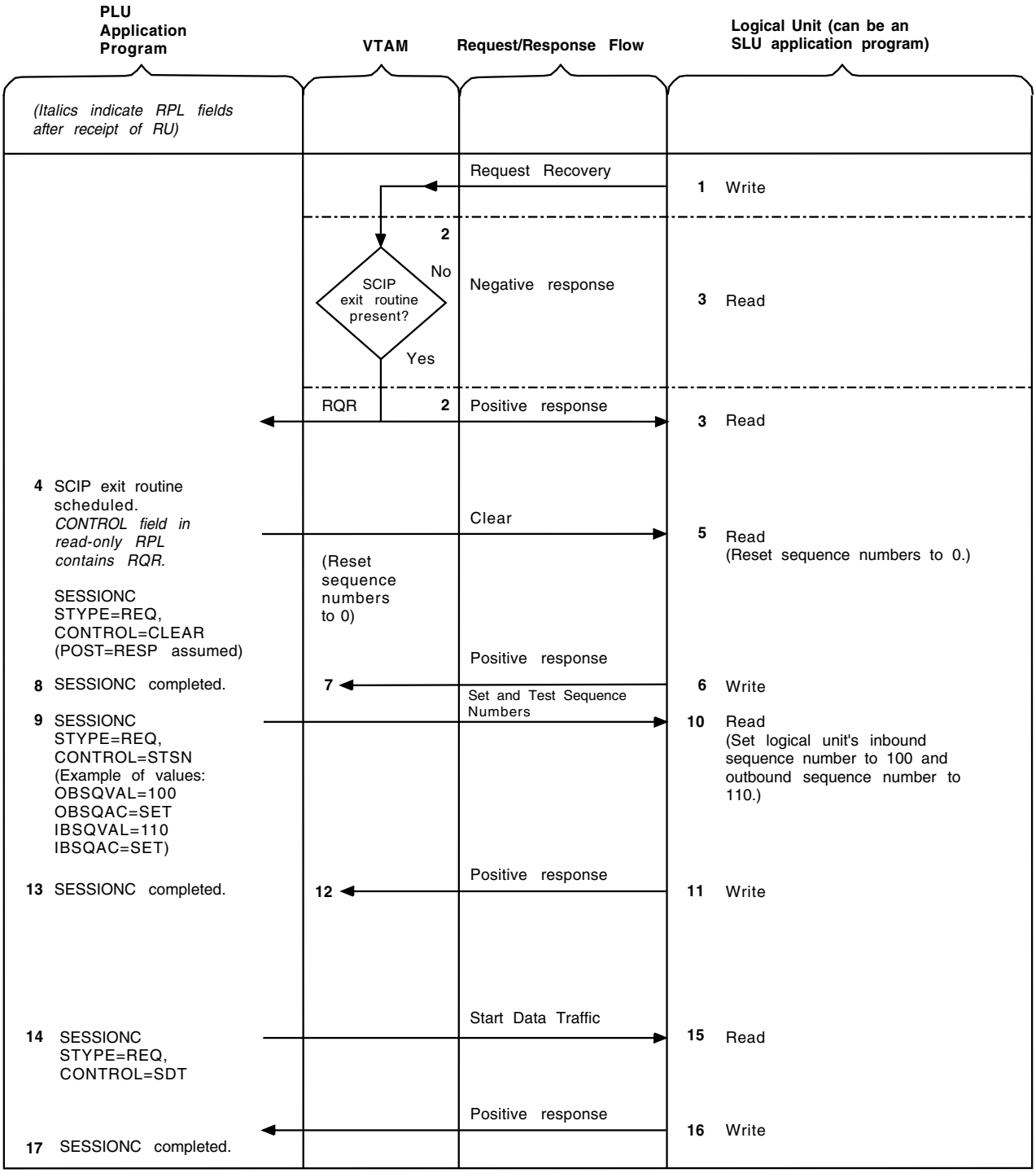


Figure 119. PLU application program resynchronizes sequence numbers with the logical unit.

For resynchronization of sequence numbers between PLU and SLU application programs, see Figure 132 on page 719 and Figure 133 on page 720. For the dialog between the PLU application program and the logical unit to establish sequence numbers, see steps 2–9 in Figure 109 on page 696 and Figure 110 on page 697. (Use STSN and any dialog at your own discretion.)

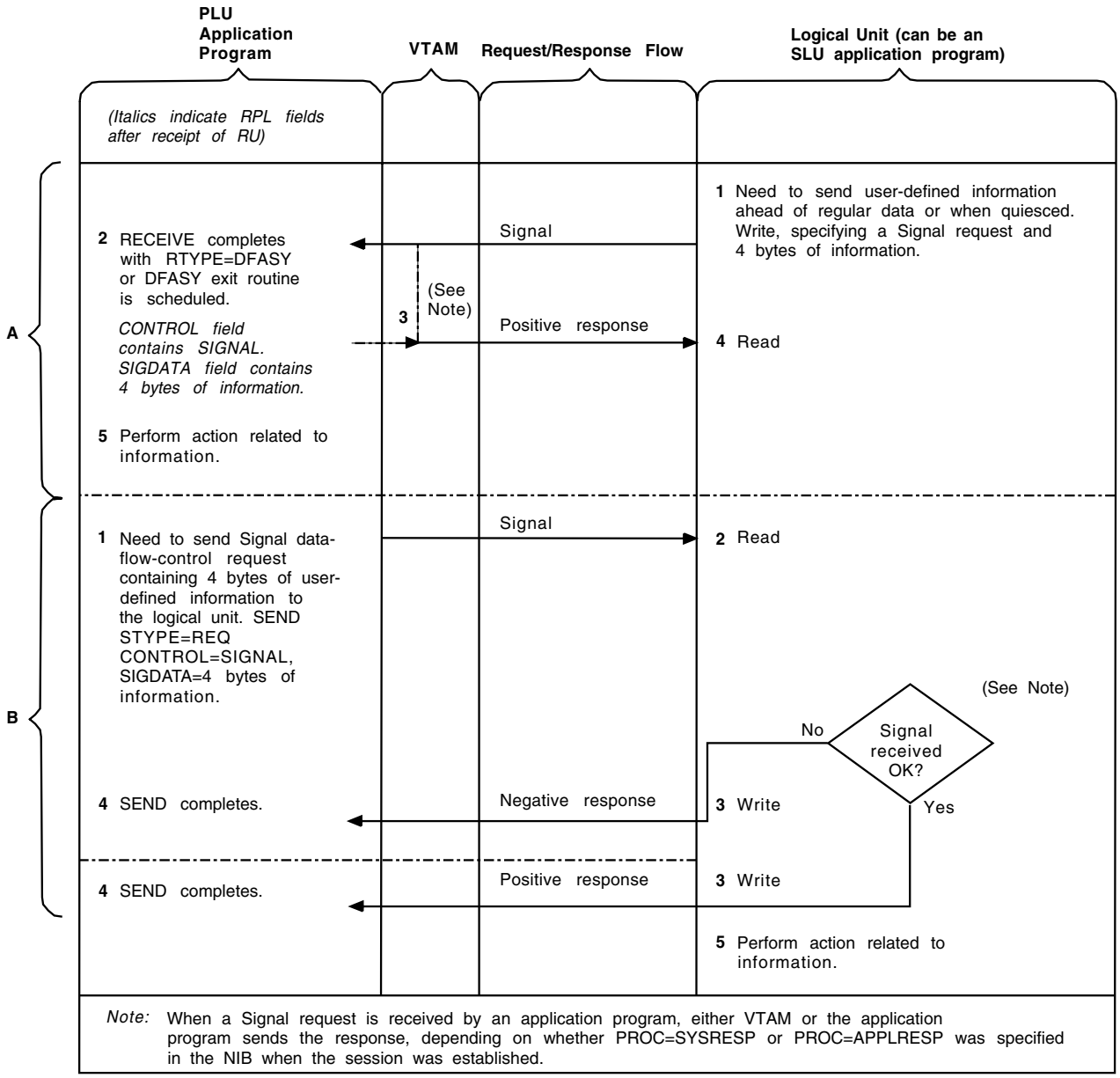


Figure 120. Application program and logical unit use the signal request

A Sent by the logical unit

B Sent by the PLU application program.

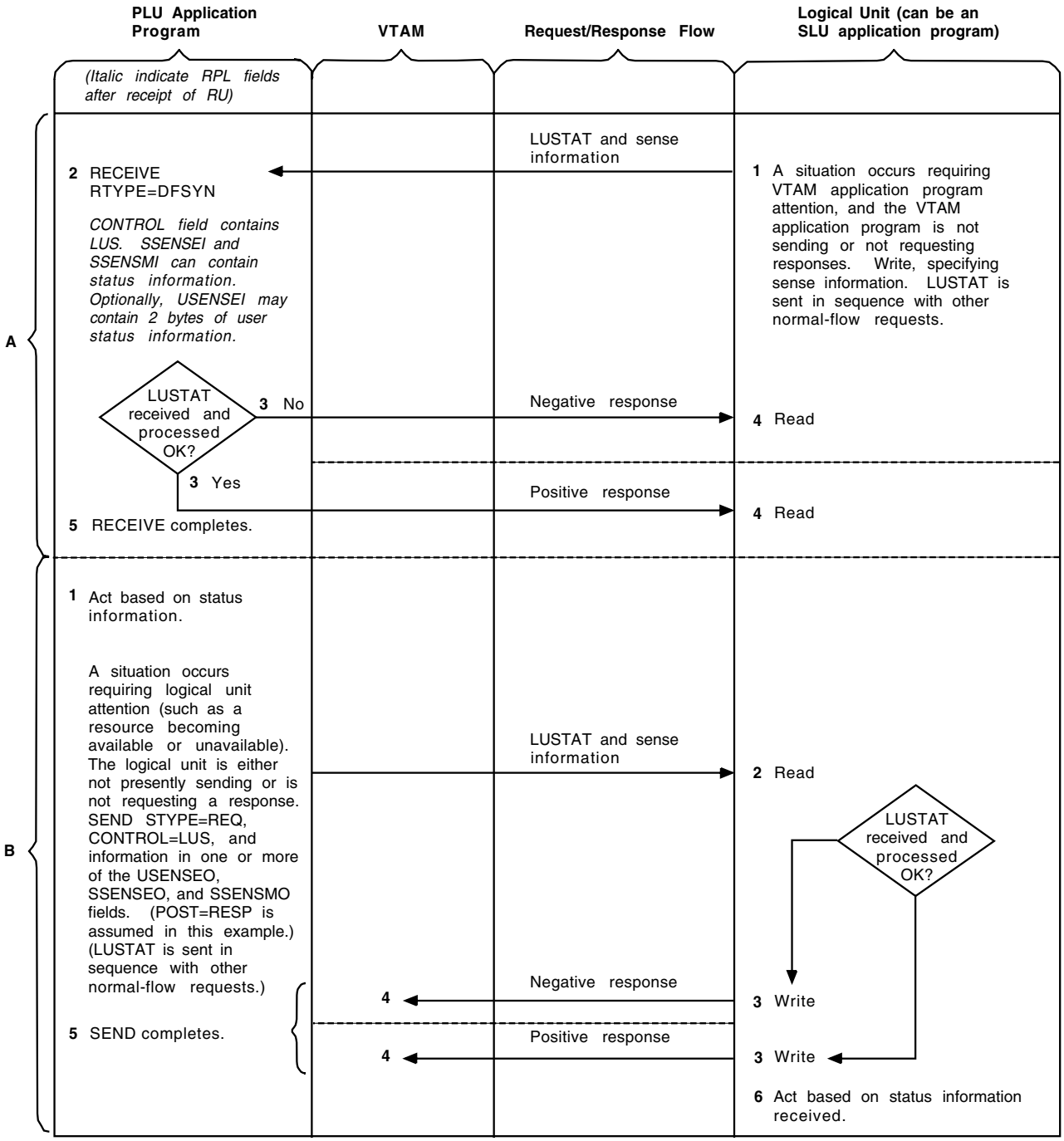


Figure 121. Application program and logical unit use the LUSTAT request

A Sent by the logical unit

B Sent by the application program.

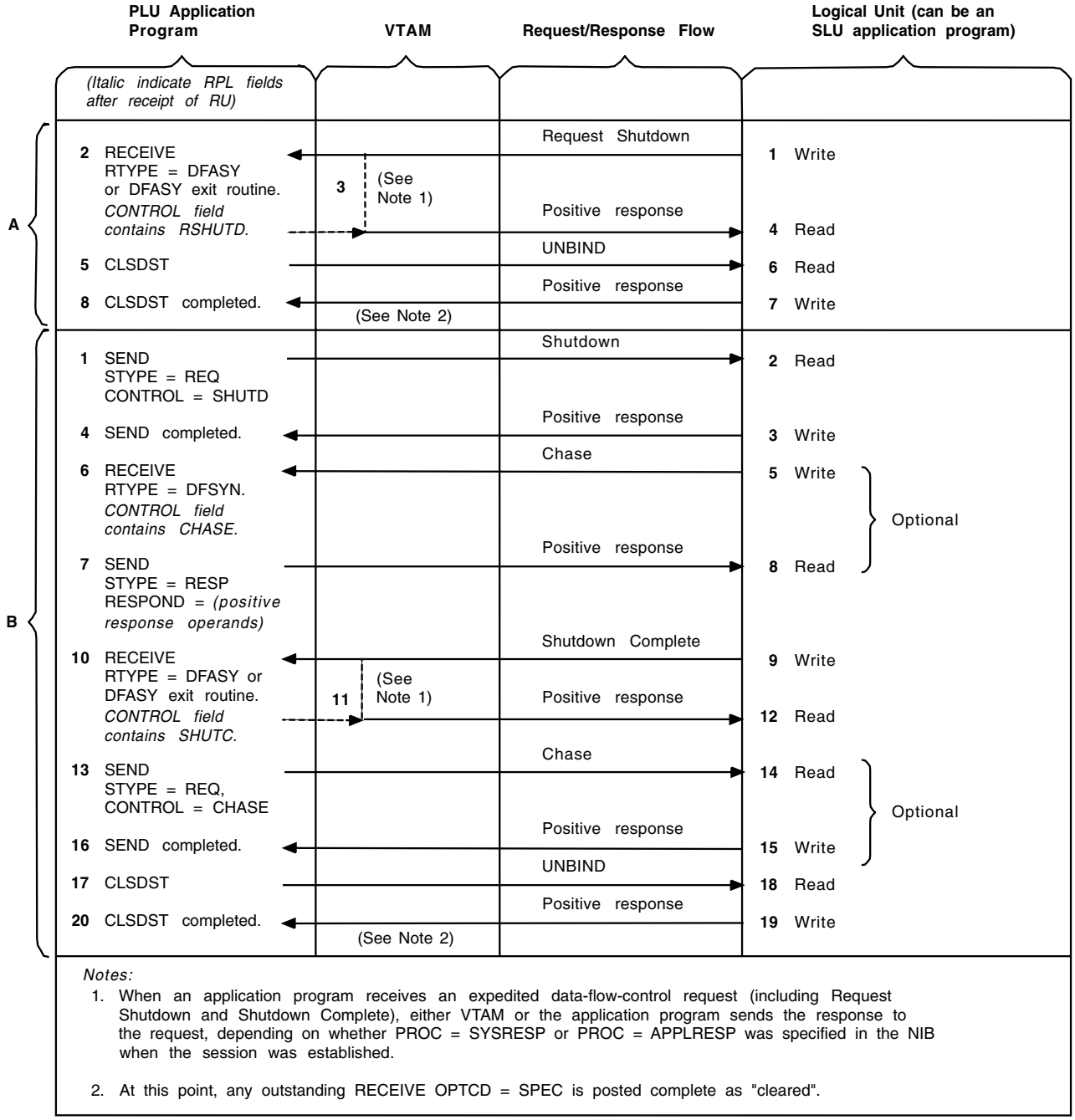


Figure 122. Operations are shut down in an orderly fashion

A The logical unit requests shutdown

B The PLU application program orders shutdown.

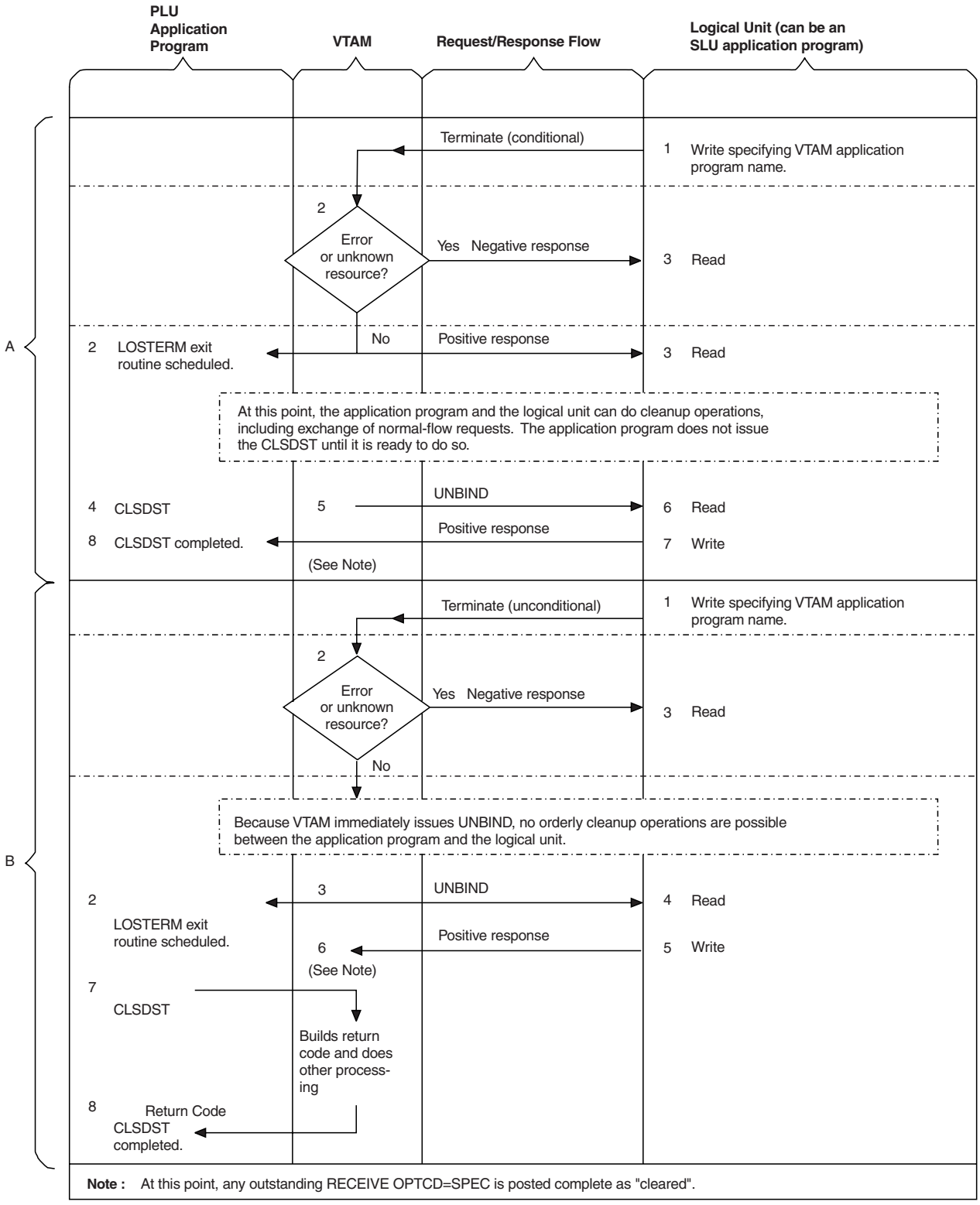


Figure 123. Logical unit terminates a session.

For termination requests from a SLU application program, see Figure 136 on page 723 and Figure 137 on page 724.

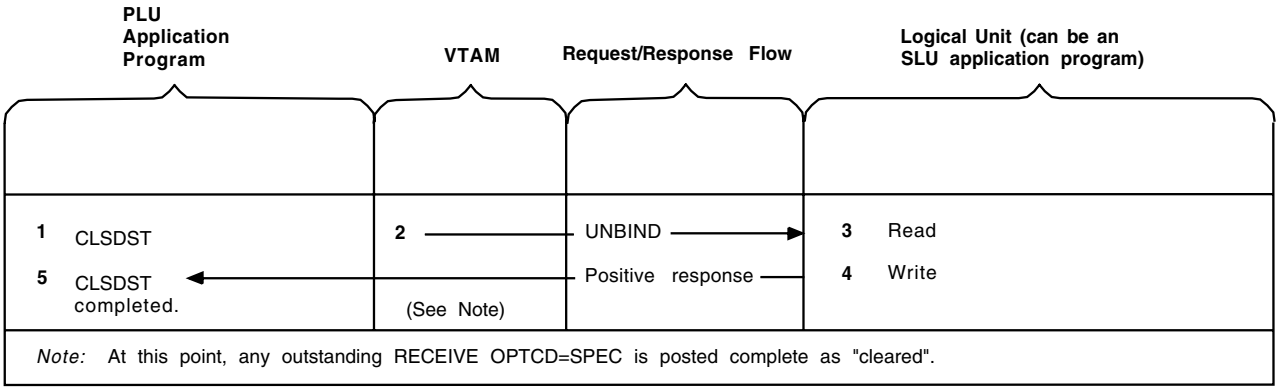


Figure 124. PLU application program terminates a session with the logical unit.

For session termination of a SLU application program, see Figure 136 on page 723, Figure 137 on page 724, Figure 138 on page 725, and Figure 139 on page 726.

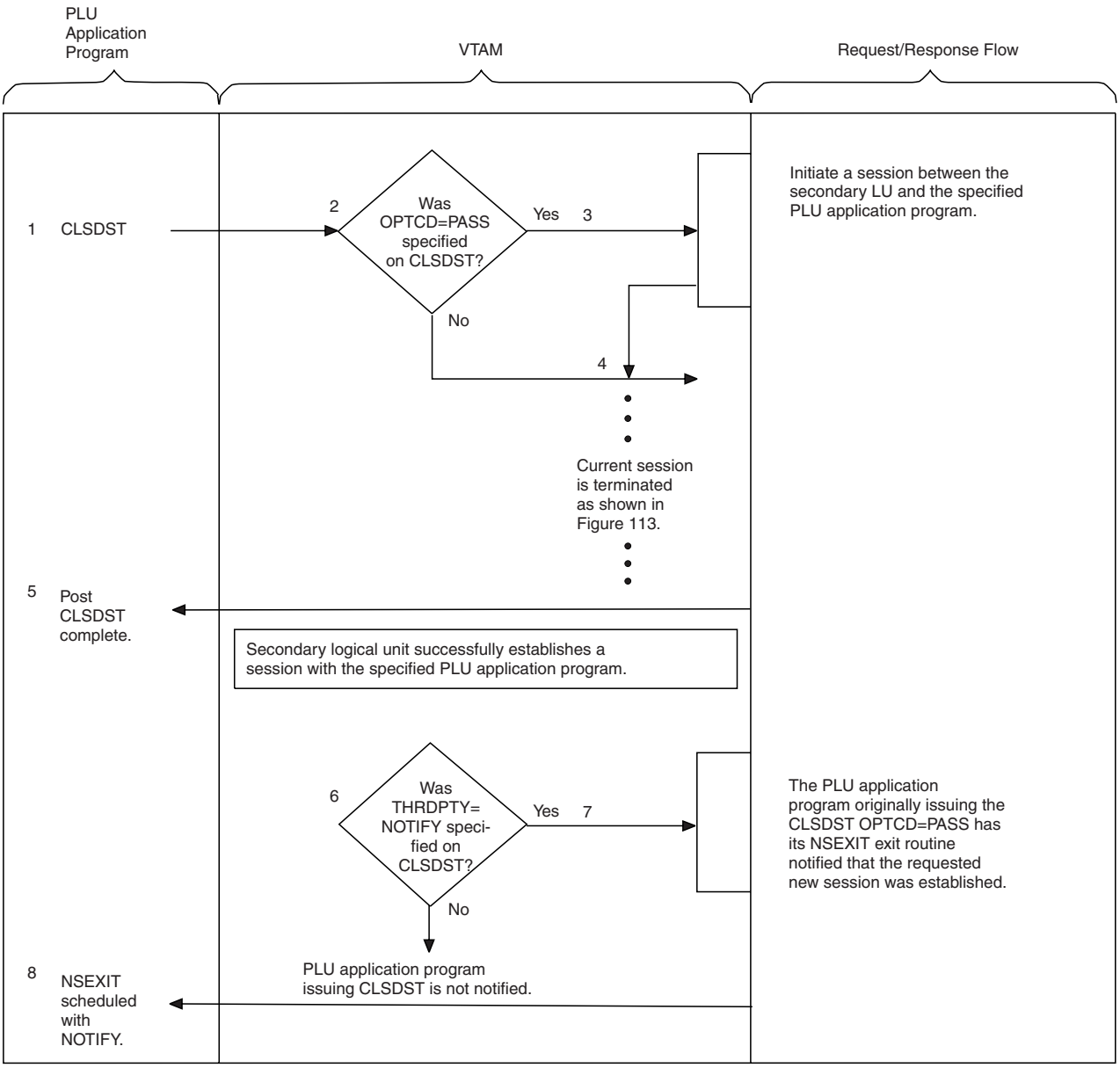


Figure 125. PLU application program terminates a session with the logical unit with a CLSDST OPTCD=PASS

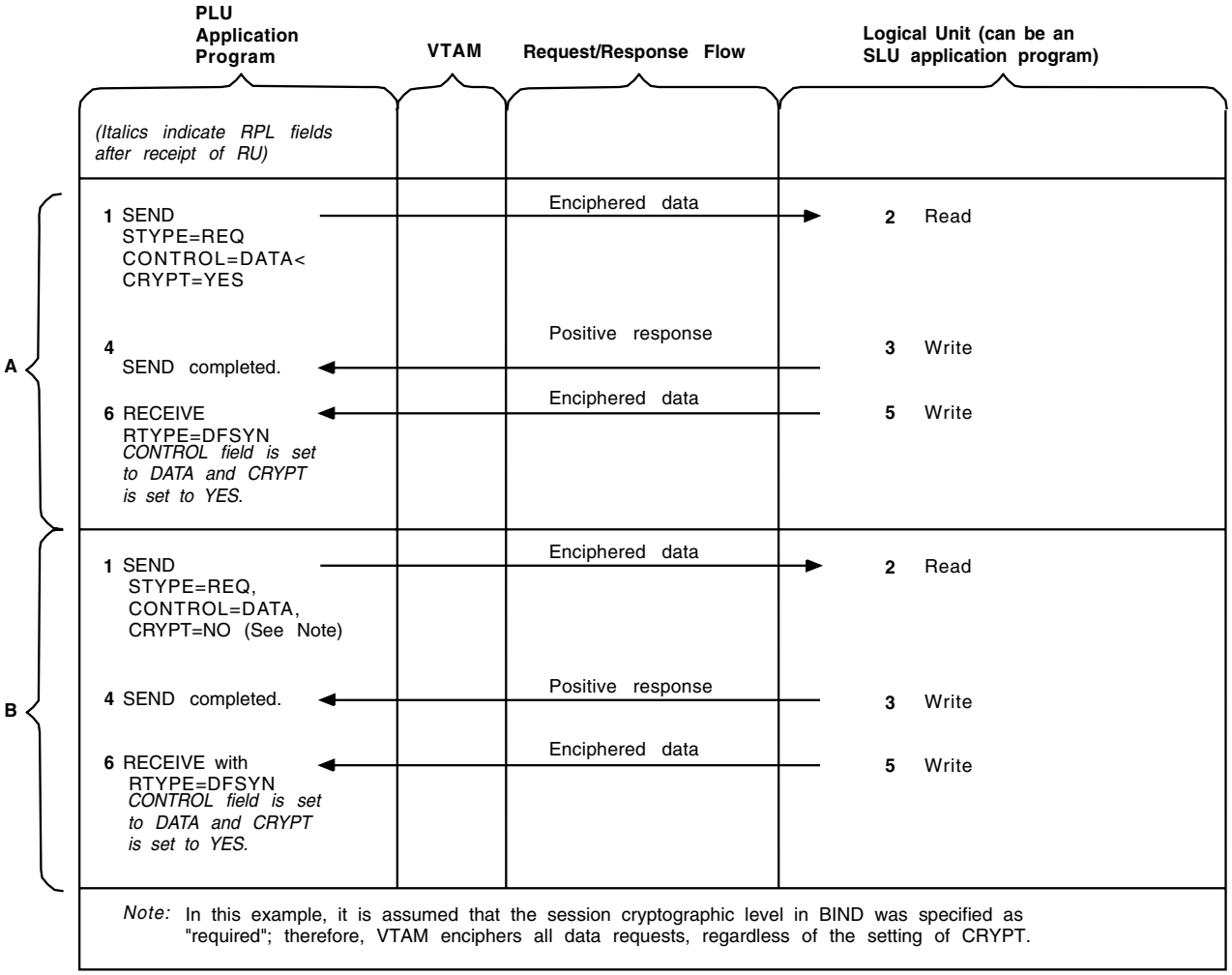


Figure 126. PLU application program and a device-type logical unit use cryptography in a required cryptographic session

A Where cryptography is specified on the SEND

B Where no cryptography is specified on the SEND.

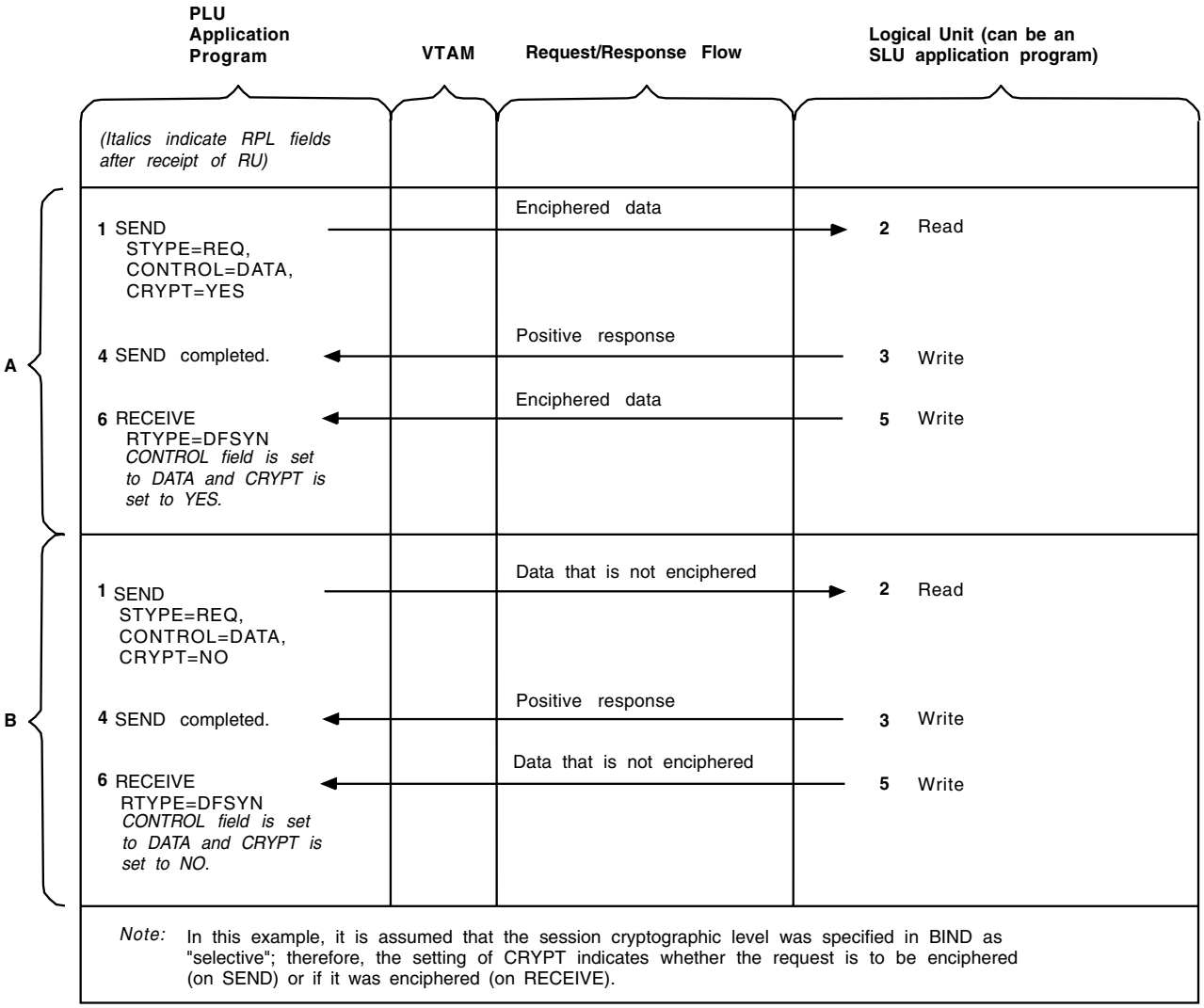


Figure 127. PLU application program and a device-type logical unit use cryptography in a selective cryptographic session

A Where cryptography is specified on the RECEIVE

B Where no cryptography is specified on the RECEIVE.

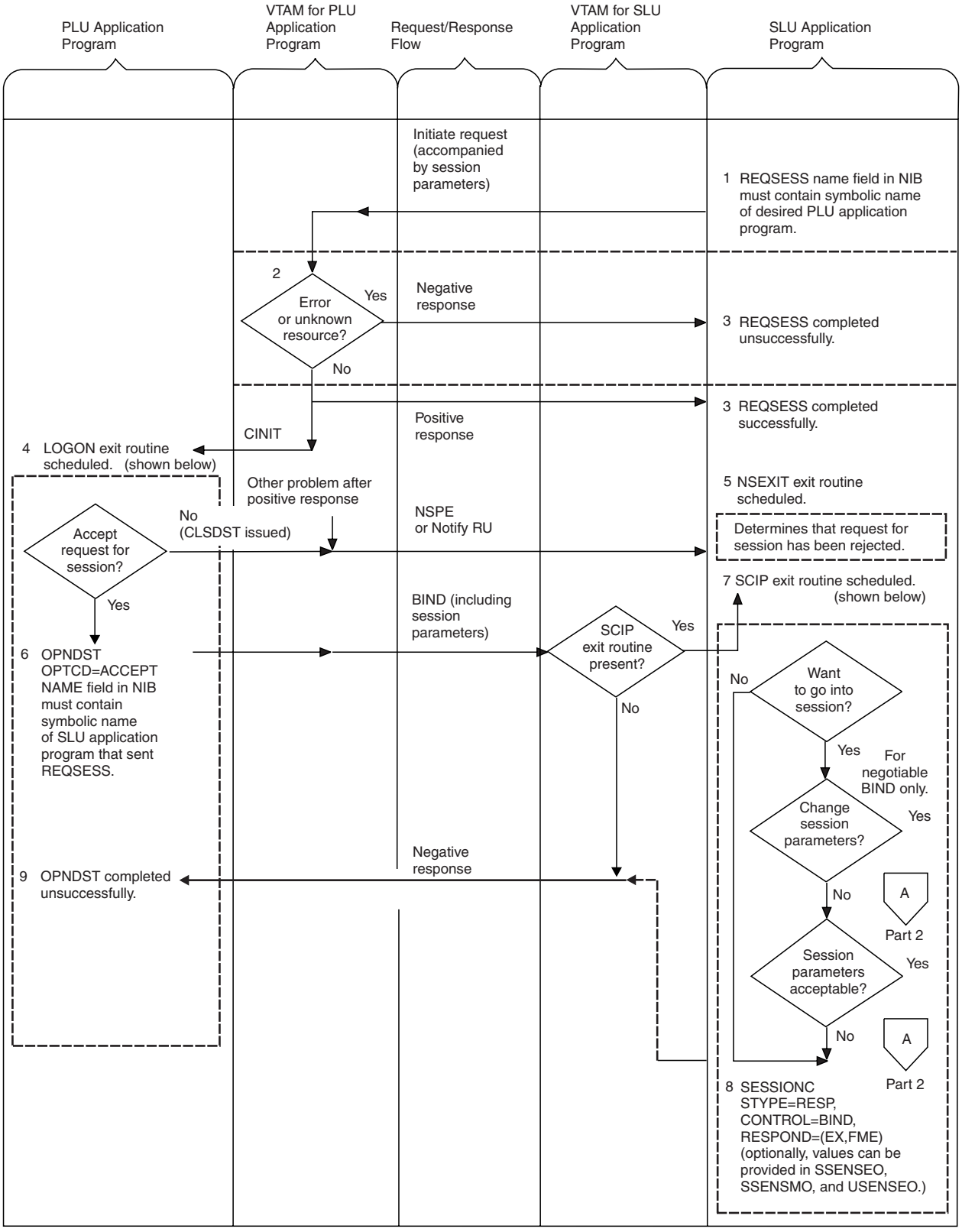


Figure 128. SLU application program requests a session with the PLU application program (Part 1 of 2)

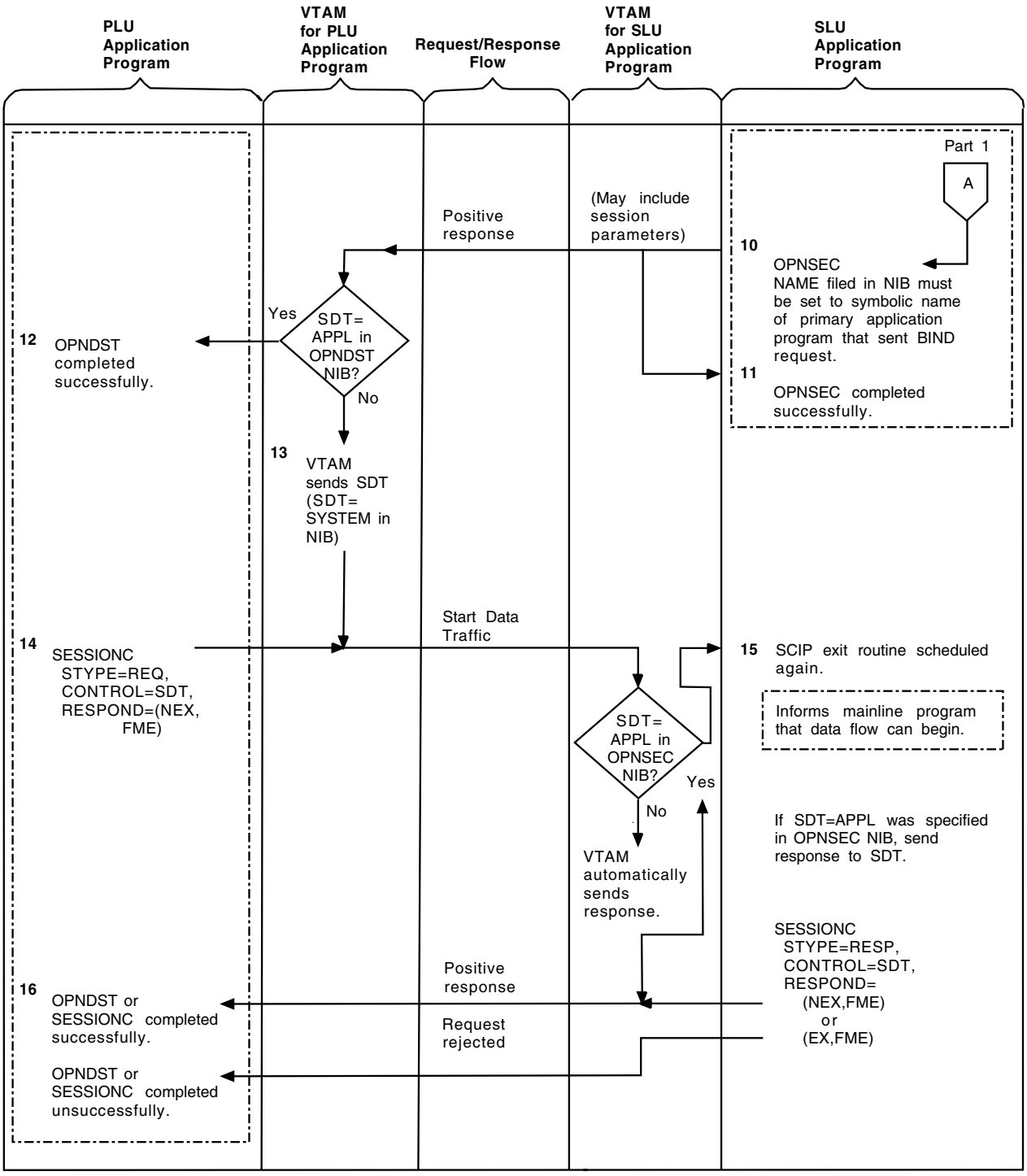


Figure 129. SLU application program requests a session with the PLU application program (Part 2 of 2)

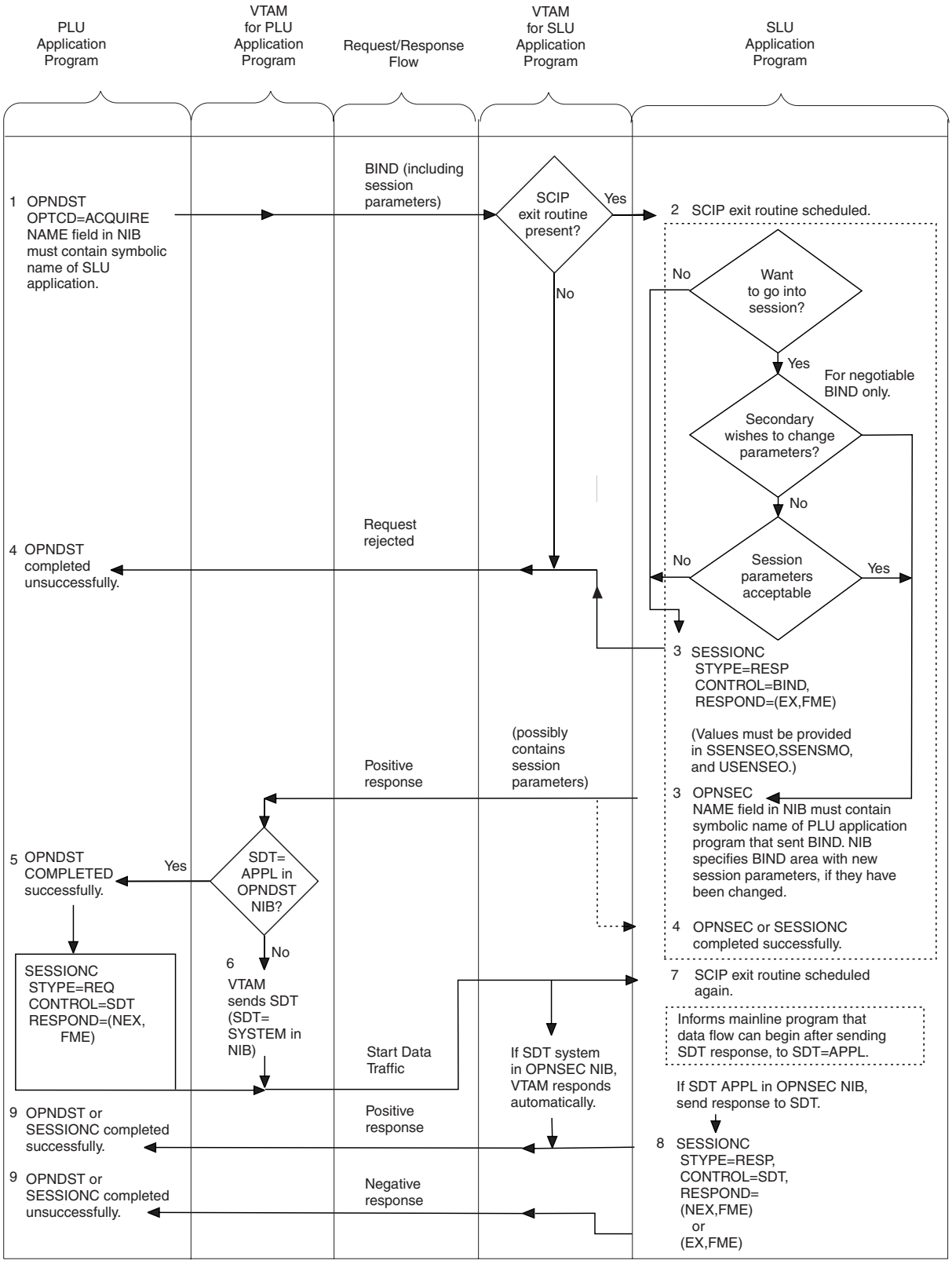


Figure 130. PLU application program acquires (initiates and establishes) a session with the SLU application program

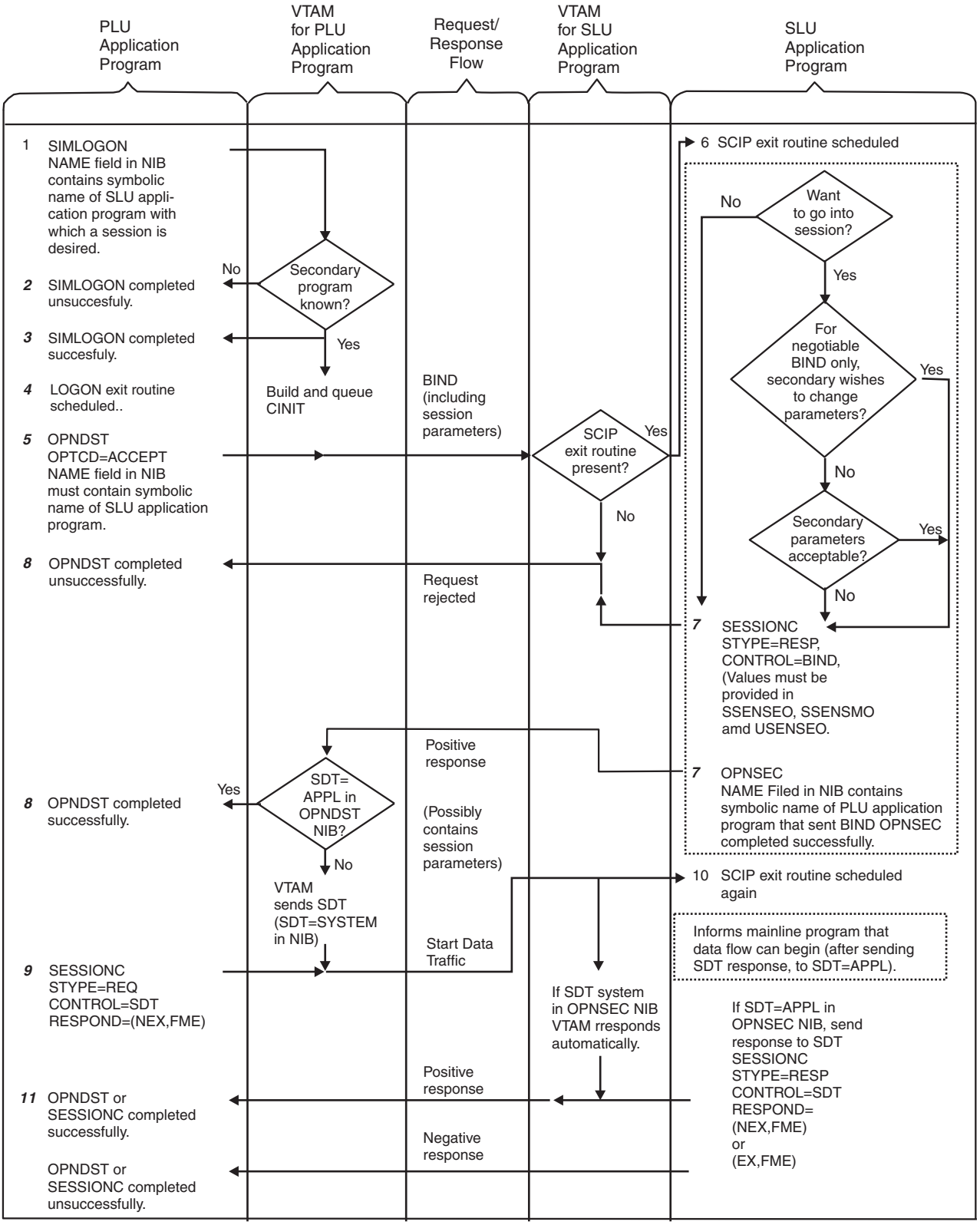


Figure 131. PLU application program issues a SIMLOGON to acquire (initiate a session with) the SLU application program

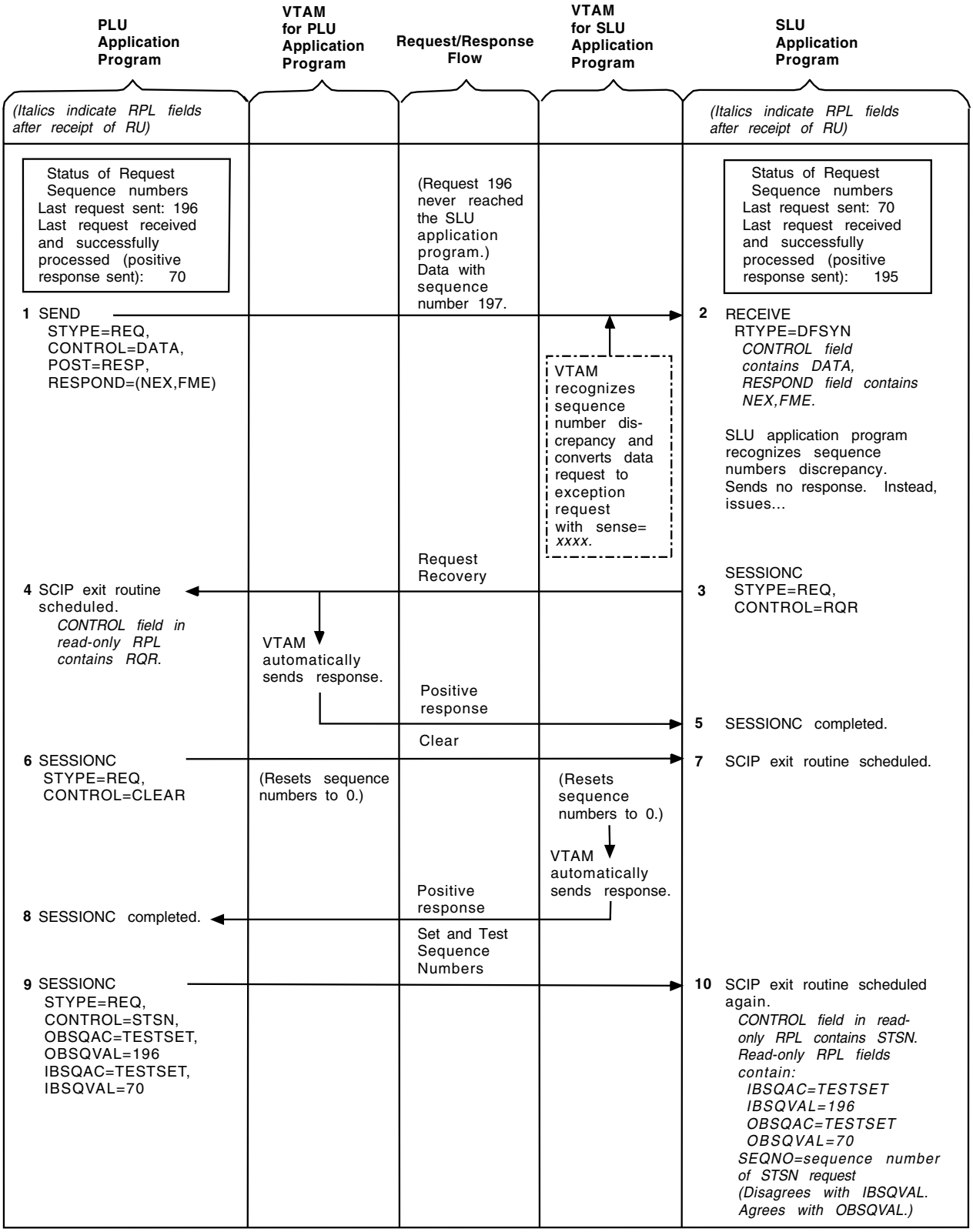


Figure 132. PLU application program resynchronizes sequence numbers with the SLU application program (Part 1 of 2)

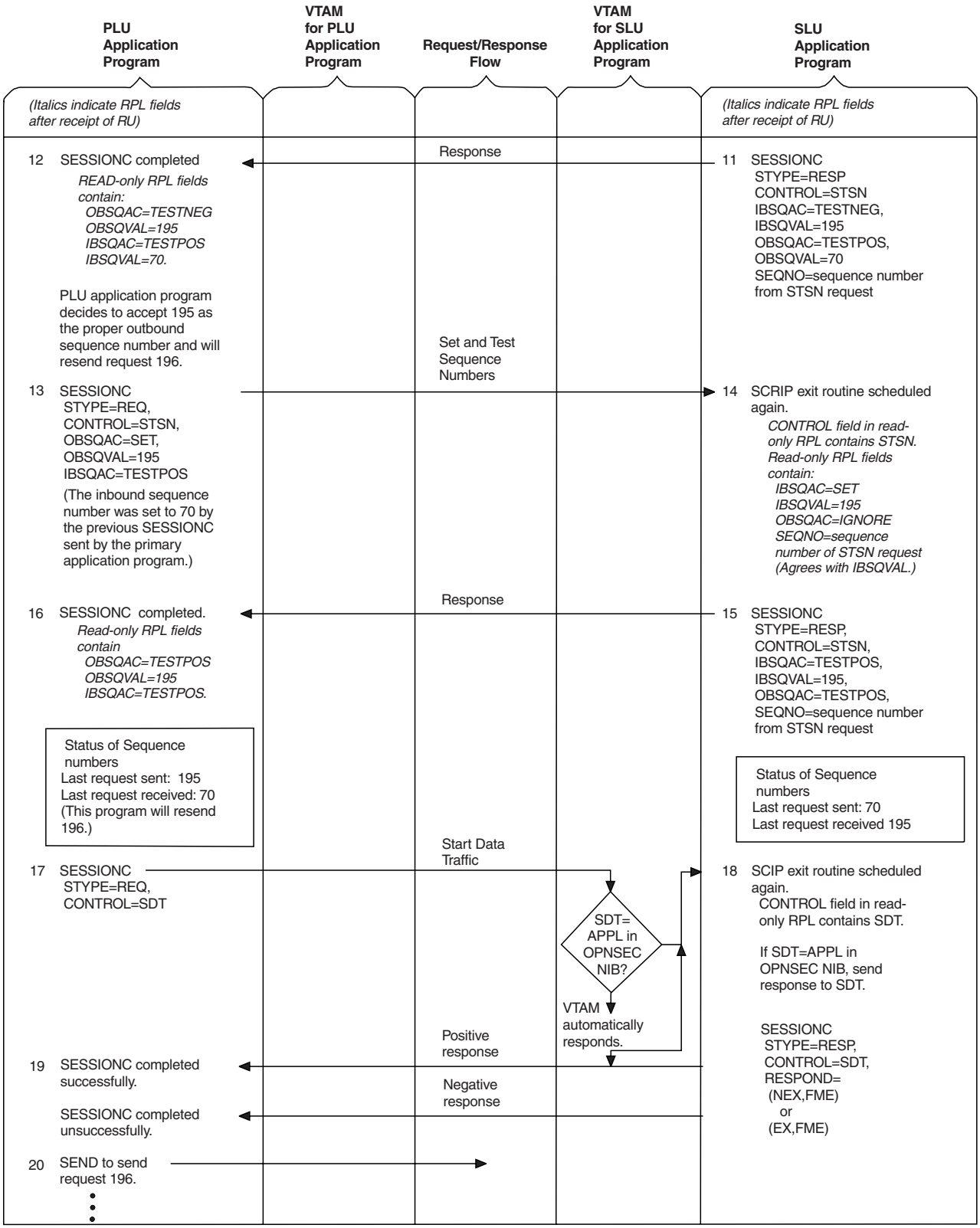


Figure 133. PLU application program resynchronizes sequence numbers with the SLU application program (Part 2 of 2)

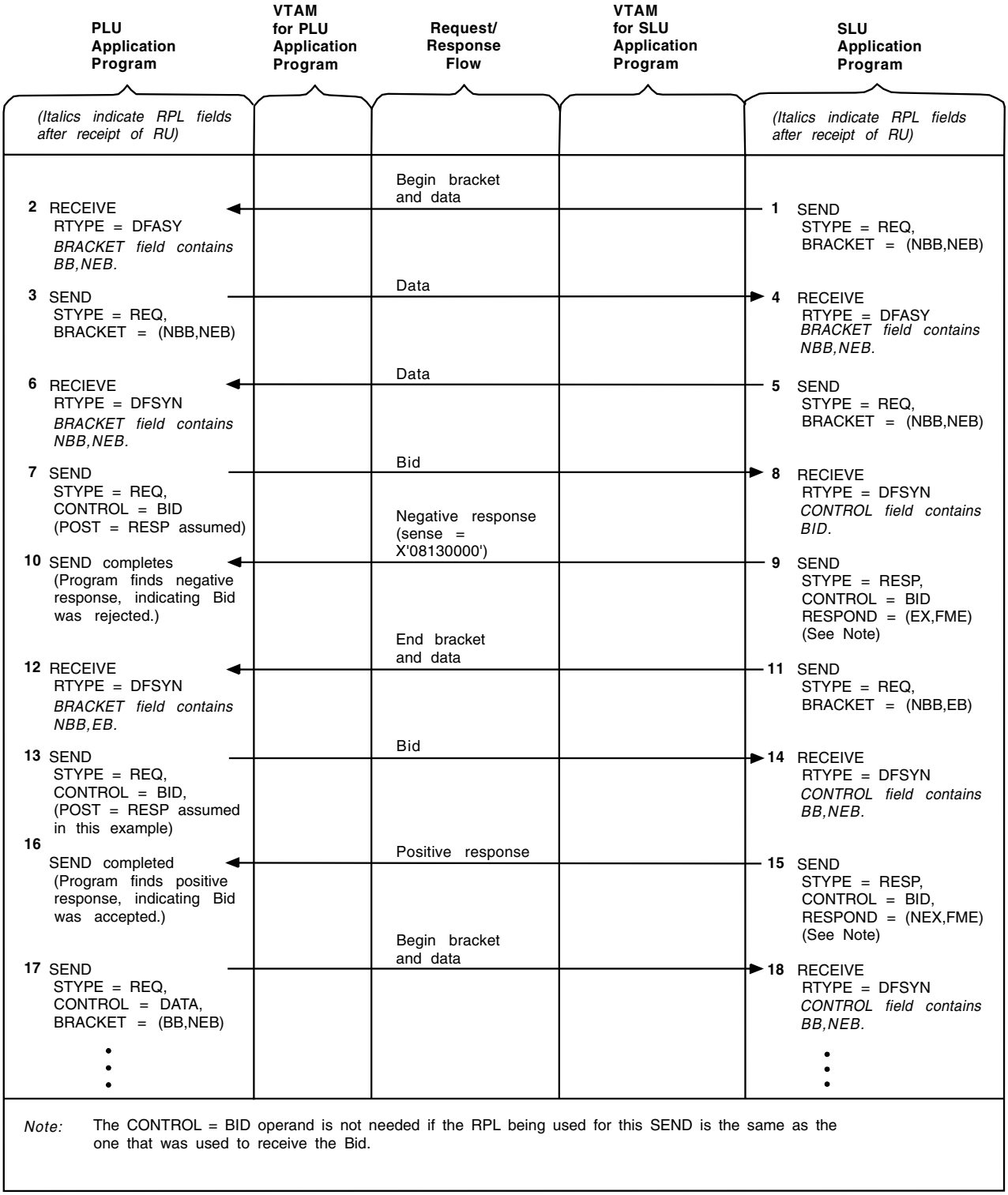


Figure 134. PLU application program and SLU application program use bracket protocol

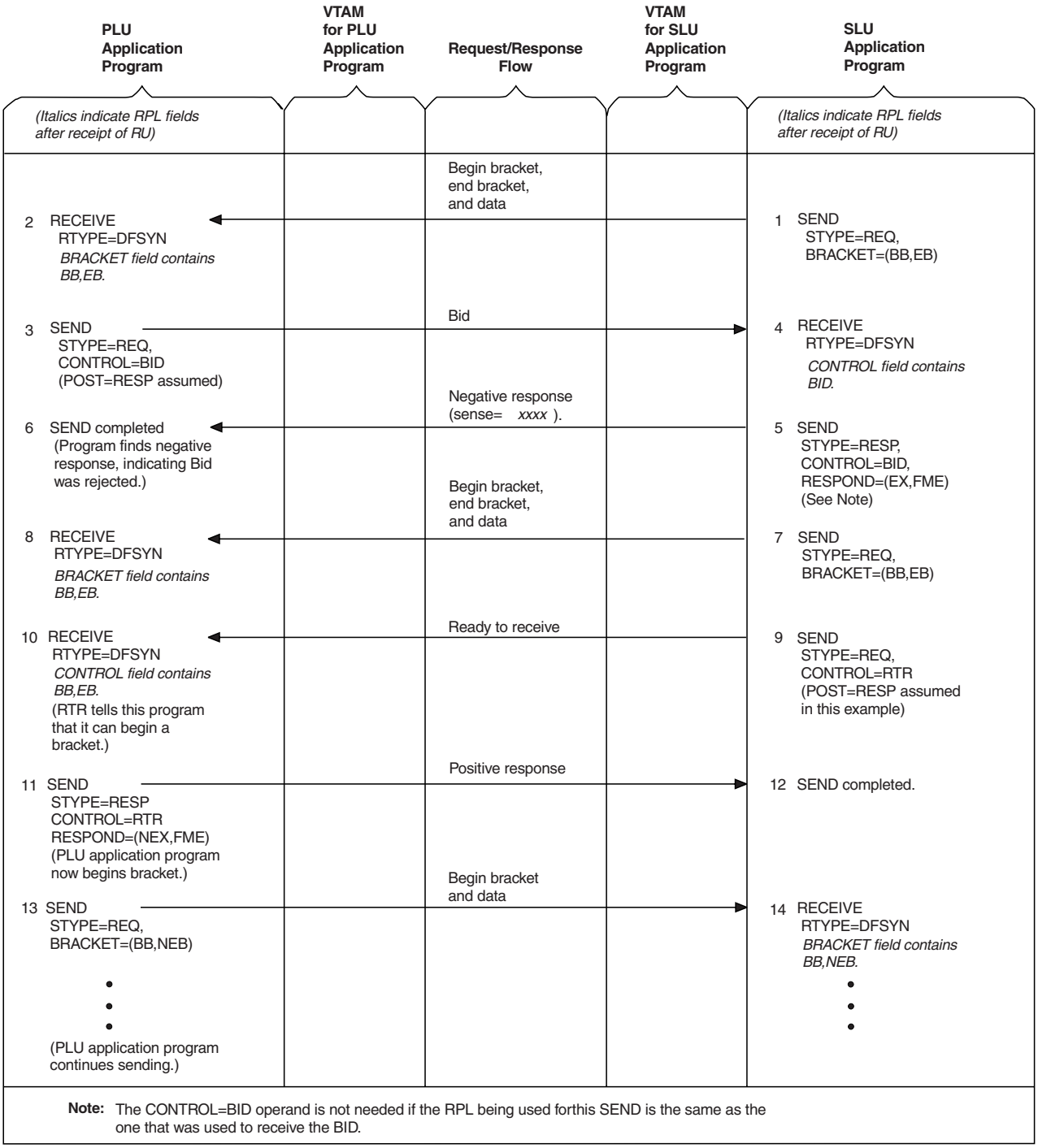


Figure 135. PLU application program and SLU use-bracket protocol: BID by PLU rejected but Ready-to-Receive follows

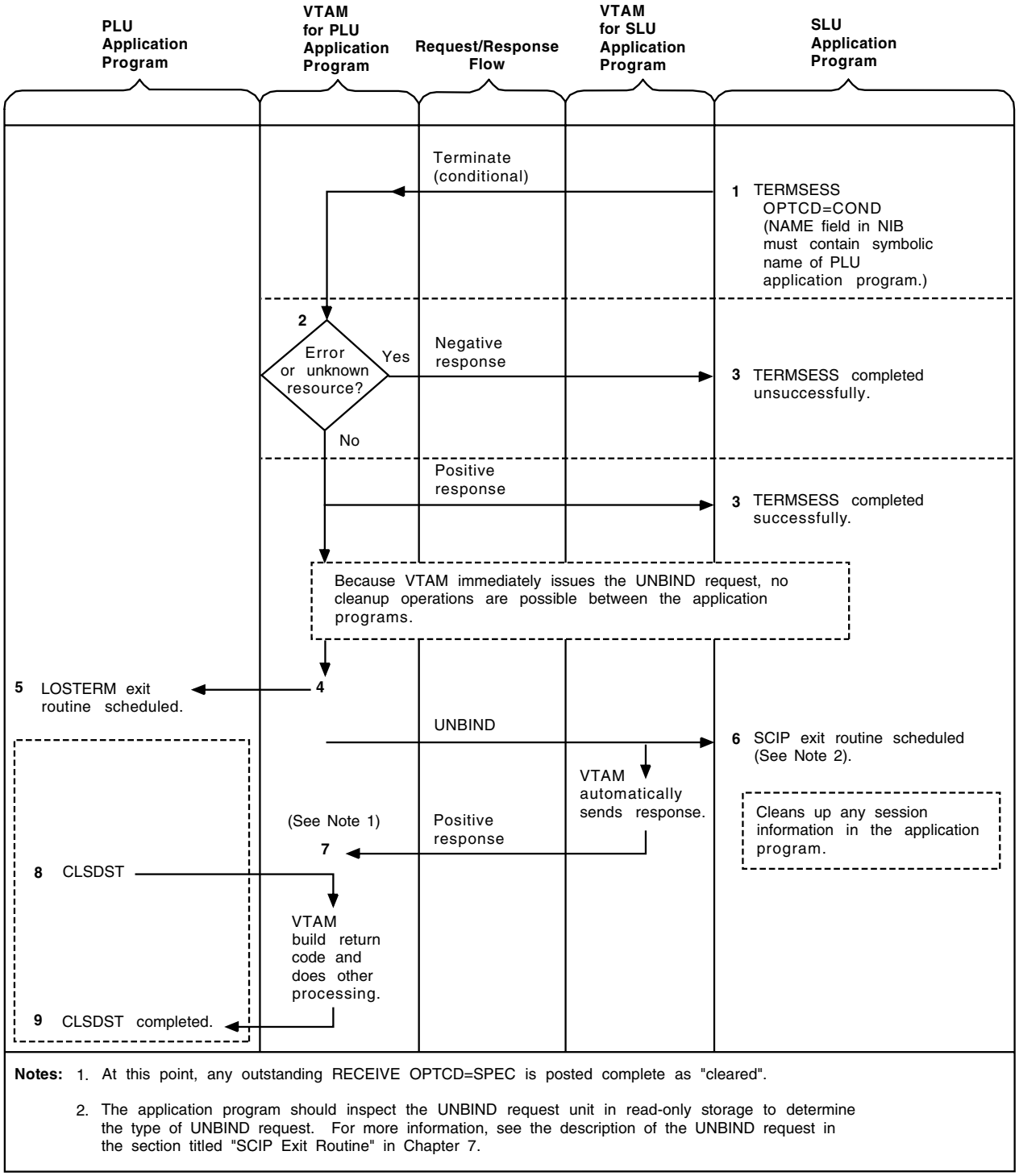


Figure 136. SLU application program sends a conditional request for session termination

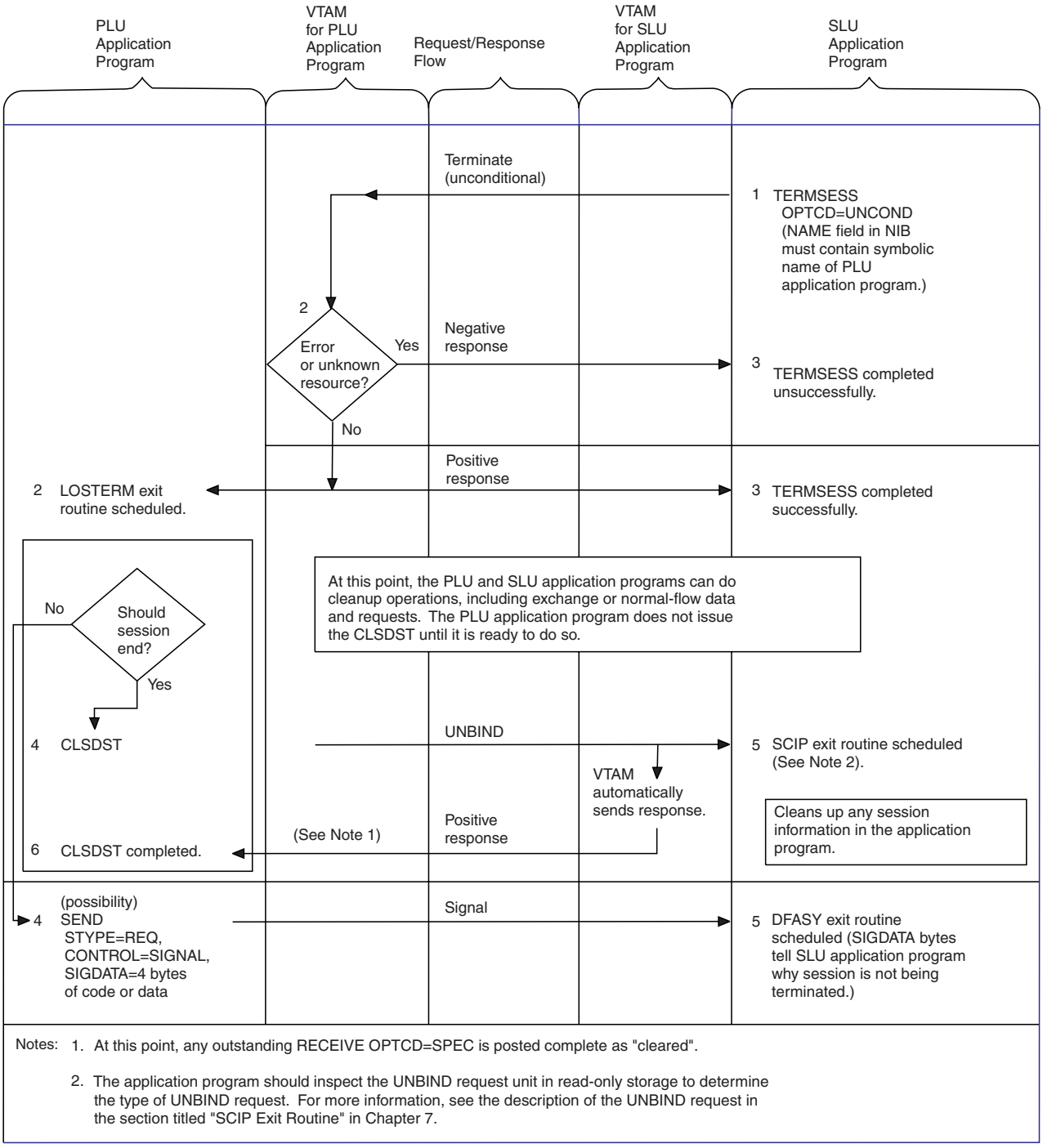
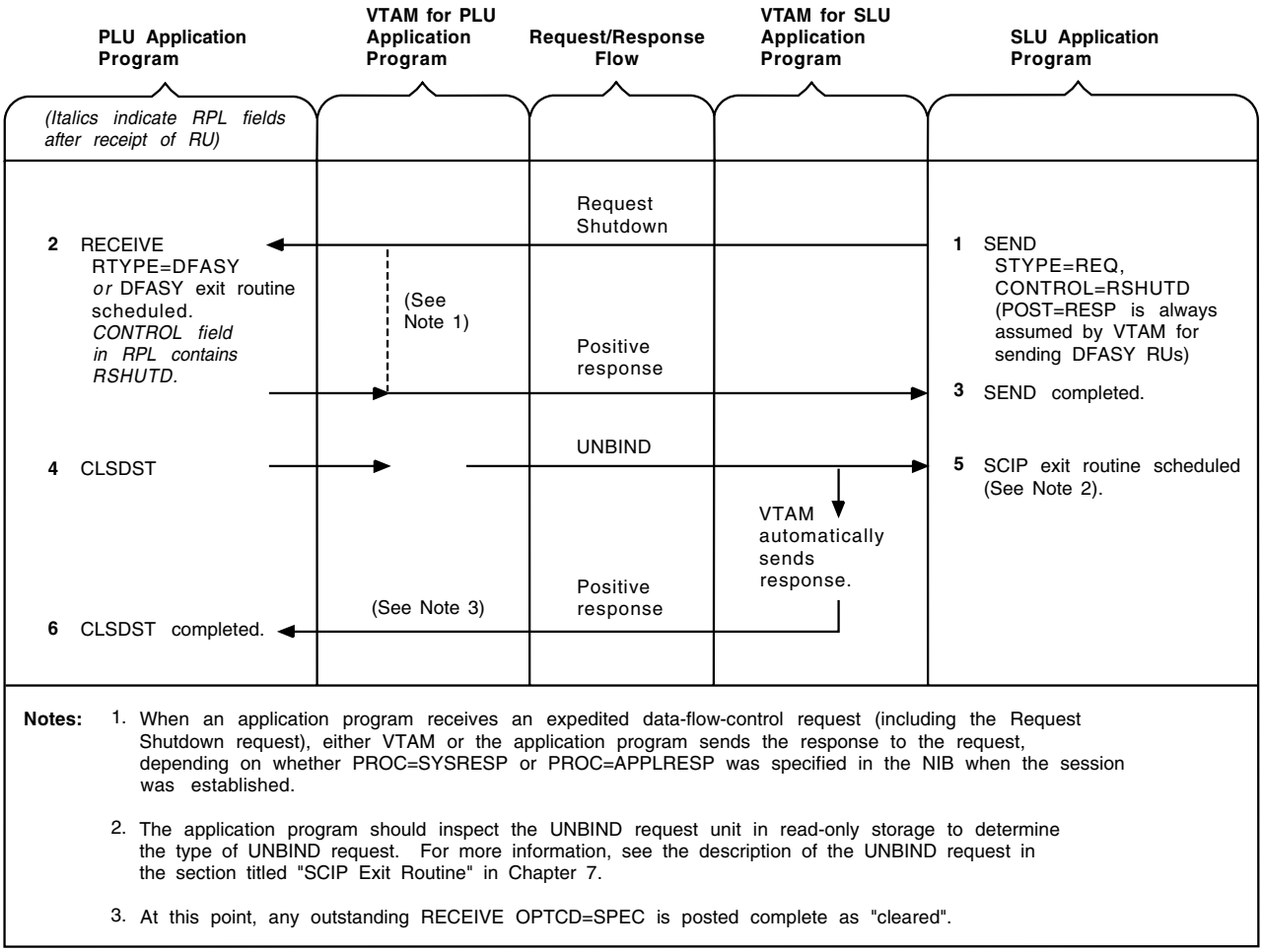


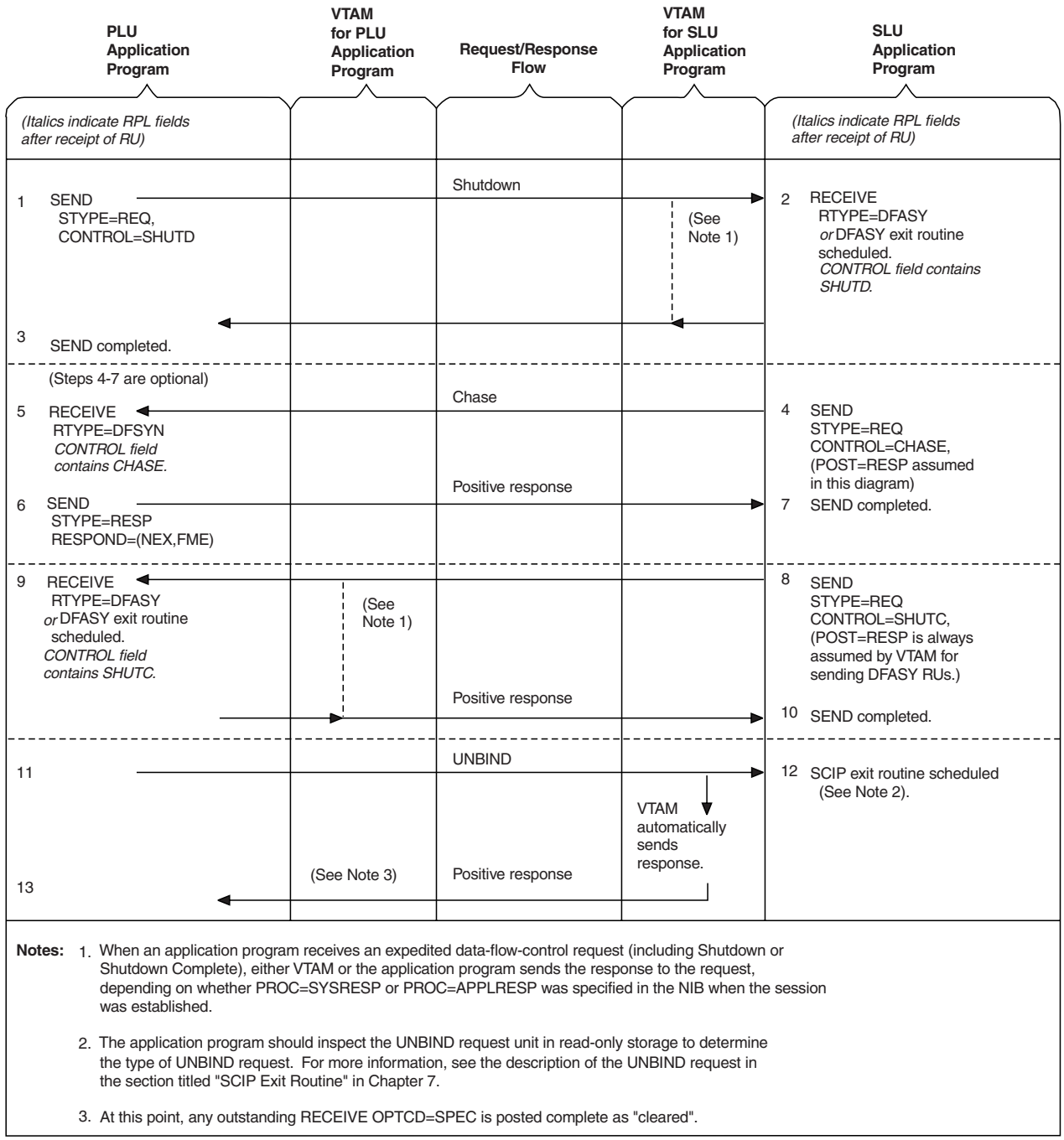
Figure 137. SLU application program sends an unconditional request for session termination



Notes:

1. When an application program receives an expedited data-flow-control request (including the Request Shutdown request), either VTAM or the application program sends the response to the request, depending on whether PROC=SYSRESP or PROC=APPLRESP was specified in the NIB when the session was established.
2. The application program should inspect the UNBIND request unit in read-only storage to determine the type of UNBIND request. For more information, see the description of the UNBIND request in the section titled "SCIP Exit Routine" in Chapter 7.
3. At this point, any outstanding RECEIVE OPTCD=SPEC is posted complete as "cleared".

Figure 138. SLU application program sends a Request Shutdown request



Notes:

1. When an application program receives an expedited data-flow-control request (including Shutdown or Shutdown Complete), either VTAM or the application program sends the response to the request, depending on whether PROC=SYSRESP or PROC=APPLRESP was specified in the NIB when the session was established.
2. The application program should inspect the UNBIND request unit in read-only storage to determine the type of UNBIND request. For more information, see the description of the UNBIND request in the section titled "SCIP Exit Routine" in Chapter 7.
3. At this point, any outstanding RECEIVE OPTCD=SPEC is posted complete as "cleared".

Figure 139. PLU application program shuts down the SLU application program

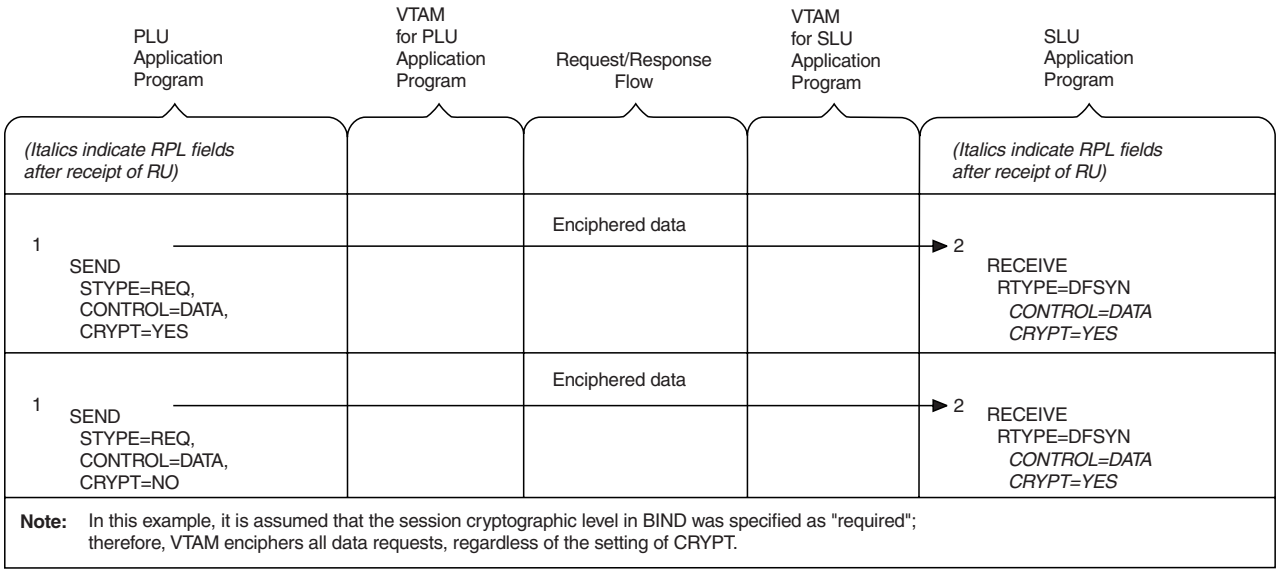


Figure 140. PLU application program and the SLU application program use cryptography in a required cryptographic session

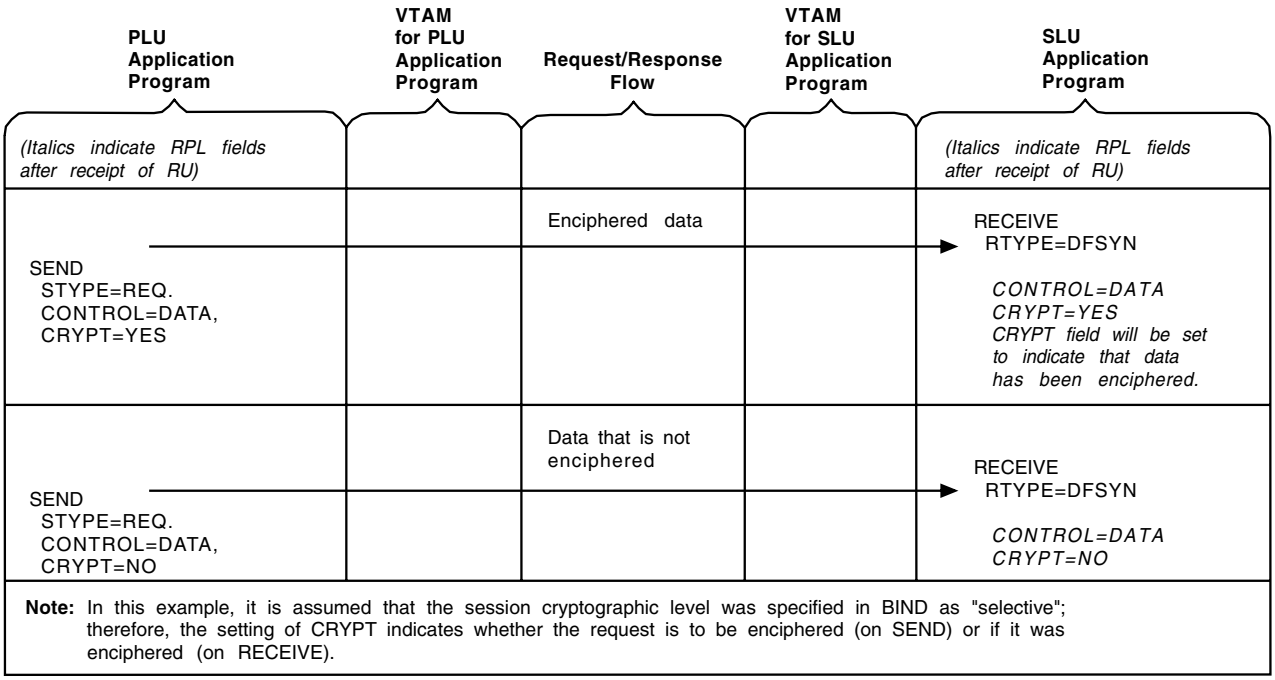


Figure 141. PLU application program and the SLU application program use cryptography in a selective cryptographic session

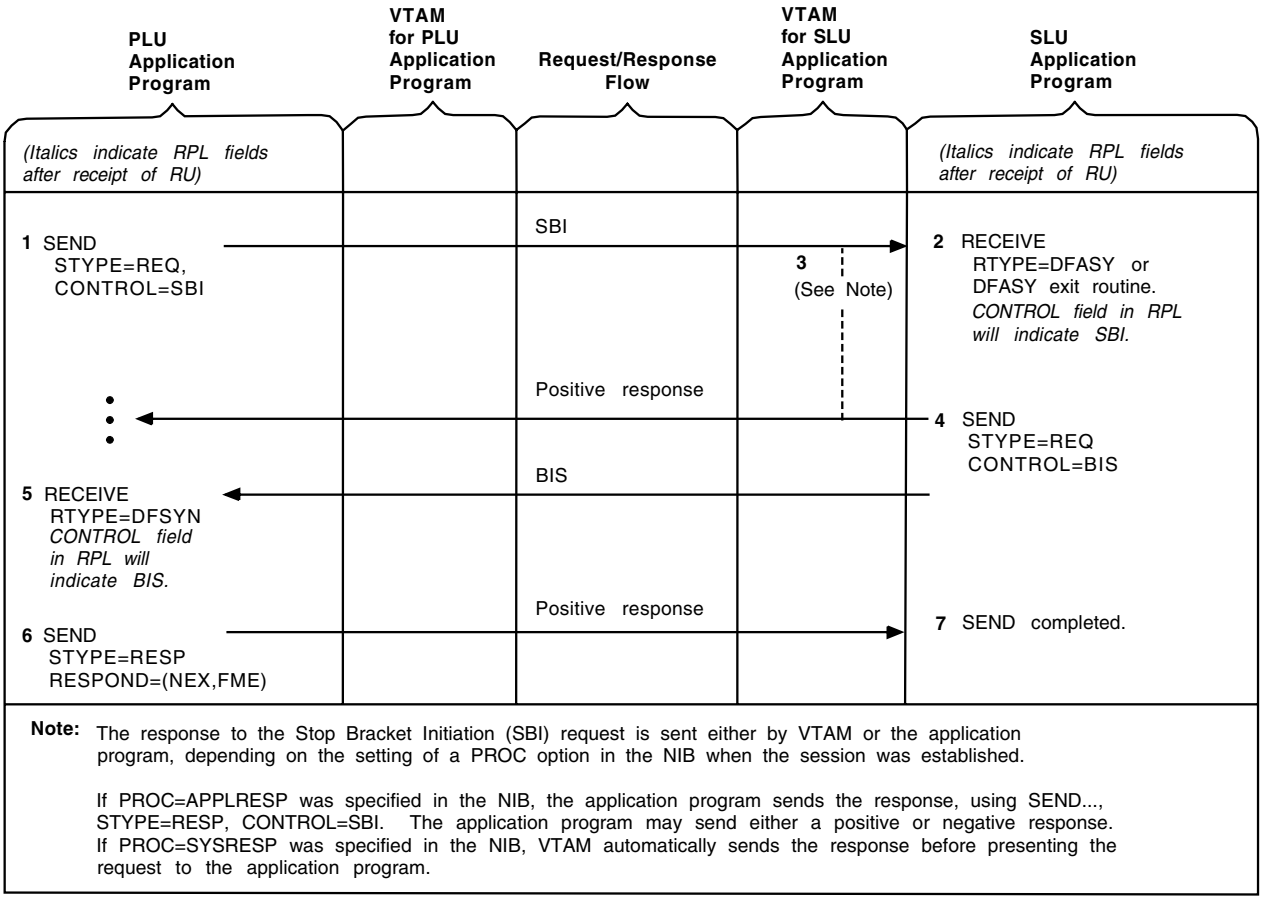


Figure 142. PLU application program stops bracket initiation

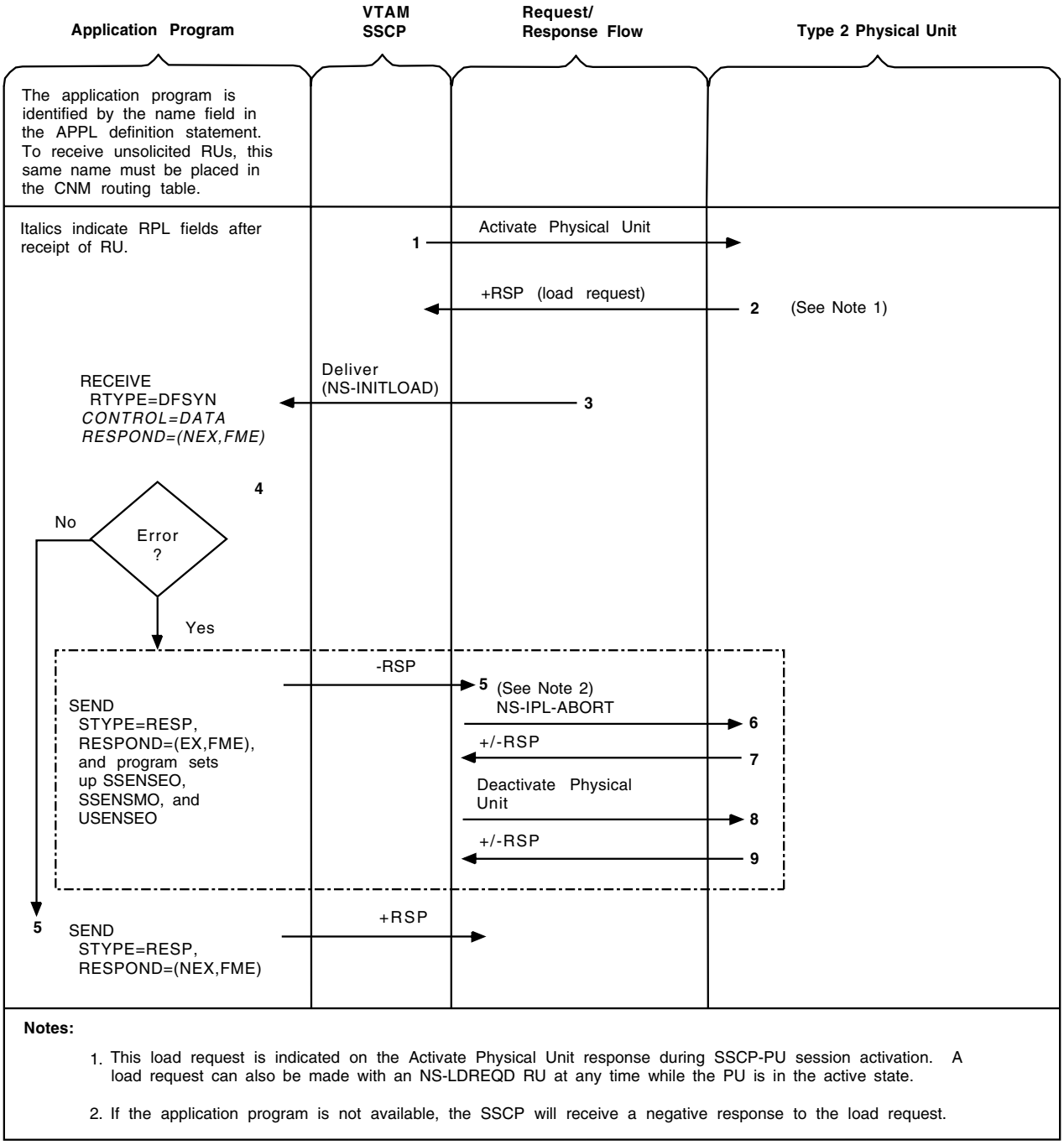
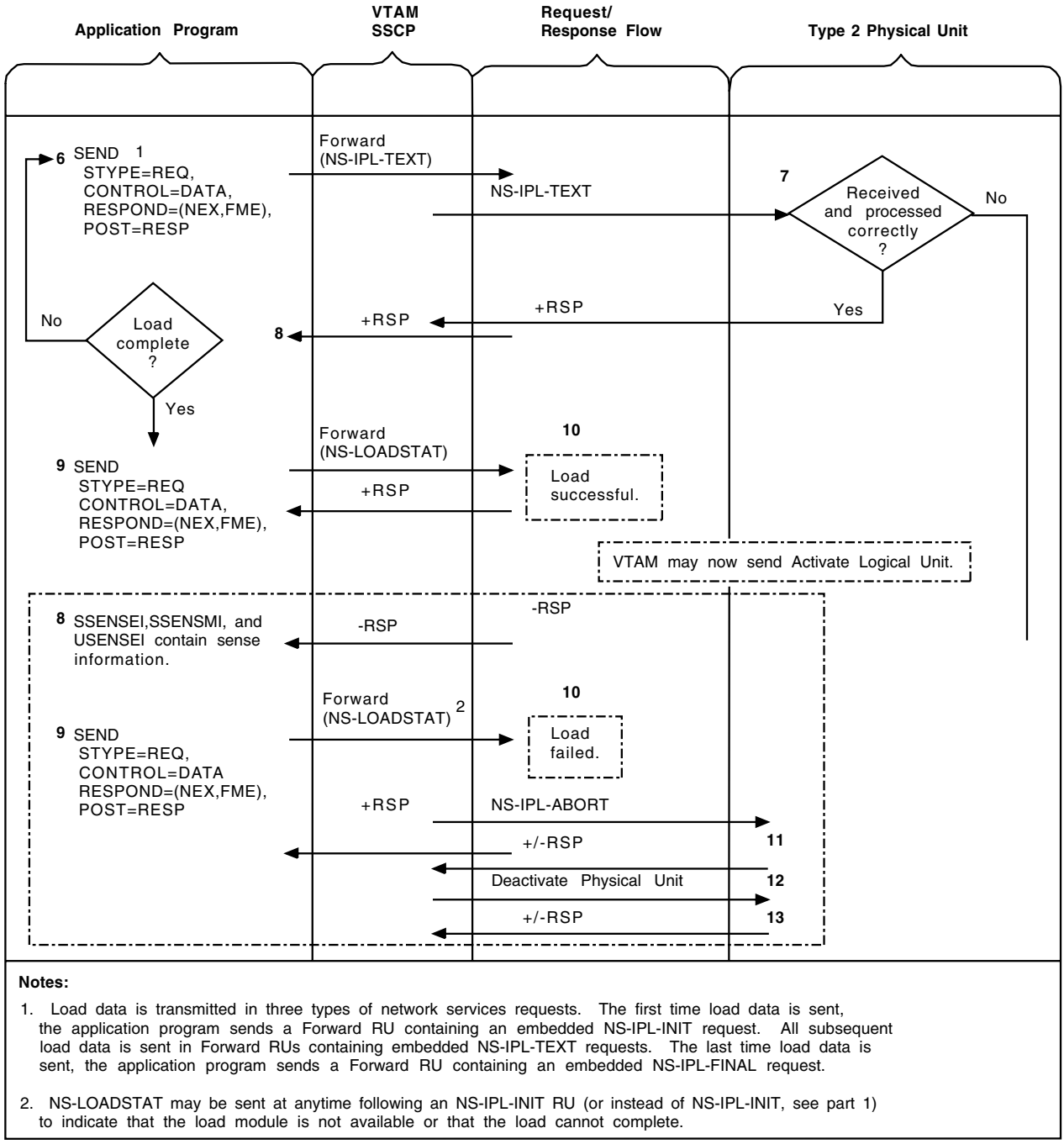


Figure 143. Application program receives a load request during physical unit activation (Part 1 of 2)



Notes:

1. Load data is transmitted in three types of network services requests. The first time load data is sent, the application program sends a Forward RU containing an embedded NS-IPL-INIT request. All subsequent load data is sent in Forward RUs containing embedded NS-IPL-TEXT requests. The last time load data is sent, the application program sends a Forward RU containing an embedded NS-IPL-FINAL request.

2. NS-LOADSTAT may be sent at anytime following an NS-IPL-INIT RU (or instead of NS-IPL-INIT, see part 1) to indicate that the load module is not available or that the load cannot complete.

Figure 144. Application program receives a load request during physical unit activation (Part 2 of 2)

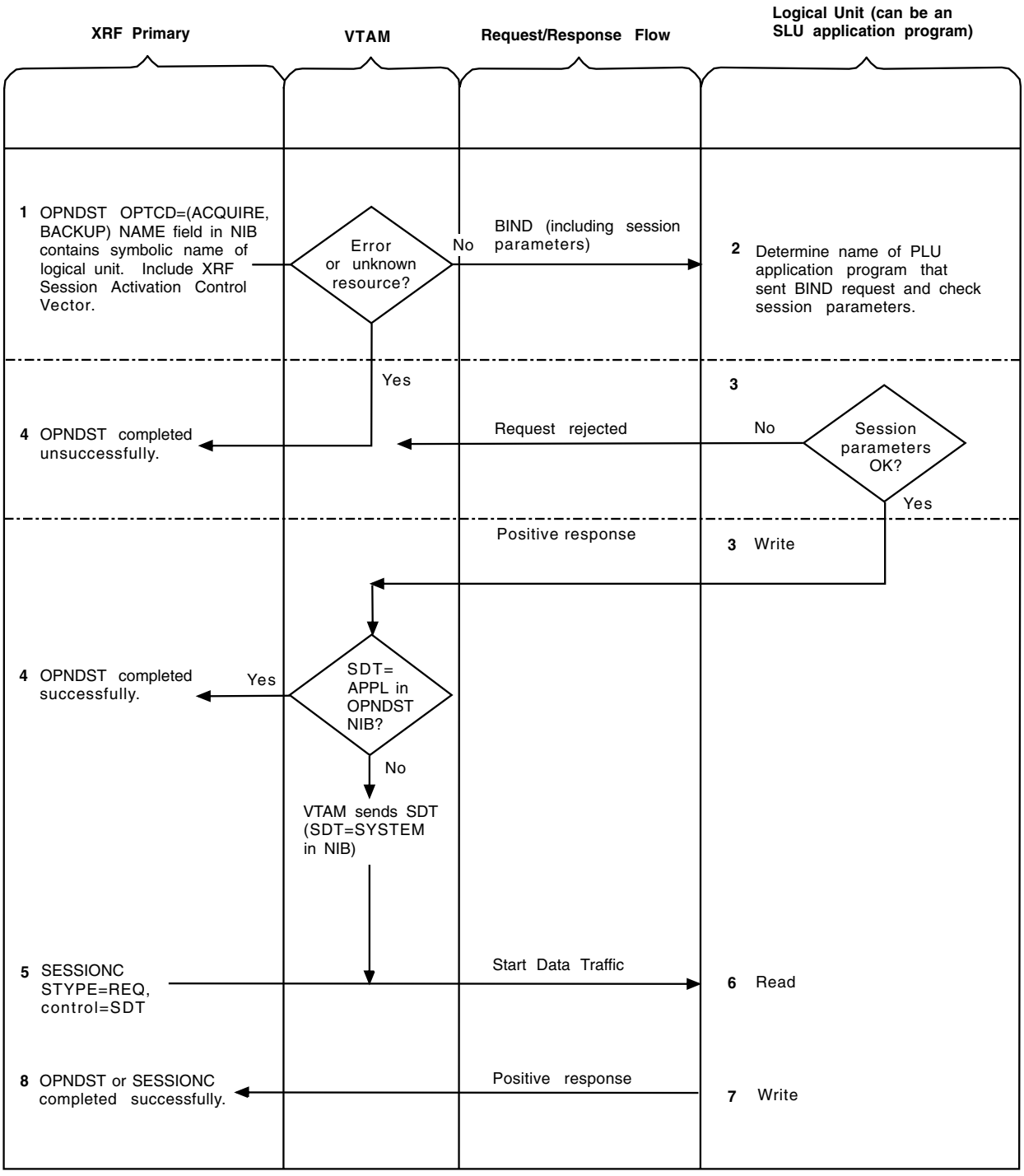


Figure 145. PLU application program acquires (initiates and establishes) a session with an XRF backup-session service-type LU.

For initiating a session from a PLU to a SLU application program, see Figure 130 on page 717 and Figure 131 on page 718.

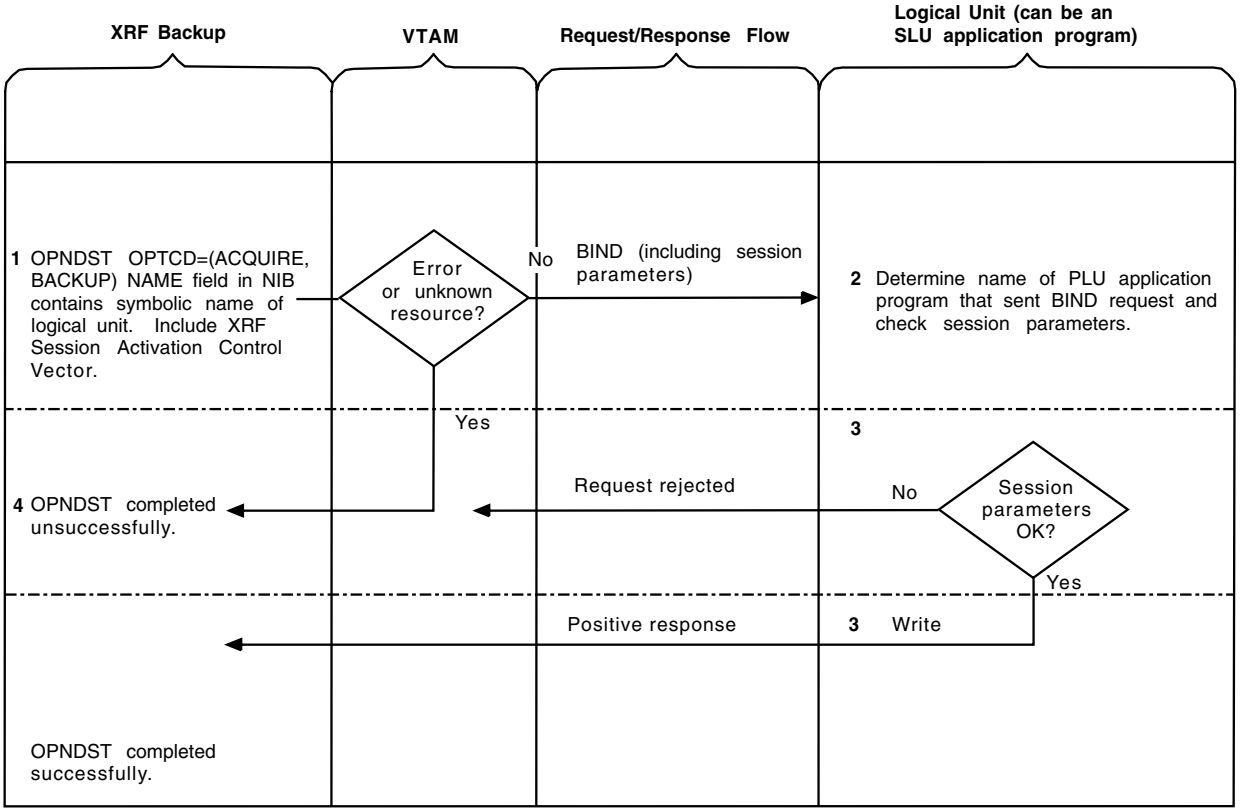


Figure 146. PLU application program acquires (initiates and establishes) a session with an XRF backup-session device-type LU.

For initiating a session from a PLU to a SLU application program, see Figure 130 on page 717 and Figure 131 on page 718.

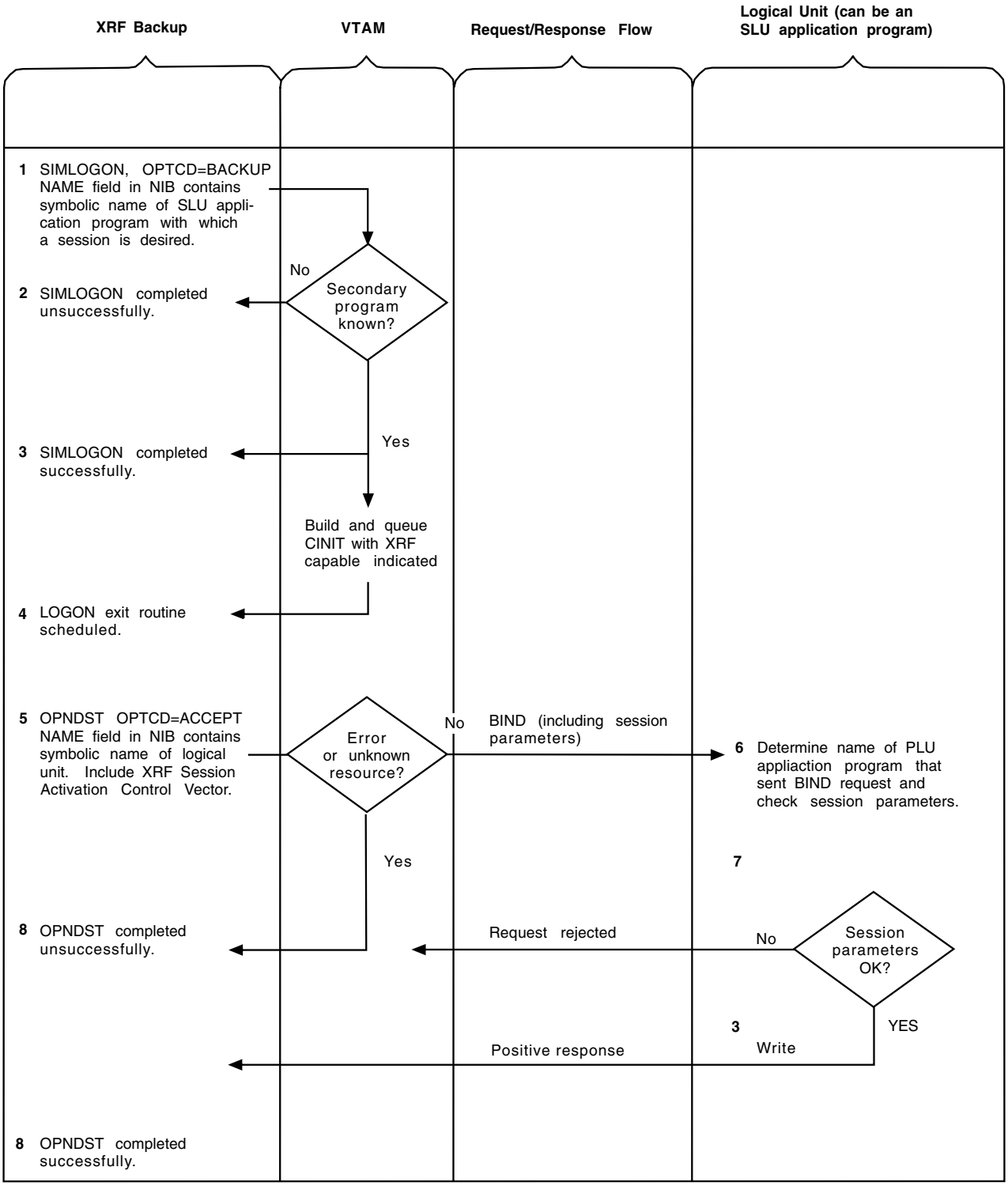


Figure 147. PLU application program issues SIMLOGON to acquire (initiate and establish) an XRF session with a device-type LU.

For initiating a session from a PLU to a SLU application program, see Figure 130 on page 717 and Figure 131 on page 718.

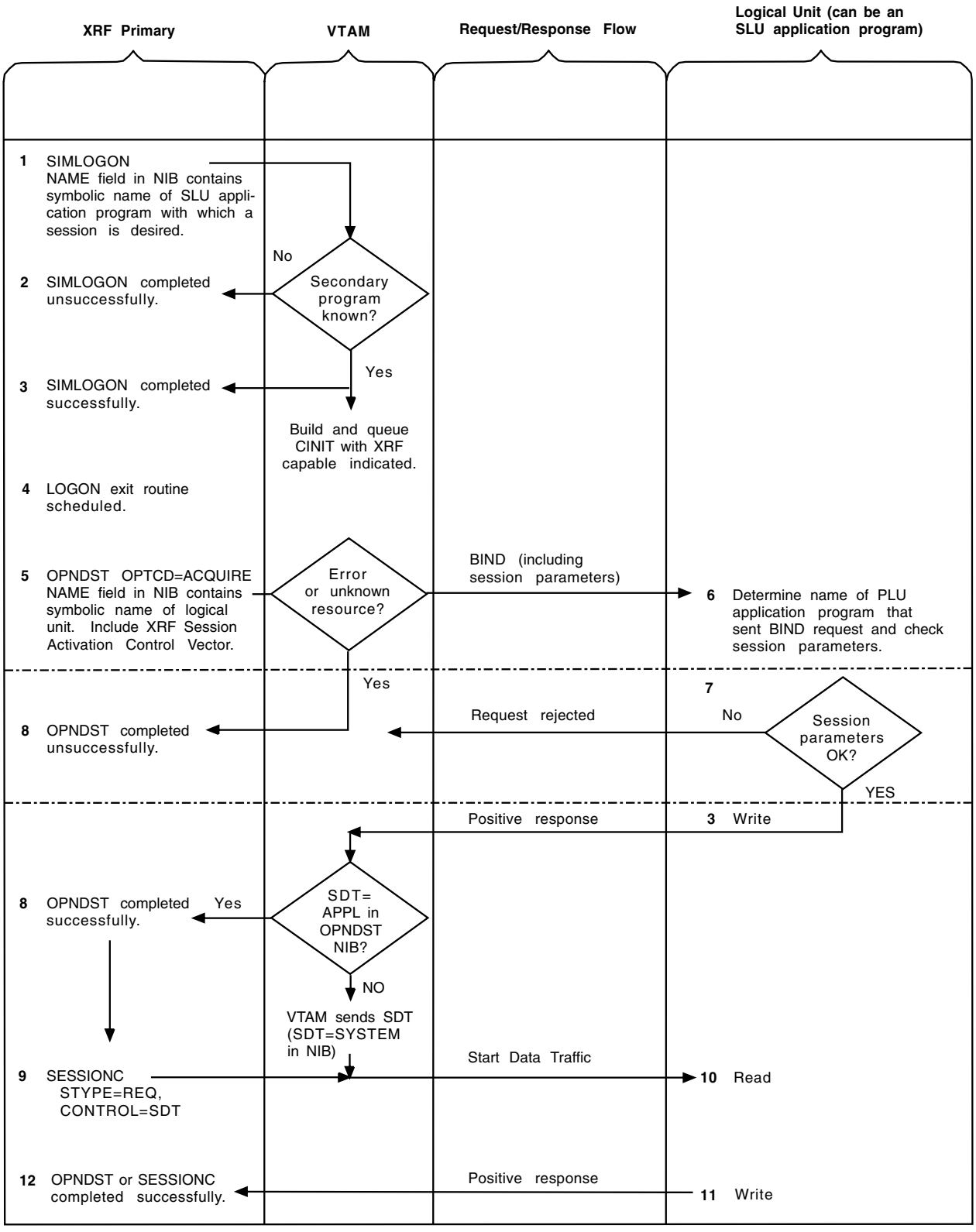


Figure 148. PLU application program issues SIMLOGON to acquire (initiate and establish) an XRF session with a device-type LU.

For initiating a session from a PLU to a SLU application program, see Figure 130 on page 717 and Figure 131 on page 718.

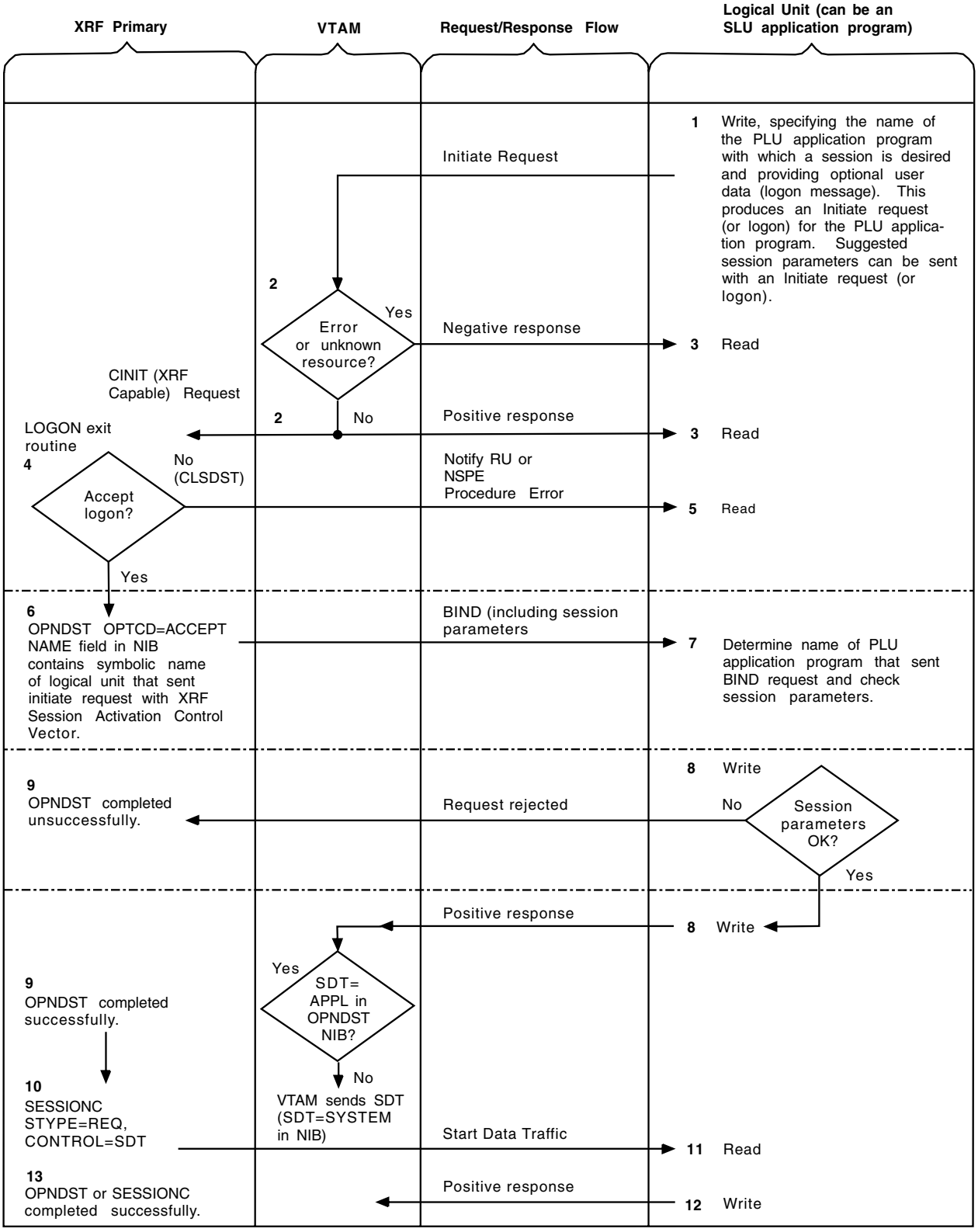


Figure 149. Device-type LU initiates and establishes a session with an XRF primary-PLU application program.

For initiating and establishing a session from a SLU application program, see Figure 128 on page 715 and Figure 129 on page 716.

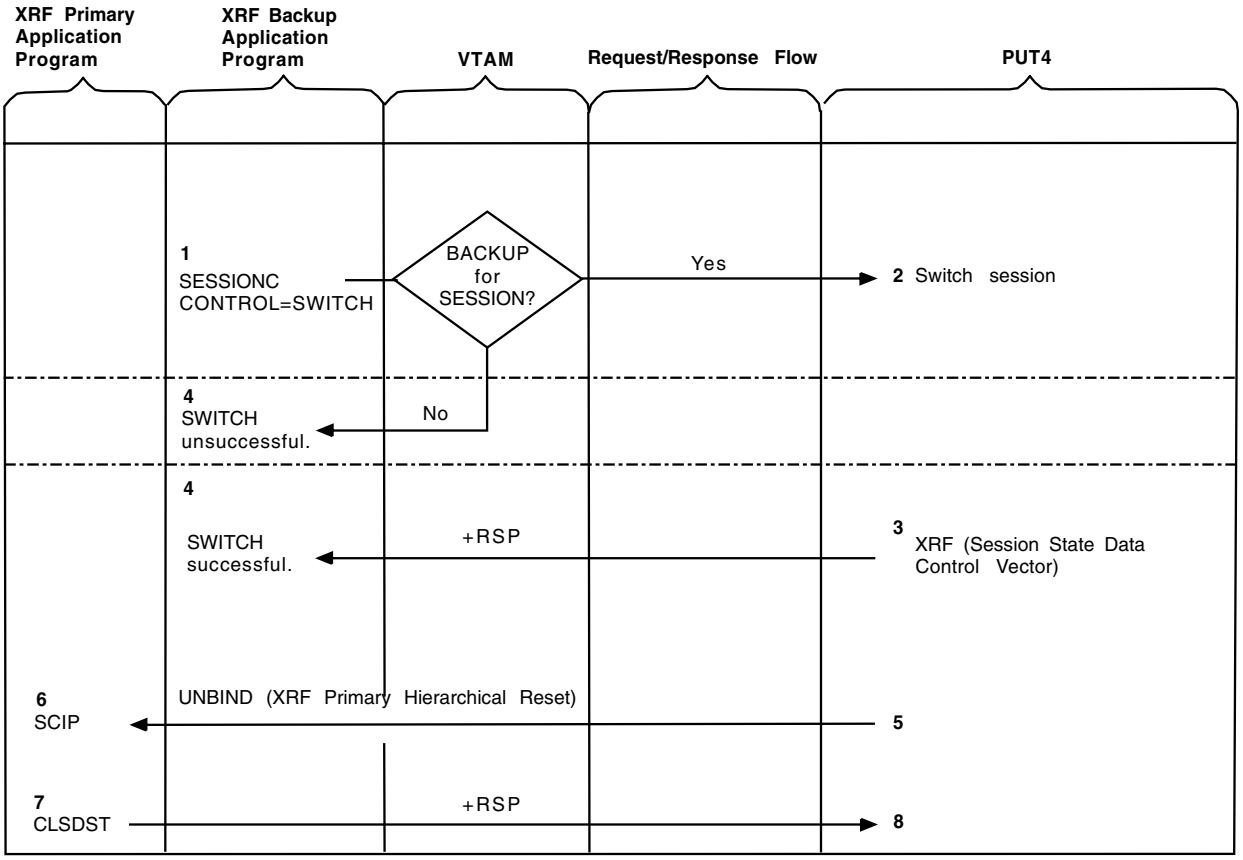


Figure 150. Switch without XRF primary failing first

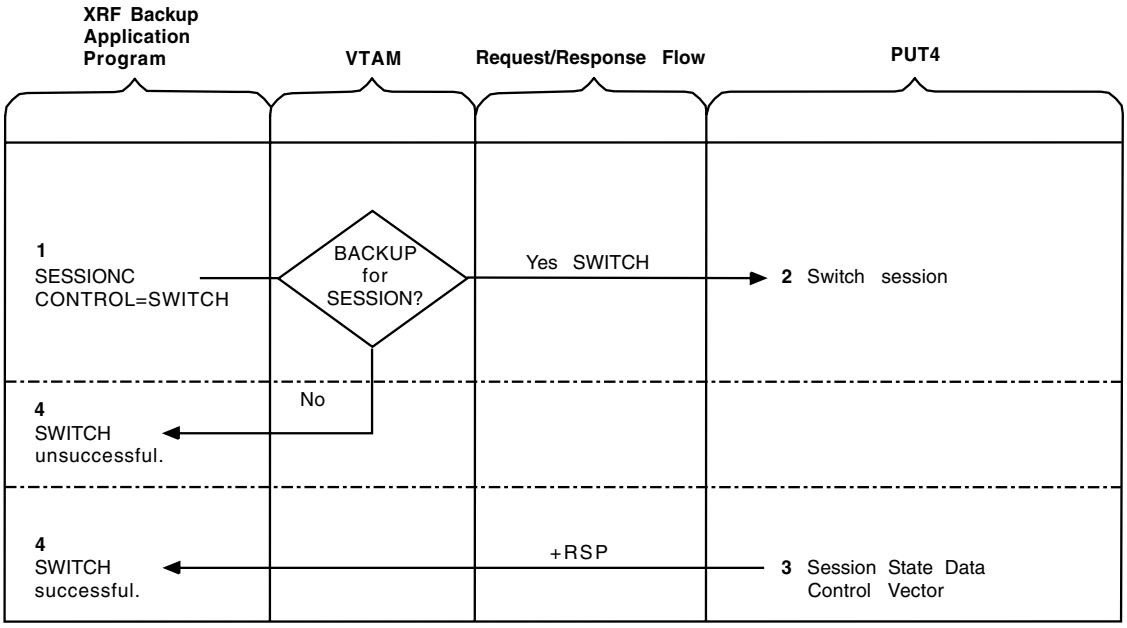


Figure 151. Switch after XRF primary fails

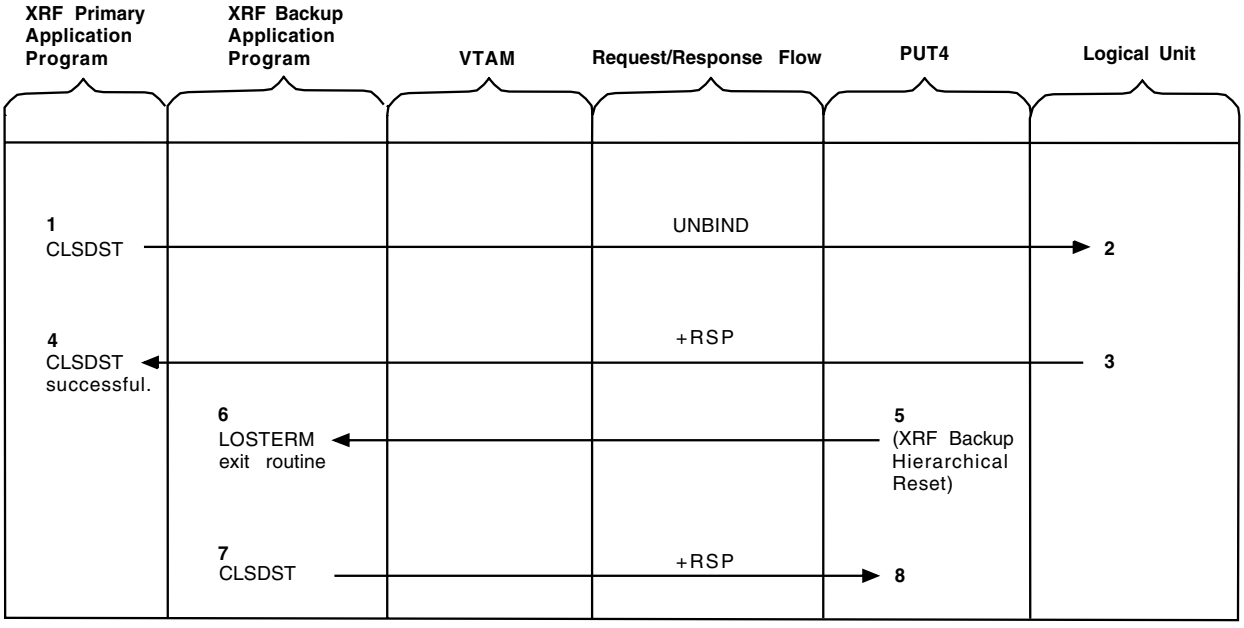


Figure 152. Normal XRF primary end of session

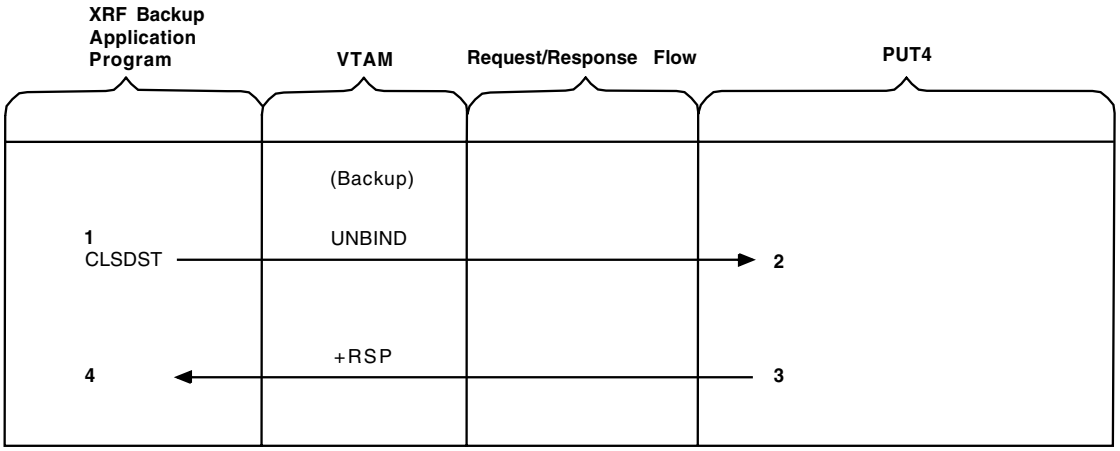


Figure 153. Normal XRF backup end of session with XRF primary

Appendix E. Control block formats and DSECTs

The ACB, EXLST, RPL, and NIB can be initialized, modified, and examined either with manipulative macroinstructions (GENCB, MODCB, SHOWCB, TESTCB) or with assembler instructions. Manipulation with assembler instructions requires access to the internal structure of the control block because displacements and bit settings must be incorporated into the assembler instructions. However, bit settings and displacements are subject to change from release to release. To avoid recoding assembler instructions when such changes occur, use a DSECT. A DSECT is an overlay containing labels that correspond to field displacements, bit settings, and byte values.

IBM-supplied DSECTs are provided as part of the system macroinstruction library in SYS1.MACLIB

You will find descriptions in this appendix. See also "Using DSECT-creating assembler instructions and macroinstructions" on page 272.

A field displacement is the displacement of a field from the beginning of the control block, as defined by the DS (or ORG) instructions in the DSECT. A bit setting is an assembler EQU instruction (such as LABEL1 EQU X'80') that identifies a particular bit or bits. The label could be used as the immediate data byte in a TM instruction, for example. A byte value is also an assembler EQU instruction (such as LABEL2 EQU X'23') that identifies a particular value in a byte. The label could be used as the immediate data byte of a CLI instruction, for example.

The format maps in this appendix, Figure 154 on page 740 through Figure 165 on page 753, show the format of the control blocks. They provide a means by which a dump of the control block can be interpreted, and they make the DSECT descriptions that accompany the maps more easily understood. The format maps and the DSECT descriptions identify both the external field name (the declarative or manipulative macroinstruction keyword as used throughout this book) and the internal field name (DSECT label) for each control block field. The DSECT descriptions are arranged in alphabetical order according to the external field name. Mutually exclusive settings are indicated by indentations in the "Meaning" column. Related settings all appear under the same external name in the "Field" column.

Some DSECTs contain maps to other DSECTs. The DSECTs containing the maps are shown under the following headings:

- "Access-method-support vector list (ISTAMSVL)" on page 781
- "Resource-information vector list (ISTRIVL)" on page 785
- "Application-ACB vector list (ISTVACBV)" on page 789

If you compare listings of the actual DSECTs with the DSECT descriptions provided here, the actual DSECTs are more extensive. The fields that have been eliminated here are primarily fields that are set and used by VTAM, not by the application program. The control block fields that you set or examine should be limited to those fields that are included in the DSECT descriptions in this book.

© Copyright IBM Corp. 2000, 2013

(For this reason, you should not attempt to use a DSECT to initialize a control block completely; use GENCB or the appropriate ACB, EXLST, RPL, or NIB macroinstruction instead.)

ACB (IFGACB)

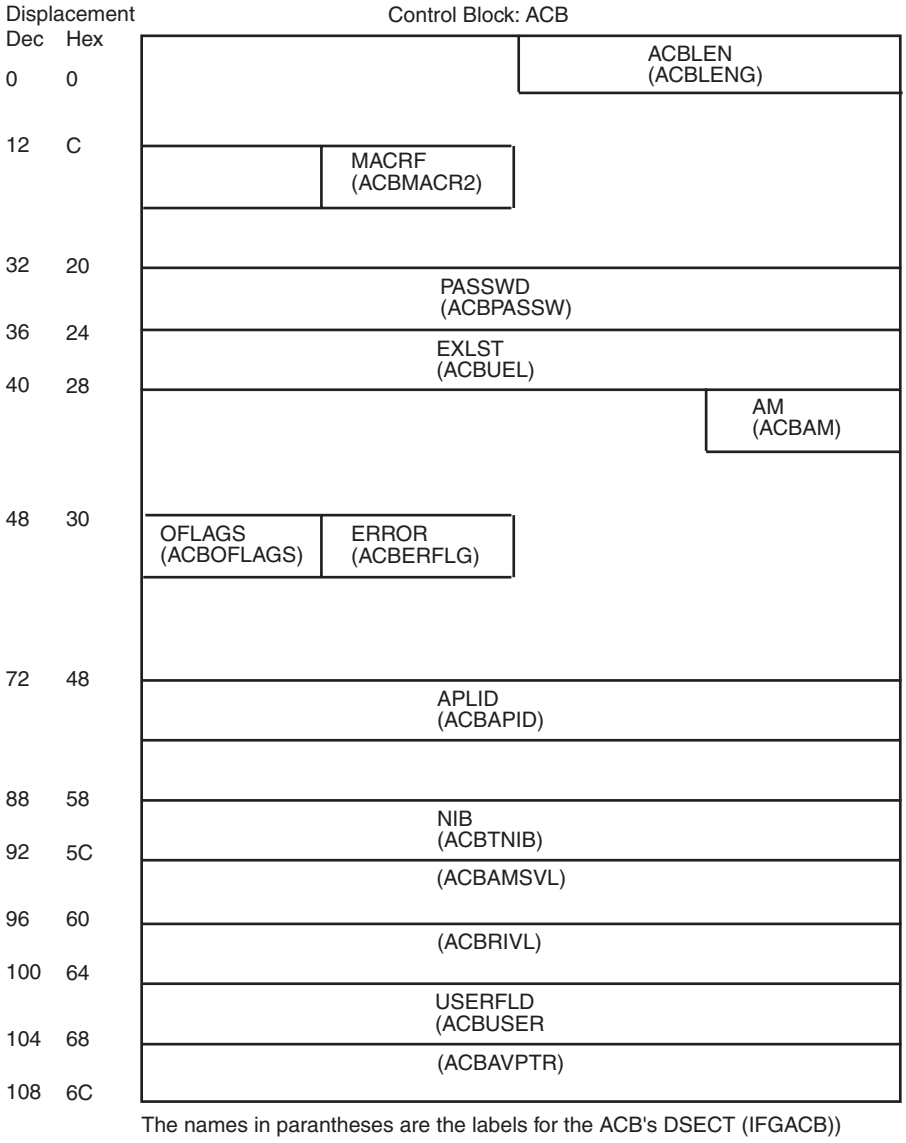


Figure 154. Format of the ACB

Table 105. ACB DSECT (IFGACB)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
ACBLEN	ACBLENG	—	XL2	ACB length	2	2
AM	ACBAM	ACBVTAM	X'60'	AM=VTAM	43	2B
APPLID	ACBAPID	—	A	APPLID address	72	48
ERROR	ACBERFLG	ACBOALR	X'04'	Already open (OPEN)	49	31

Table 105. ACB DSECT (IFGACB) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
		ACBCALR	X'04'	Already closed (CLOSE)	49	31
		ACBONVRT	X'14'	No virtual memory for VTAM	49	31
		—	X'24'	The password specified by the ACB did not match the corresponding password in the APPL entry, or the ACB did not specify a password and the APPL contained one, or the security management product determined that the user was not authorized to open the ACB.	49	31
		ACBCAQNR	X'40'	Outstanding OPNDST OPTCD=ACQUIRE not released	49	31
		ACBCDSNR	X'42'	Destinations not released (CLOSE)	49	31
		ACBRNOCF	X'4C'	Requests are queued or VTAM is waiting for a reply (CLOSE)	49	31
		ACBOANAT	X'50'	VTAM not specified during system generation	49	31
		ACBOAHLT	X'52'	VTAM is halting (OPEN)	49	31
		ACBOAVFY	X'54'	APPLID is not valid (OPEN)	49	31
		ACBOANSN	X'56'	APPLID is name of non-APPL (OPEN)	49	31
		ACBOAPAA	X'58'	APPL is already active (OPEN)	49	31
		ACBOAPNM	X'5A'	No matching APPL found (OPEN)	49	31
		ACBOVINA	X'5C'	VTAM in system but inactive (OPEN)	49	31
		ACBOAPSE	X'5E'	APPLID not in requestor's space	49	31
		ACBOUNDF	X'60'	Undefined system error	49	31
		ACBOAPLE	X'62'	APPLID length not valid (OPEN)	49	31
		ACBOPWSE	X'64'	Password not in requestor's space (OPEN)	49	31
		ACBOPWLE	X'66'	Password length not valid (OPEN)	49	31
		ACBRNOOF	X'68'	Two or more VTAM APPL statements have AUTH=PPO. One is already open and a second is trying to open. (OPEN)	49	31
		ACBOAVSE	X'6A'	Application vector not in requestor's space (OPEN)	49	31
		ACBOALSE	X'6C'	Application vector length invalid (OPEN)	49	31
		ACBTVTCL	X'70'	OPEN or CLOSE rejected; application program is being closed and is unavailable.	49	31
		ACBESME	X'72'	External security manager error (OPEN)	49	31

Table 105. ACB DSECT (IFGACB) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
		ACBOPSNE	X'74'	Takeover rejected because original application did not enable persistence (OPEN)	49	31
		ACBOPSNC	X'76'	Opening application did not specify PERSIST=YES on the ACB (OPEN)	49	31
		ACBOPSMM	X'78'	Conflicting parameter values between the ACB of the opening application and the ACB of the application pending recovery conflict (OPEN)	49	31
		ACBMOPEN	X'7A'	OPEN ACB by monitor application failed because a monitor application was already active	49	31
		ACMSTONP	X'7C'	SNPS takeover OPEN ACB failed because the active application does not accept takeover requests	49	31
		ACBNAUTH	X'8C'	Monitor application not CNM or POA authorized (OPEN)	49	31
		ACBOACT	X'BC'	ACB active	49	31
		ACBCBUSY	X'BC'	ACBBUSY	49	31
		ACBTANAE	X'F4'	Not authorized for SRBEXIT=YES (OPEN)	49	31
		ACBTNBSE	X'F6'	NIB storage address not valid (OPEN)	49	31
		ACBTNBOE	X'FA'	NIB options not valid (OPEN)	49	31
		ACBTRTTE	X'FE'	Duplicate unsolicited RU routing requested	49	31
EXLST	ACBUEL	—	A	EXLST address	36	24
MACRF	ACBMACR2	ACBLOGON	X'08'	MACRF=NLOGON	13	D
NIB	ACBTNIB	—	A	NIB address	88	58
OFLAGS	ACBOFLGS	ACBOPEN	X'10'	OFLAGS=OPEN	48	30
PASSWD	ACBPASSW	—	A	Password address	32	20
	ACBOPT1	ACBPSINS	X'08'	Persistent recovery or takeover application instance	82	52
PERSIST		ACBPLUSC	X'10'	Application program is capable of persistent LU-LU session support	82	52
SRBEXIT		ACBSRBSP	X'20'	Application program specified SRBEXIT on ACB	82	52
		ACBSRBEX	X'40'	Application program specified on APPL statement or ACB that exits are driven under an SRB	82	52
KEEPFRR		ACBKPFRR	X'80'	Application program specified that VTAM keep the FRR stack	82	52
FDX	ACBOPT2	ACBAFDX	X'80'	APPC full-duplex and expedited data supported	83	53

Table 105. ACB DSECT (IFGACB) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
FORCETKO		ACBFRCTO	X'08'	OPEN ACB might trigger MNPS forced takeover processing	83	53
USERFLD	ACBUSER	—	—	User field	100	64
	ACBAMSVL	—	—	Access-method-support vector-list address	92	5C
	ACBRIVL	—	—	Resource-information vector-list address	96	60
	ACBAVPTR	—	—	Application-ACB vector-list address	104	68

ASDP (ISTASDP)

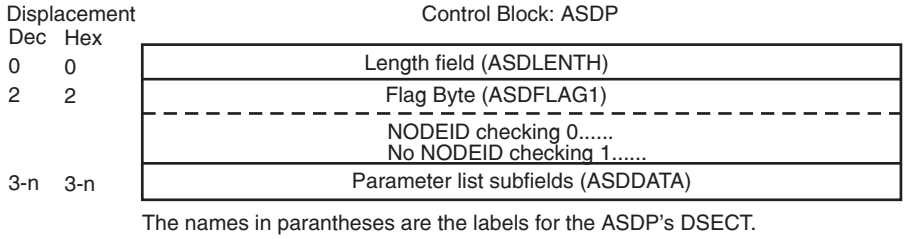


Figure 155. Format of ISTASDP

Displacement		Control Block: ASDSUBFD (Dial Number Subfield)
Dec	Hex	
0	0	Subfield type X'00' (ASDSFTYP)
1	1	Length of data (ASDSFLEN)
2-n	2-n	Dial Number (ASDDIALN)

Figure 156. Dial number subfield

Displacement		Control Block: ASDSUBFD (Direct Call Line Name Subfield)
Dec	Hex	
0	0	Subfield type X'01' (ASDSFTYP)
1	1	Length of data (ASDSFLEN)
2-n	2-n	Direct Call Line Name (ASDDCLNM)

Figure 157. Direct call line name subfield

Displacement		Control Block: ASDSUBFD (IDBLK/IDNUM Subfield)
Dec	Hex	
0	0	Subfield type X'02' (ASDSFTYP)
1	1	Length of data (ASDSFLEN)
2-5	2-5	IDBLK/IDNUM (ASDIDBKN)

Figure 158. IDBLK/IDNUM subfield

Displacement		Control Block: ASDSUBFD (CPNAME Subfield)
Dec	Hex	
0	0	Subfield type X'03' (ASDSFTYP)
1	1	Length of data (ASDSFLEN)
2-n	2-n	CPNAME (ASDCPNAM)

Figure 159. CPNAME subfield

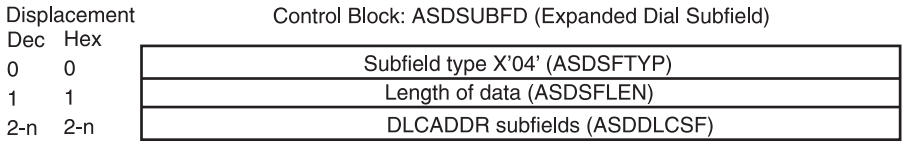


Figure 160. Expanded dial information subfield

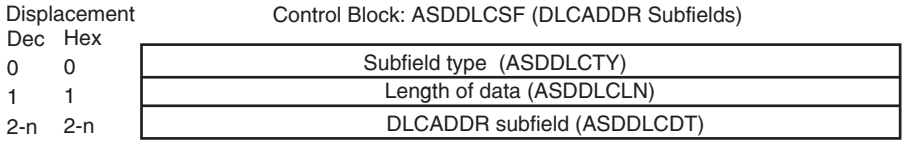


Figure 161. DLCADDR subfield

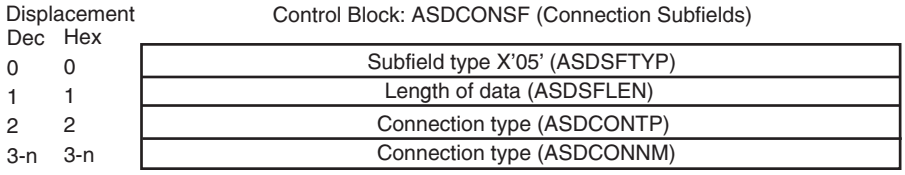


Figure 162. Connection name subfield

Table 106. ISTASDP DSECT

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
ASDLENTH		HL1	Parameter List Length	0	0
ASDFLAG1	ASDNIDCK	X'80'	No NODEID check flag	2	2
ASDDATA		0X	Begin subfield data	4-n	4-n

Table 107. ISTASDP subfield DSECT (ASDSUBFD)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
ASDSFTYP		XL1	Type of subfield	0	0
	ASDTDLNM	X'00'	Dial number subfield type	0	0
	ASDTDCLN	X'01'	Direct call line subfield type	0	0
	ASDTIDBN	X'02'	IDBLK/IDNUM subfield type	0	0
	ASDTCPNM	X'03'	CPNAME subfield type	0	0
	ASDTDLCA	X'04'	Expanded dial information	0	0
	ASDTCONN	X'05'	Connection name	0	0
ASDSFLEN		XL1	Subfield data length	1	1
ASDSFDTA		0X	Subfield data	2	2
ASDDIALN		0C	Dial number	2	2
ASDDCLNM		0C	Direct call line name	2	2

Table 107. ISTASDP subfield DSECT (ASDSUBFD) (continued)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
ASDIDBKN		F	IDBLK/IDNUM	2	2
ASDCPNAM		0C	CPNAME	2	2
ASDCONNM		0C	Connection name	2	2
ASDCONTP		XL1	Connection name type	1	1
	ASDCTGPN	X'00'	GRPNM	0	0

Table 108. ISTASDP DLCADDR subfield DSECT (ASDDLCSF)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
ASDDLCHD		XL2	Header	0	0
ASDDLCTY		XL1	Type	0	0
ASDDLCLN		XL1	Length	1	1
ASDDLCDT		X'00'	Begin DLCADDR	2	2

BLENT (ISTBLENT)

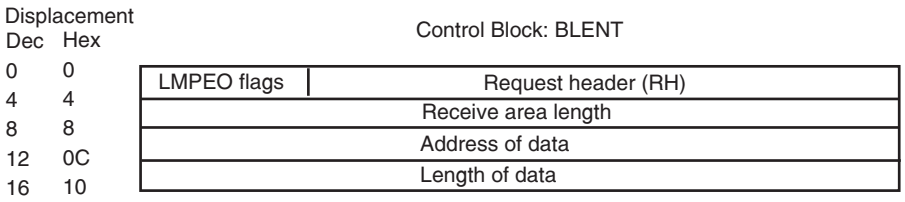


Figure 163. Format of BLENT

Table 109. Buffer list entry DSECT (ISTBLENT)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
BLENT		XL16	Buffer list entry	0	0
BLEFLAGS		X	Flags	0	0
	BLELMPEO	X'C0'	LMPEO control flags	0	0
	BLEBEGRU	X'80'	This entry begins an RU. LMPEO cannot split the RU.	0	0
	BLEENDRU	X'40'	This entry ends an RU. VTAM will begin (with the next buffer list entry) to split the next set of data unless the next buffer list entry precludes it.	0	0
		X'3F'	Reserved	0	0

Table 109. Buffer list entry DSECT (ISTBLENT) (continued)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
BLERH		XL3	Request header	1	1
BLEBUFL		F	Receive area length	4	4
BLEAREA		A	Address of data	8	8
BLERLEN		F	Length of data	12	C

Control vector hex 29 (CV29)

Table 110. Control vector hex 29

Name	Type	Length	Description	Offsets	
				Dec	Hex
ISTV29	STRUCTURE	91	Session state control vector	0	0
V29KEY	CHARACTER	1	VECTOR KEY X'29'	0	0
V29LENTH	UNSIGNED	1	LENGTH OF VECTOR DATA	1	1
V29VECDA	CHARACTER	89	VECTOR DATA	2	2
V29SWTYP	BITSTRING	1	SWITCH TYPE	2	2
V29SWTRQ	B'1111 ....'		SWITCH REQUEST		
V29SWTST	B'.... 1111'		SWITCH STATE		
V29DAFLW	BITSTRING	1	DATA FLOW INDICATORS	3	3
V29DFSTP	B'1... ....'		LAST REQUEST OR RESPONSE WAS: 0 = SENT PLU-TO-SLU 1 = SENT SLU-TO-PLU		
V29DFLPX	B' .1.. ....'		LAST REQUEST OR RESPONSE WAS: 0 = NORMAL-FLOW 1 = EXPEDITED-FLOW		
V29DFLPS	B' ..1. ....'		THE LAST PIU 0 = A REQUEST 1 = A RESPONSE		
V29DFNXP	B' ...1 ....'		EXPEDITED RESPONSE REQUIRED FROM THE SLU 0 = EXPEDITED RESPONSE WAS SENT TO THE PLU 1 = EXPEDITED RESPONSE WAS NOT SENT TO THE PLU		
V29DFNXS	B' .... 1...'		EXPEDITED RESPONSE REQUIRED FROM THE PLU 0 = EXPEDITED RESPONSE WAS SENT TO THE SLU 1 = EXPEDITED RESPONSE WAS NOT SENT TO THE SLU		
V29DFPBF	B' .... .1..'		IF PACING REQUEST WAS SENT TO THE PLU 0 = PACING RESPONSE WAS SENT TO THE SLU BY THE PLU 1 = PACING RESPONSE WAS SENT TO THE SLU BY BOUNDARY FUNCTION		

Table 110. Control vector hex 29 (continued)

Name	Type	Length	Description	Offsets	
				Dec	Hex
*	B' .... ..11'		RESERVED-NOT AVAILABLE		
*	CHARACTER	1	FLAGS	4	4
V29EXTFD	B' 1... ....'		1= EXTENDED NORMAL FLOW FIELDS EXIST AT END OF CV 0= NO EXTENSION EXISTS		
*	B' .111 1111'		RESERVED-NOT AVAILABLE		
V29PS	CHARACTER	43	PRIMARY TO SECONDARY DATA	5	5
V29PSNFL	CHARACTER	24	PLU-TO-SLU NORMAL FLOW INFORMATION	5	5
V29PSNCH	CHARACTER	5	LAST REQUEST SENT PLU-TO-SLU	5	5
V29PSNCS	UNSIGNED	2	SEQUENCE NUMBER OF THE FOLLOWING RH	5	5
V29PSNCR	CHARACTER	3	RH OF FIRST IN CHAIN OR ONLY IN CHAIN REQUEST SENT TO SLU	7	7
V29PSNTH	CHARACTER	10	NORMAL FLOW REQUEST INFORMATION FROM THE TH	10	A
V29PSNTS	UNSIGNED	2	LAST REQUEST SEQUENCE NUMBER SENT PLU-TO-SLU	10	A
V29PSNRH	CHARACTER	3	RH ASSOCIATED WITH THE FOLLOWING RU	12	C
V29PSNTU	CHARACTER	5	FIRST 5 BYTES OF LAST NORMAL- FLOW REQUEST RU SENT PLU-TO-SLU	15	F
V29PSLRP	CHARACTER	9	LAST RESPONSE SENT PLU-TO-SLU	20	14
V29PSLRS	UNSIGNED	2	SEQUENCE NUMBER OF THE LAST RESPONSE SENT	20	14
V29PSLRH	CHARACTER	2	FIRST 2 BYTES OF THE RH ASSOCIATED WITH FOLLOWING RESPONSE	22	16
V29PSLRU	CHARACTER	5	FIRST 5 BYTES OF THE LAST NORMAL FLOW RESPONSE SENT PLU-TO-SLU	24	18
V29PSEFL	CHARACTER	19	PLU-TO-SLU EXPEDITED FLOW INFORMATION	29	1D
V29PSERQ	CHARACTER	10	LAST EXPEDITED-FLOW REQUEST SENT PLU-TO-SLU	29	1D
V29PSEQS	UNSIGNED	2	LAST REQUEST SEQUENCE NUMBER SENT PLU-TO-SLU	29	1D
V29PSEQH	CHARACTER	3	RH ASSOCIATED WITH THE FOLLOWING REQUEST RU	31	1F

Table 110. Control vector hex 29 (continued)

Name	Type	Length	Description	Offsets	
				Dec	Hex
V29PSEQU	CHARACTER	5	FIRST 5 BYTES OF LAST NORMAL- FLOW REQUEST RU SENT PLU-TO-SLU	34	22
V29PSERP	CHARACTER	9	LAST EXPEDITED-FLOW RESPONSE SENT PLU-TO-SLU	39	27
V29PSEPS	UNSIGNED	2	LAST EXPEDITED-FLOW RESPONSE SEQUENCE NUMBER SENT TO PLU-SLU	39	27
V29PSEPH	CHARACTER	2	BYTE 0 AND 1 OF RH ASSOCIATED WITH FOLLOWING RESPONSE RU	41	29
V29PSEPU	CHARACTER	5	FIRST 5 BYTES OF LAST NORMAL- FLOW RESPONSE RU SENT PLU-TO-SLU	43	2B
V29SP	CHARACTER	43	SECONDARY TO PRIMARY DATA	48	30
V29SPNFL	CHARACTER	24	SLU-TO-PLU NORMAL FLOW INFORMATION	48	30
V29SPNCH	CHARACTER	5	LAST REQUEST SENT SLU-TO-PLU	48	30
V29SPNCS	UNSIGNED	2	SEQUENCE NUMBER OF THE FOLLOWING RH	48	30
V29SPNCR	CHARACTER	3	RH OF FIRST IN CHAIN OR ONLY IN CHAIN REQUEST SENT TO PLU	50	32
V29SPNTH	CHARACTER	10	NORMAL FLOW REQUEST INFORMATION FROM THE TH	53	35
V29SPNTS	UNSIGNED	2	LAST REQUEST SEQUENCE NUMBER SENT SLU-TO-PLU FROM THE TH	53	35
V29SPNRH	CHARACTER	3	RH ASSOCIATED WITH THE FOLLOWING RU	55	37
V29SPNTU	CHARACTER	5	FIRST 5 BYTES OF LAST NORMAL- FLOW REQUEST RU SENT SLU-TO-PLU	58	3A
V29SPLRP	CHARACTER	9	LAST RESPONSE SENT SLU-TO-PLU	63	3F
V29SPLRS	UNSIGNED	2	SEQUENCE NUMBER OF THE LAST RESPONSE SENT	63	3F
V29SPLRH	CHARACTER	2	FIRST 2 BYTES OF THE RH ASSOCIATED WITH FOLLOWING RESPONSE	65	41
V29SPLRU	CHARACTER	5	FIRST 5 BYTES OF THE LAST NORMAL FLOW RESPONSE SENT SLU-TO-PLU	67	43
V29SPEFL	CHARACTER	19	SLU-TO-PLU EXPEDITED FLOW INFORMATION	72	48

Table 110. Control vector hex 29 (continued)

Name	Type	Length	Description	Offsets	
				Dec	Hex
V29SPERQ	CHARACTER	10	LAST EXPEDITED-FLOW REQUEST SENT SLU-TO-PLU	72	48
V29SPEQS	UNSIGNED	2	LAST REQUEST SEQUENCE NUMBER SENT SLU-TO-PLU	72	48
V29SPEQH	CHARACTER	3	RH ASSOCIATED WITH THE FOLLOWING REQUEST RU	74	4A
V29SPEQU	CHARACTER	5	FIRST 5 BYTES OF LAST NORMAL- FLOW REQUEST RU SENT SLU-TO-PLU	77	4D
V29SPERP	CHARACTER	9	LAST EXPEDITED-FLOW RESPONSE SENT SLU-TO-PLU	82	52
V29SPEPS	UNSIGNED	2	LAST EXPEDITED-FLOW RESPONSE SEQUENCE NUMBER SENT TO SLU-TO-PLU	82	52
V29SPEPH	CHARACTER	2	BYTE 0 AND 1 OF RH ASSOCIATED WITH FOLLOWING RESPONSE RU	84	54
V29SPEPU	CHARACTER	5	FIRST 5 BYTES OF LAST NORMAL- FLOW RESPONSE RU SENT SLU-TO-PLU	86	56
V29END	CHARACTER		END OF V29 CONTROL BLOCK	91	5B

Table 111. Control vector hex 29 (constants)

Name	Type	Length	Description	Value
V29KEYNO	HEX	1	VECTOR KEY IS '29'X	29
V29PLUS	BIT	0	PLUS ACTIVE - SESSION RESUMED PER APPLICATION REQUEST	0011

EXLST (IFGEXLST)

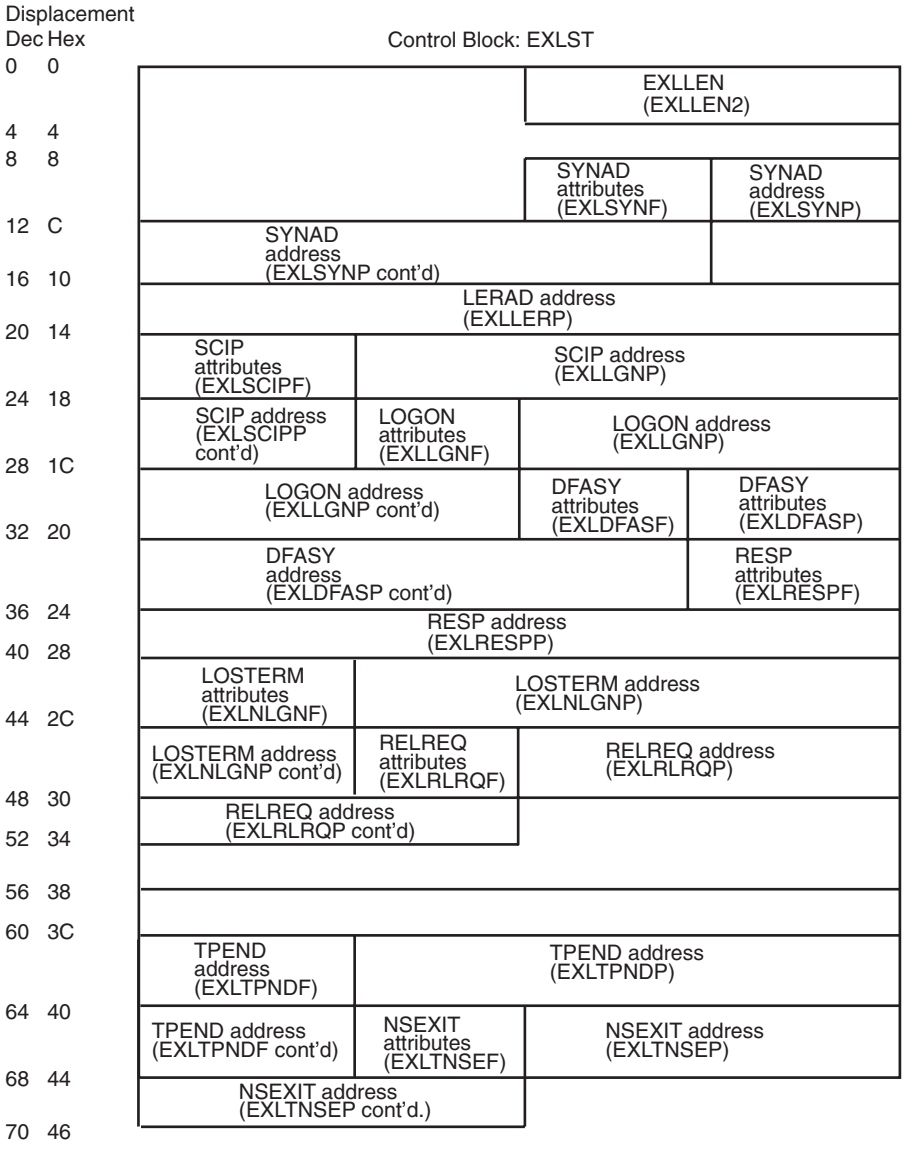


Figure 164. Format of the EXLST

Table 112. EXLST DSECT (IFGEXLST)

Field	DSECT label	DSECT EQU label	DSU or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
EXLLEN	EXLLEN2	—	XL2	EXLST length	2	2
DFASY	EXLDFASF	EXLDFASS	X'80'	DFASY exit present	30	1E
	EXLDFASP	—	A	DFASY exit address	31	1F
LERAD	EXLLERF	EXLLERS	X'80'	LERAD exit present	15	F
	EXLLGNP	—	A	LERAD exit address	16	10
LOGON	EXLLGNF	EXLNLGNS	X'80'	LOGON exit present	25	19
	EXLLGNP	—	A	LOGON exit address	26	1A
LOSTERM	EXLNLGNF	EXLNLGNS	X'80'	LOSTERM exit present	40	38

Table 112. EXLST DSECT (IFGEXLST) (continued)

Field	DSECT label	DSECT EQU label	DSU or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
	EXLNLGNP	—	A	LOSTERM exit address	41	29
NSEXIT	EXLTNSEF	EXLTNSES	X'80'	NSEXIT exit present	65	41
	EXLTNSEP	—	A	NSEXIT address	66	42
RELREQ	EXLRLRQF	EXLRLRQS	X'80'	RELREQ exit present	45	2D
	EXLRLRQP	—	A	RELREQ exit address	46	2E
RESP	EXLRESPF	RXLRESPS	X'80'	RESP exit present	35	23
	EXLRESPP	—	A	RESP exit address	36	24
SCIP	EXLSCIPF	EXLSCIPS	X'80'	SCIP exit present	20	14
	EXLSCIPP	—	A	SCIP exit address	21	15
SYNAD	EXLSYNF	EXLSYNS	X'80'	SYNAD exit present	10	A
	EXLSYNP	—	A	SYNAD exit address	11	B
TPEND	EXLTPNDF	EXLTPNDS	X'80'	TPEND exit present	60	3C
	EXLTPNDP	—	A	TPEND exit address	61	3D

MTS override (ISTMTS)

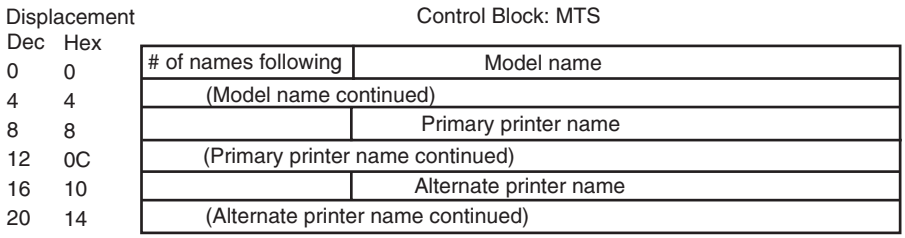


Figure 165. Format of MTS

Table 113. MTS override DSECT (ISTMTS)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
MTSNMCNT		FL1	Number of names that follow	0	0
MTSMDL		CL8	Model name	1	1
MTSPR1		CL8	Primary printer name	9	9
MTSPR2		CL8	Alternate printer name	17	11

NIB (ISTDNIB)

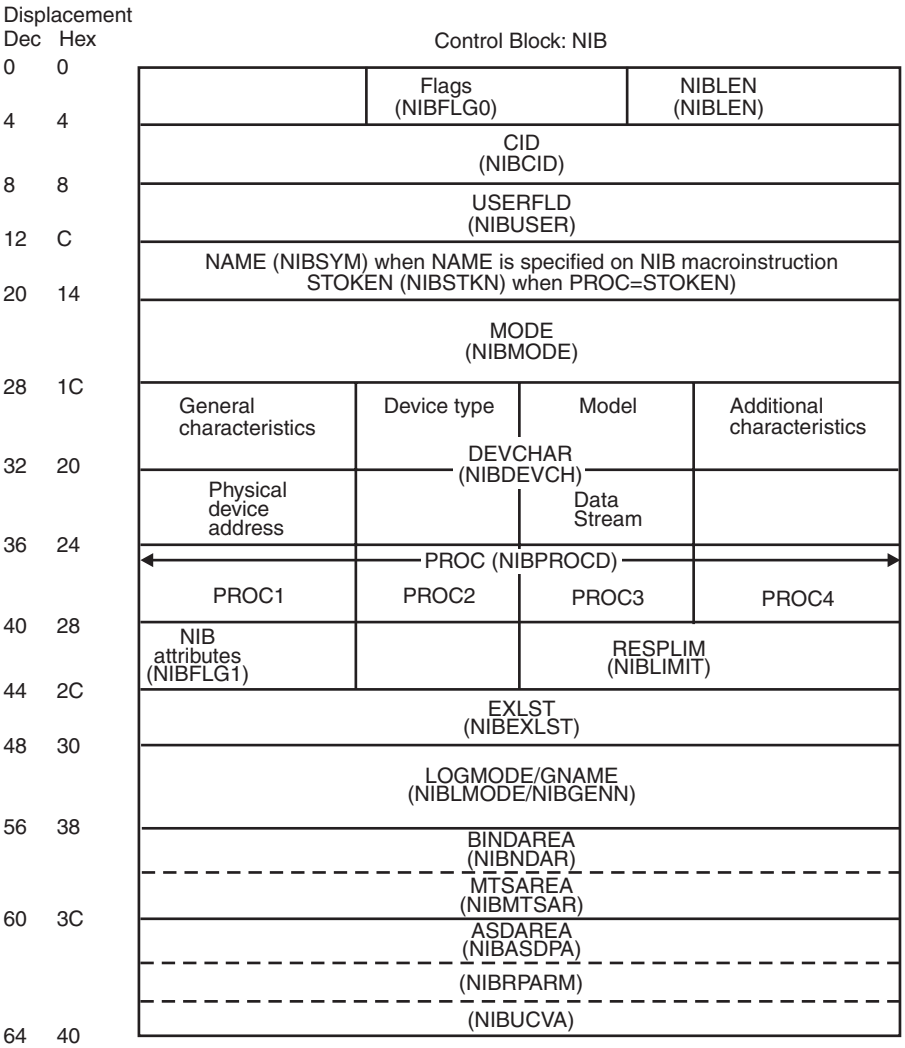


Figure 166. Format of the NIB

For ISTDVCHR, see Table 115 on page 756. For ISTDPROC, see Table 116 on page 761. For ISTDBIND, see Figure 177 on page 818 for DSECT pointed to by BNDAREA.

Table 114. NIB DSECT (ISTDNIB)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
AFFIN	NIBFLG0	NIBAFFIN	X'10'	Indicates (on SETLOGON) application owns all affinities	01	01
BNDAREA	NIBNDAR	—	A	BIND area address	56	38
MTSAREA	NIBMTSAR	—	A	MTS area address **Note:** MTSAREA and BNDAREA are mutually exclusive.	56	38

Table 114. NIB DSECT (ISTDNIB) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
ASDPAREA	NIBFLG1	NIBASDP	X'01'	Indicates application-supplied dial parameters	40	28
ASDPAREA	NIBASDPA	—	A	Address of application-supplied dial parameters	60	3C
CID	NIBCID	—	XL4	Communication ID	4	4
CON	NIBFLG1	NIBCON	X'40'	CON=YES	40	28
DEVCHAR	NIBDEVCH	—	XL8	(See ISTDVCHR, Table 115 on page 756)	28	1C
EXLST	NIBEXLST	—	A	EXLST address	44	2C
GNAME	NIBGENN	—	CL8	Generic name	48	30
LISTEND	NIBFLG1	NIBLAST	X'80'	LISTEND=NO	40	28
LOGMODE/ GNAME	NIBLMODE/ NIBGENN	—	CL8	LOGMODE value	48	30
CINIT request canceled	NIBFLG1	NIBNACLQ	X'08'	If OPNDST OPTCD=ACCEPT failed, the pending CINIT request has been canceled.	40	28
LUAFFIN	NIBFLGO	NIBAFFIN NIBLAFFN	X'12'	Indicates (on OPNDST or OPNSEC) the application owns the affinity for this LU.		
			X'02'	Indicates (on OPNDST or OPNSEC) the application does not own the affinity for this LU.		
MODE	NIBMODE	—	CL8	MODE value	20	14
NAME or PROC= STOKEN	NIBSYM or NIBSTKN	—	CL8	NAME value or STOKEN	12	C
NIBLEN	NIBLEN	—	X	NIB length	3	3
PROC	NIBPROCD	—	XL4	(See ISTDPROC, Table 116 on page 761)	36	24
RESPLIM	NIBLIMIT	—	XL2	RESPLIM value	42	2A
SDT	NIBFLG1	NIBSDAPP	X'20'	SDT=APPL	40	28
USERFLD	NIBUSER	—	XL4	USERFLD value	8	8
—	NIBFLG0	NIBNNAMS	X'80'	Application network name used	1	1
	NIBFLG2	NIBCSEL	X'80'	Indicates selective encryption was used for session	41	29

Table 114. NIB DSECT (ISTDNIB) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
	NIBFLG2	NIBCREQ	X'40'	Indicates required encryption was used for session	41	29
	NIBFLG2	NIBPSPLU	X'20'	Indicates that the application is a PLU	41	29
	NIBFLG2	NIBPSDFS	X'10'	DFSYN data mode setting	41	29
	NIBFLG2	NIBPSDFA	X'08'	DFASY data mode setting	41	29
	NIBFLG2	NIBPSRSP	X'04'	RESP data mode setting	41	29
	NIBFLG2	NIBURCTS	X'02'	Reserved	41	29
	NIBFLG2	NIBSLWRK	X'80'	SIMLOGON successful for this NIB	41	29
–	NIBRPARM	–	A	Restore parameter list address	60	3C
		NIBRBNDP	A	Pointer to BIND information	0	0
		NIBRC29P	A	Pointer to control vector hex 29 information	4	4
		NIBRSQPP	A	Pointer to session qualifier pair	8	8
		NIBRMDNP	A	Pointer to mode name	8	8
		NIBRSIDP	A	Pointer to session instance identifier	12	C
		NIBRFM5P	A	Pointer to FMH5	16	10
		NIBRBIDP	A	Pointer to BID	20	14
		NIBRBISP	A	Pointer to BIS	24	18
–	NIBUCVA	--	A	User CV pointer	60	3C
		NIBVECL	HL2	Length of vector including this field	0	0
		NIBVEC	0X	Control vector data	2	2

Note: Certain fields marked "Reserved" can be set to nonzero values; however, no reserved fields should be examined by an application program.

NIB DEVCHAR (ISTDVCHR)

Table 115. NIB's DEVCHAR DSECT (ISTDVCHR)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
DEVSHCH	DEVCHAR	X	Reserved	28	1C

Table 115. NIB's DEVCHAR DSECT (ISTDVCHR) (continued)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
	DEVINPUT	X'80'	Reserved	28	1C
	DEVOTPUT	X'40'	Reserved	28	1C
	DEVCONVR	X'20'	Reserved	28	1C
	DEVSUBND	X'10'	Reserved	28	1C
	DEVSPS	X'08'	Reserved	28	1C
	DEVNNSPT	X'04'	Reserved	28	1C
	DEVCCTL	X'02'	Reserved	28	1C
	DEVSPOLL	X'01'	Reserved	28	1C
DEVTCODE	DEVCHAR2	X	Reserved	29	1D
	DEV2740	X'01'	Reserved	29	1D
	DEV2741	X'02'	Reserved	29	1D
	DEV1050	X'03'	Reserved	29	1D
	DEVTWX	X'04'	Reserved	29	1D
	DEVWTTY	X'05'	Reserved	29	1D
	DEV115A	X'06'	Reserved	29	1D
	DEV83B3	X'07'	Reserved	29	1D
	DEV2715	X'08'	Reserved	29	1D
	DEV2770	X'09'	Reserved	29	1D
	DEV2780	X'0A'	Reserved	29	1D
	DEV3725	X'0B'	Reserved	29	1D
	DEV3780	X'0C'	Reserved	29	1D
	DEV1130	X'0D'	Reserved	29	1D
	DEV1800	X'0E'	Reserved	29	1D
	DEVDAN	X'0F'	Reserved	29	1D
	DEV3125	X'11'	Reserved	29	1D
	DEV3135	X'12'	Reserved	29	1D
	DEVSYS3	X'13'	Reserved	29	1D
	DEV2701	X'14'	Reserved	29	1D
	DEV2703	X'15'	Reserved	29	1D
	DEV3704	X'16'	Reserved	29	1D
	DEV3705	X'17'	Reserved	29	1D
	DEV2980	X'18'	Reserved	29	1D
	DEV3277	X'19'	Reserved	29	1D
	DEV3284	X'1A'	Reserved	29	1D
	DEV3286	X'1B'	Reserved	29	1D
	DEV3275	X'1C'	Reserved	29	1D
	DEV3741	X'1D'	Reserved	29	1D
	DEV3747	X'1E'	Reserved	29	1D

Table 115. NIB's DEVCHAR DSECT (ISTDVCHR) (continued)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
	DEVRSV05	X'1F'	Reserved	29	1D
	DEVRSV06	X'20'	Reserved	29	1D
	DEVRSV07	X'21'	Reserved	29	1D
	DEVRSV08	X'22'	Reserved	29	1D
	DEVMTA	X'28'	Reserved	29	1D
	DEV2972	X'33'	Reserved	29	1D
	DEV3271	X'34'	Reserved	29	1D
	DEVCC	X'35'	Reserved	29	1D
	DEV3272	X'36'	Reserved	29	1D
	DEV1052	X'64'	Reserved	29	1D
	DEV1053	X'65'	Reserved	29	1D
	DEV1054	X'66'	Reserved	29	1D
	DEV1055	X'67'	Reserved	29	1D
	DEV1056	X'68'	Reserved	29	1D
	DEV1057	X'69'	Reserved	29	1D
	DEV1058	X'6A'	Reserved	29	1D
	DEV1092	X'6B'	Reserved	29	1D
	DEV1093	X'6C'	Reserved	29	1D
	DEVLU	X'6D'	Reserved	29	1D
	DEV545	X'78'	Reserved	29	1D
	DEV1017	X'79'	Reserved	29	1D
	DEV1018	X'7A'	Reserved	29	1D
	DEV2203	X'7B'	Reserved	29	1D
	DEV2213	X'7C'	Reserved	29	1D
	DEV2265	X'7D'	Reserved	29	1D
	DEV2502	X'7E'	Reserved	29	1D
	DEV50	X'7F'	Reserved	29	1D
	DEV1255	X'80'	Reserved	29	1D
	DEV5496	X'81'	Reserved	29	1D
DEVMCODE		X	Device model code	30	1E
	DEVMOD1	X'00'	Device is designated as Model 1	30	1E
	DEVMOD2	X'01'	Device is designated as Model 2	30	1E
DEVFLAGS		X	Flags	31	1F
	DEVFCCTL	X'F0'	Reserved	31	1F
	DEVCBSC	X'80'	Reserved	31	1F
	DEVCSSL	X'40'	Reserved	31	1F
	DEVCRVB	X'20'	Reserved	31	1F

Table 115. NIB's DEVCHAR DSECT (ISTDVCHR) (continued)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
	DEVCSWL	X'10'	LU associated with PU on switched line	31	1F
	DEVCHAR3	X'0F'	Reserved	31	1F
	DEVCATTN	X'08'	Reserved	31	1F
	DEVCCHEK	X'04'	Reserved	31	1F
	DEVCSTCL	X'02'	Reserved	31	1F
	DEVCSPLN	X'01'	Reserved	31	1F
DEVPHYSA		X	Physical device address, for example, address of 3270 "from" terminal for copy operation. Valid only for certain LUs.	32	20
DEVMODE		X	Reserved	33	21
	DEVREC	X'80'	Reserved	33	21
	DEVBASIC	X'40'	Reserved	33	21
		X'3F'	Reserved	33	21
	DEVADVFE	X'01'	3270 extended data stream	33	21
DEVAUXTP			Device data stream compatibility characteristics	34	22
	DEVA2780	X'90'	2780 data stream	34	22
	DEVA1050	X'80'	1050 data stream	34	22
	DEVA2740	X'40'	2740 data stream	34	22
	DEVA83B3	X'30'	83B3 data stream	34	22
	DEVATWX	X'20'	TWX data stream	34	22
	DEVA115A	X'10'	115A data stream	34	22
	DEVAWTTY	X'08'	WTTY data stream	34	22
	DEVA2741	X'04'	2741 data stream	34	22
	DEVAUND	X'00'	No additional data stream characteristics defined	34	22
	DEVA3780	X'91'	Reserved	34	22
	DEVARS05	X'92'	Reserved	34	22
	DEVARS06	X'93'	Reserved	34	22
	DEVARS07	X'94'	Reserved	34	22
	DEVARS08	X'95'	Reserved	34	22
DEVLANG		X	Reserved	35	23
	DEVQUERY	X'80'	Query bit	35	23
	DEVLANG	X'7F'	Language code	35	23
		X'02'	Code=ARA Arabic (language is written from right to left and is not supported by MVS Message Service)	35	23

Table 115. NIB's DEVCHAR DSECT (ISTDVCHR) (continued)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
		X'03'	Code=CHT Traditional Chinese	35	23
		X'04'	Code=CHS Simplified Chinese	35	23
		X'05'	Code=DAN Danish	35	23
		X'06'	Code=DEU German	35	23
		X'07'	Code=DES Swiss German	35	23
		X'08'	Code=ELL Greek	35	23
		X'09'	Code=ENG UK English	35	23
		X'00'	No code US English (default)	35	23
		X'01'	Code=ENU US English	35	23
		X'0A'	Code=ESP Spanish	35	23
		X'0B'	Code=FIN Finnish	35	23
		X'0C'	Code=FRA French	35	23
		X'0D'	Code=FRB Belgian French	35	23
		X'0E'	Code=FRC Canadian French	35	23
		X'0F'	Code=FRS Swiss French	35	23
		X'10'	Code=HEB Hebrew (language is written from right to left and is not supported by MVS Message Service)	35	23
		X'12'	Code=ISL Icelandic	35	23
		X'13'	Code=ITA Italian	35	23
		X'14'	Code=ITS Swiss Italian	35	23
		X'11'	Code=JPN Japanese	35	23
		X'15'	Code=KOR Korean	35	23
		X'16'	Code=NLD Dutch	35	23
		X'17'	Code=NLB Belgian Dutch	35	23
		X'18'	Code=NOR Norwegian	35	23
		X'19'	Code=PTG Portuguese	35	23
		X'1A'	Code=PTB Brazil Portuguese	35	23
		X'1B'	Code=RMS Rhaeto-Romanic	35	23
		X'1C'	Code=RUS Russian	35	23
		X'1D'	Code=SVE Swedish	35	23
		X'1E'	Code=THA Thai	35	23
		X'1F'	Code=TRK Turkish	35	23
		X'3F'	No code unknown language code	35	23

Note: Certain fields marked “Reserved” can be set to nonzero values; however, no reserved fields should be examined by an application program.

NIB PROC (ISTDPROC)

Table 116. NIB's PROC DSECT (ISTDPROC)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
PROC	PROPROC1	PROTRUNC	X'40'	PROC=TRUNC	36	24
		PROXPOPT	X'20'	PROC=BINARY	36	24
		PRODFASY	X'10'	PROC=DFASYX	36	24
		PRORESPX	X'08'	PROC=RESPX	36	24
		PROCA	X'04'	PROC=CA	36	24
		PROCS	X'02'	PROC=CS	36	24
		PRORPLC	X'01'	PROC=RPLC	36	24
	PROPROC2	PROERPO	X'40'	PROC=NERPOUT	37	25
		PROLGOT	X'20'	PROC=NLGOUT	37	25
		PROSYSR	X'10'	PROC=APPLRESP	37	25
		PROFIFOR	X'08'	PROC=ORDRESP	37	25
		PRONTFL	X'04'	PROC=NTMFLL	37	25
		PROEMLC	X'02'	PROC=ELC	37	25
		PROCFTX	X'01'	PROC=CONFTXT	37	25
		PRONEGBD	X'80'	PROC=NEGBIND	37	25
	PROPROC3	PROERPI	X'40'	PROC=NERPIN	38	26
		PROLGIN	X'20'	PROC=NLGIN	38	26
		PRONTO	X'10'	PROC=NTIMEOUT	38	26
		PROMONIT	X'04'	PROC=MONITOR	38	26
	PROPROC4	PROEIB	X'80'	PROC=EIB	39	27
		PROCNDCS	X'40'	PROC=CONDCS	39	27
		PROSTOKN	X'20'	PROC=STOKEN	39	27
		PROMODB	X'08'	PROC=BLOCK	39	27
		PROMODM	X'04'	PROC=MSG	39	27
		PROMODT	X'02'	PROC=TRANS	39	27
		PROMODC	X'01'	PROC=CONT	39	27

NRIPL (ISTNRIPL)

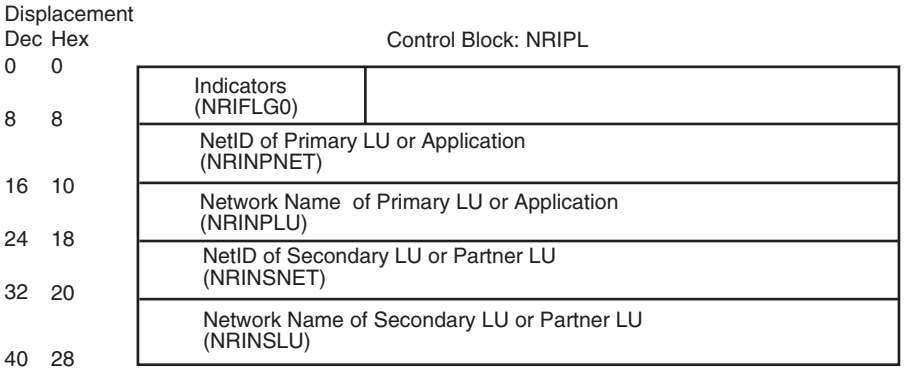


Figure 167. Format of NRIPL

Table 117. NRIPL DSECT (ISTNRIPL)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when field value	Dec offset	Hex offset
ISTNRIPL			Network Resource Identifier Parameter List	0	0
NRIFLG0		XL1	Indicators	0	0
	NRIASLU	X'80'	Application PLU/SLU Indication (not meaningful at ATTN.CNOS exit) ON - Application is SLU OFF - Application is PLU	0	0
	NRINCONW	X'40'	Contention Winner indication (only at ATTN.FMH5 and ATTN.LOSS exits) ON - SLU is Contention Winner OFF - PLU is Contention Winner	0	0
		X'3C'	Reserved	0	0
	NRINNAMS	X'02'	Application is using its Network Name on this session while supporting a Generic Name (only at LOGON and SCIP exits)	0	0
		X'01'	Reserved	0	0
		CL7	Reserved	1	1
NRIPNET		CL8	NetID of Primary LU (for ATTN.CNOS Application NetID)	8	8
NRINPLU		CL8	Primary LU Network Name (for ATTN.CNOS exit: Application Network Name)	16	10
NRINSNET		CL8	NetID of Secondary LU (for ATTN.CNOS exit: Partner LU NetID)	24	18
NRINSLU		CL8	Secondary LU Network Name (for ATTN.CNOS exit: Partner LU Network Name)	32	20

Request/response header (ISTRH)

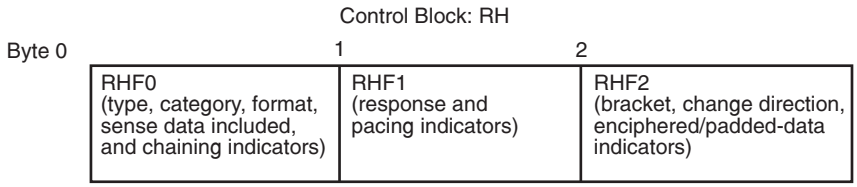


Figure 168. Format of the RH

This note lists the meanings of the codes in the following table.

Note: Legend:

(Q) Means applicable to a request header

(S) Means applicable to a response header

(B) Means applicable to either a request or response header.

Table 118. Request/response header DSECT (ISTRH)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
RH		XL3			
RHF0		X	RH byte 0	0	0
	RHQS	X'80'	RH type indicator: **(Q) 0** request **(S) 1** response	0	0
	RHQSREQ	X'00'	Request unit	0	0
	RHQSRSP	X'80'	Response unit	0	0
	RHRUCAT	X'60'	RU category indicator: **(B) 00** FM data **(B) 01** network control **(B) 10** data flow control **(B) 11** session control	0	0
	RHFMD	X'00'	Function management data RU	0	0
	RHNC	X'20'	Network control RU	0	0
	RHDFC	X'40'	Data flow control RU	0	0
	RHSC	X'60'	Session control RU	0	0
	RHF0B3	X'10'	**(B)** Reserved	0	0

Table 118. Request/response header DSECT (ISTRH) (continued)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
	RHFI	X'08'	Format indicator:	0	0
			(B) 0 no FM header, or character coded network services RU		
			(B) 1 FM header included, or field formatted network services RU		
	RHSDI	X'04'	Sense data included indicator:	0	0
			(B) 0 sense data not included		
			(B) 1 sense data included		
	RHBCI	X'02'	Begin chain indicator:	0	0
			(Q) 0 not beginning of chain		
			(B) 1 beginning of chain		
	RHECI	X'01'	End chain indicator:	0	0
			(Q) 0 not end of chain		
			(B) 1 end of chain		
RHF1		X	RH byte 1	1	1
	RHDR1	X'80'	Definite response 1 indicator:	1	1
			(Q) 0 definite response 1 not required		
			(Q) 1 definite response 1 required		
			(S) 0 not definite response 1		
			(S) 1 definite response		
	RHCI	X'40'	Compression indicator:	1	1
			(Q) 0 RU is not compressed		
			(Q) 1 RU is compressed		
			(S) reserved		
	RHDR2	X'20'	Definite response 2 indicator:	1	1
			(Q) 0 definite response 2 not required		
			(Q) 1 definite response 2 required		
			(S) 0 not definite response 2		
			(S) 1 definite response 2		

Table 118. Request/response header DSECT (ISTRH) (continued)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
	RHERI	X'10'	Exception response indicator: **(Q) 0** definite or no response required **(Q) 1** exception response required	1	1
	RHRTI	X'10'	**(S) 0** positive response **(S) 1** negative response	1	1
	RHF1B4	X'08'	**(B)** Reserved	1	1
	RHRLWS	X'04'	Request larger window indicator: **(Q) 0** larger pacing window not requested **(Q) 1** larger pacing window requested **(S)** reserved	1	1
	RHQRI	X'02'	Queued response indicator: **(B) 0** the response can flow ahead of the requests **(B) 1** the response cannot flow ahead of the requests **Note:** The value of this field is set on a request and checked on the associated response.	1	1
	RHPI	X'01'	Pacing indicator: **(B)** Reserved for VTAM use	1	1
RHF2		X	RH byte 2	2	2
	RHBBI	X'80'	Begin bracket indicator: **(Q) 0** not begin bracket **(Q) 1** begin bracket **(S)** Reserved	2	2
	RHEBI	X'40'	End bracket indicator: **(Q) 0** not end bracket **(Q) 1** end bracket **(S)** Reserved	2	2
	RHCDI	X'20'	Change direction indicator: **(Q) 0** not change direction **(Q) 1** change direction **(S)** Reserved	2	2
	RHF2B3	X'10'	**(B)** Reserved	2	2

Table 118. Request/response header DSECT (ISTRH) (continued)

DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Dec offset	Hex offset
	RHCSI	X'08'	Code selection indicator:	2	2
			(Q) 0 code 0		
			(Q) 1 code 1		
			(S) Reserved		
	RHEDI	X'04'	Enciphered data indicator:	2	2
			(Q) 0 data is not enciphered		
			(Q) 1 data is enciphered		
			(S) Reserved		
	RHPDI				
	RHPRI	X'02'	Padded data indicator:	2	2
			(B) Reserved for VTAM use		
	RHCEBI	X'01'	Conditional end bracket indicator:	2	2
			(Q) 0 not conditional end bracket		
			(Q) 1 conditional end bracket		
			(S) Reserved		

RPL (IFGRPL)

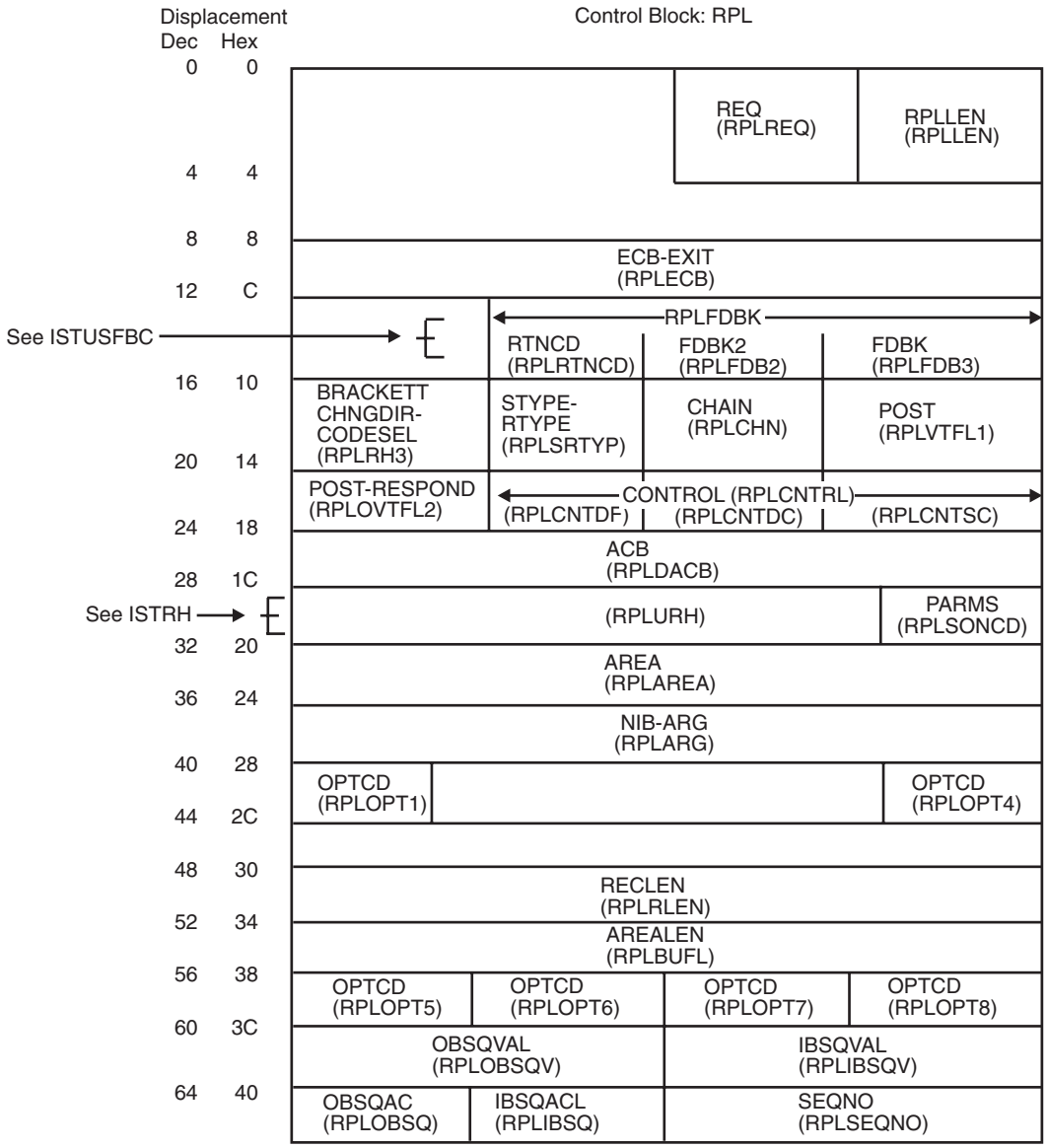


Figure 169. Format of the RPL (Part 1 of 2)

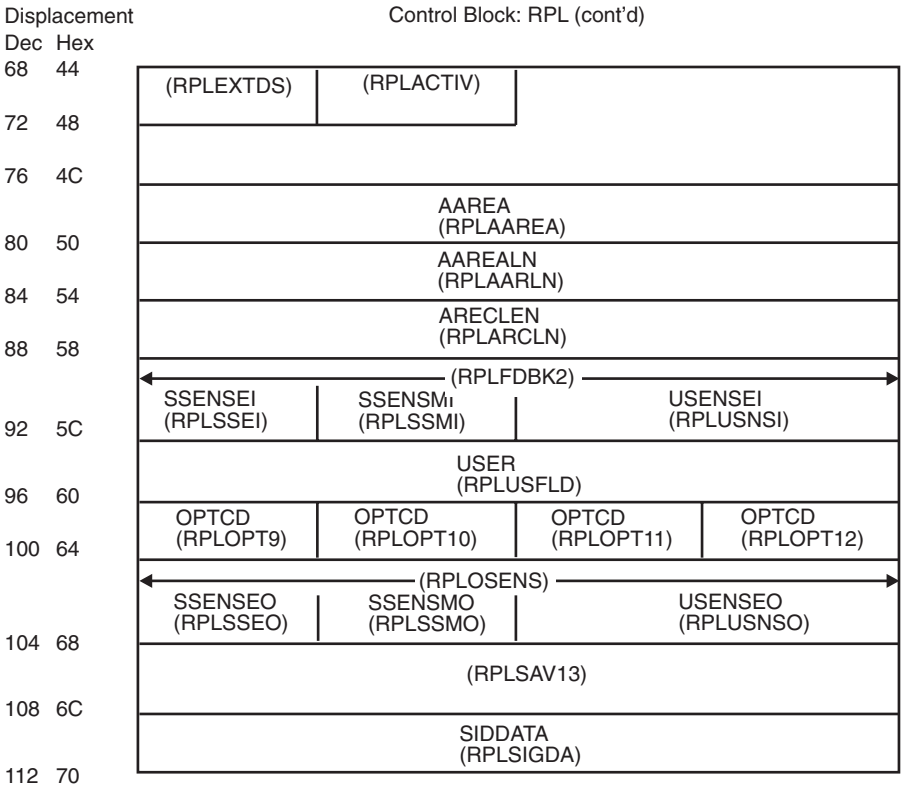


Figure 170. Format of the RPL (Part 2 of 2)

Note: For ISTUSFBC, see Table 120 on page 774. For ISTRH, see Table 118 on page 763.

Table 119. RPL DSECT (IFGRPL)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
AAREA	RPLAAREA	—	A	Alternate data area address	76	4C
AAREALN	RPLAARLN	—	F	Alternate data area length	80	50
ACB	RPLDACB	—	A	ACB address	24	18
AREA	RPLAREA	—	A	Data area address	32	20
AREALEN	RPLBUFL	—	F	Data area length	52	34
ARECLEN	RPLARCLN	—	F	Data length	84	54
ARG	RPLEXTDS	RPLNIB	X'04'	RPLARG points to an NIB	68	44
	RPLARG	—	XL4	NIB address if RPLEXTDS=RPLNIB; CID otherwise (see NIB parameter)	36	24
BRACKET	RPLRH3	RPLBB	X'80'	BRACKET=BB	16	10
		RPLEB	X'40'	BRACKET=EB	16	10
		RPLCEB	X'01'	BRACKET=CEB	16	10
BRANCH	RPLEXTDS	RPLBRANC	X'02'	BRANCH=YES	68	44
CHAIN	RPLCHN	RPLFIRST	X'80'	CHAIN=FIRST	18	12
		RPLMIDLE	X'40'	CHAIN=MIDDLE	18	12

Table 119. RPL DSECT (IFGRPL) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
		RPLLAST	X'20'	CHAIN=LAST	18	12
		RPLONLY	X'10'	CHAIN=ONLY	18	12
CHNGDIR	RPLRH3	RPLCMD	X'20'	CHNGDIR=CMD	16	10
CODESEL	RPLRH3	RPLCS1	X'08'	CODESEL=ALT	16	10
CONTROL (settings mutually exclusive)	RPLCNTRL	—	XL3	Request unit control codes	21	15
	RPLCNTDF	RPLDATA	X'80'	CONTROL=DATA	21	15
		RPLCNCEL	X'40'	CONTROL=CANCEL	21	15
		RPLQC	X'20'	CONTROL=QC	21	15
		RPLQEC	X'10'	CONTROL=QEC	21	15
		RPLCHASE	X'08'	CONTROL=CHASE	21	15
		RPLRELQ	X'04'	CONTROL=RELQ	21	15
	RPLCNTDC	RPLBID	X'80'	CONTROL=BID	22	16
		RPLRTR	X'40'	CONTROL=RTR	22	16
		RPLLUS	X'20'	CONTROL=LUS	22	16
		RPLSIGNL	X'10'	CONTROL=SIGNAL	22	16
		RPLTBIND	X'08'	CONTROL=BIND	22	16
		RPLTUNBD	X'04'	CONTROL=UNBIND	22	16
		RPLSBI	X'02'	CONTROL=SBI	22	16
		RPLBIS	X'01'	CONTROL=BIS	22	16
	RPLCNTSC	RPLSDT	X'80'	CONTROL=SDT	23	17
		RPLCLEAR	X'40'	CONTROL=CLEAR	23	17
		RPLSTSN	X'20'	CONTROL=STSN	23	17
		RPLSHUTD	X'10'	CONTROL=SHUTD	23	17
		RPLSHUTC	X'08'	CONTROL=SHUTC	23	17
		RPLRQR	X'04'	CONTROL=RQR	23	17
		RPLRSHUT	X'02'	CONTROL=RSHUTD	23	17
		RPLSWTCH	X'01'	CONTROL=SWITCH	23	17
CRYPT	RPLEXTDS	RPLCRYP	X'08'	CRYPT=YES	68	44
ECB	RPLOPT1	RPLECBIN	X'01'	External ECB used	40	28
	RPLECB	—	A	Address of external ECB if RPLOPT1 equal to RPLECBIN and RPLEXTDS equal to RPLNEXIT and RPLEXTDS not equal to RPLEXIT (see EXIT parameter)	8	8
EXIT	RPLEXTDS	RPLNEXIT	X'40'	No RPL exit specified	68	44
		RPLEXIT	X'20'	RPL exit specified	68	44

Table 119. RPL DSECT (IFGRPL) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
	RPLECB	—	A	Address of RPL exit if RPLEXTDS not equal to RPLNEXIT and RPLEXTDS equal to RPLEXIT and RPLOPT1 not equal to RPLECBIN (see ECB parameter)	8	8
FDBK	RPLFDB3	—	XL1	Feedback data flags	15	F
FDBK2	RPLFDB2	—	XL1	Feedback reason code (see Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651)	14	E
IBSQAC	RPLIBSQ	RPLISET	X'80'	IBSQAC=SET	65	41
		RPLITST	X'40'	IBSQAC=TESTSET	65	41
		RPLIRSET	X'20'	IBSQAC=RESET	65	41
		RPLIIGN	X'10'	IBSQAC=IGNORE	65	41
		RPLIPOS	X'08'	IBSQAC=TESTPOS	65	41
		RPLINEG	X'04'	IBSQAC=TESTNEG	65	41
		RPLIINV	X'02'	IBSQAC=INVALID	65	41
IBSQVAL	RPLIBSQV	—	XL2	STSN inbound sequence number	62	3E
NIB	RPLEXTDS	RPLNIB	X'04'	RPLARG points to an NIB	68	44
	RPLARG	—	XL4	NIB address if RPLEXTDS=RPLNIB; CID otherwise (see ARG parameter)	36	24
OBSQAC	RPLOBSQ	RPLOSET	X'80'	OBSQAC=SET	64	40
		RPLOTST	X'40'	OBSQAC=TESTSET	64	40
		RPLORSET	X'20'	OBSQAC=RESET	64	40
		RPLOIGN	X'10'	OBSQAC=IGNORE	64	40
		RPLOPOS	X'08'	OBSQAC=TESTPOS	64	40
		RPLONEG	X'04'	OBSQAC=TESTNEG	64	40
		RPLOINV	X'02'	OBSQAC=INVALID	64	40
OBSQVAL	RPLOBSQV	—	XL2	STSN outbound sequence number	60	3C
OPTCD	RPLOPT1	RPLASY	X'08'	OPTCD=ASY	40	28
OPTCD	RPLOPT4	RPLPERS	X'80'	SETLOGON OPTCD=PERSIST	43	2B
		RPLNPERS	X'40'	SETLOGON OPTCD=NPERSIST	43	2B
		RPLINQPS	X'20'	INQUIRE OPTCD=PERSESS	43	2B
		RPLOPRES	X'10'	OPNDST OPTCD=RESTORE	43	2B
		RPLSLTMR	X'08'	PSTIMER specified on SETLOGON	43	2B
		RPLNQN	X'04'	INQUIRE OPTCD=NQN	43	2B
		RPLEXTOP	X'01'	RPLOPT4 byte holds an extended OPTCD value, see RPLOPT4X description for byte value meanings instead of other RPLOPT4 bit definitions.	43	2B
	RPLOPT4X			Extended OPTCD value byte		
		RPLSTGNA	X'01'	SETLOGON OPTCD=GNAMEADD	43	2B

Table 119. RPL DSECT (IFGRPL) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
		RPLSTGND	X'03'	SETLOGON OPTCD=GNAMEDEL	43	2B
		RPLSTGNS	X'05'	SETLOGON OPTCD=GNAMESUB	43	2B
		RPLIQSNM	X'21'	INQUIRE OPTCD=SESSNAME	43	2B
		RPLCGEAF	X'41'	CHANGE OPTCD=ENDAFFIN	43	2B
		RPLCGEF	X'43'	CHANGE OPTCD=ENDAFFNF	43	2B
		RPLSP_NoTimer _FTOALL	X'81'	SETLOGON OPTCD=PERSIST, (PARMS=(FORCETKO=ALL))	43	2B
		RPLSP_NoTimer _FTONONE	X'83'	SETLOGON OPTCD=PERSIST, (PARMS=(FORCETKO=NONE))	43	2B
		RPLSP_NoTimer _FTOSNGL	X'85'	SETLOGON OPTCD=PERSIST, PARMS=(FORCETKO=SINGLE))	43	2B
		RPLSP_NoTimer _FTOMULT	X'87'	SETLOGON OPTCD=PERSIST, PARMS=(FORCETKO=MULTI))	43	2B
		RPLSP_Timer _FTOALL	X'89'	SETLOGON OPTCD=PERSIST, PARMS=(FORCETKO=ALL, PSTIMER=value))	43	2B
		RPLSP_Timer _FTONONE	X'8B'	SETLOGON OPTCD=PERSIST, (PARMS=(FORCETKO=NONE, PSTIMER=value))	43	2B
		RPLSP_Timer _FTOSNGL	X'8D'	SETLOGON OPTCD=PERSIST, PARMS=(FORCETKO=SINGLE, PSTIMER=value))	43	2B
		RPLSP_Timer _FTOMULT	X'8F'	SETLOGON OPTCD=PERSIST, PARMS=(FORCETKO=MULTI, PSTIMER=value))	43	2B
	RPLOPT5	RPLDLGIN	X'80'	OPTCD=CS	56	38
		RPLTNFY	X'40'	PARMS=THRDPTY= NOTIFY	56	38
		RPLPSOPT	X'20'	OPTCD=PASS	56	38
		RPLNODE	X'02'	OPTCD=ANY	56	38
		RPLCNDCS	X'01'	OPTCD=CONDCS (used along with RPLDLGIN: RPLDLGIN = 1 and RPLCNDCS = 1 indicates Conditional CS mode, RPLDLGIN = 1 and RPLCNDCS = 0 indicates normal CS mode)	56	38
	RPLOPT6	RPLCOND	X'10'	OPTCD=COND	57	39
		RPLNCOND	X'08'	OPTCD=UNCOND	57	39
		RPLXBUFL	X'04'	OPTCD=XBUFLST	57	39
		RPLBUFFL	X'02'	OPTCD=BUFFLST	57	39
		RPLCONTC	X'01'	OPTCD=CONTCHN	57	39
	RPLOPT7	RPLCNALL	X'80'	OPTCD=CONALL	58	3A
		RPLCNANY	X'40'	OPTCD=CONANY	58	3A
		RPLQOPT	X'10'	OPTCD=Q	58	3A
		RPLRLSOP	X'04'	OPTCD=RELRQ	58	3A

Table 119. RPL DSECT (IFGRPL) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
		RPLLMPEO	X'01'	OPTCD=LMPEO	58	3A
	RPLOPT8	RPLODACQ	X'80'	OPTCD=ACQUIRE	59	3B
		RPLODACQ	X'40'	OPTCD=ACCEPT	59	3B
		RPLUSRRH	X'01'	OPTCD=USERRH	59	3B
	RPLOPT9	RPLLOGON	X'80'	OPTCD=LOGONMSG	96	60
		RPLDEVCH	X'40'	OPTCD=DEVCHAR	96	60
		RPLTERMS	X'20'	OPTCD=TERMS	96	60
		RPLCOUNT	X'10'	OPTCD=COUNTS	96	60
		RPLAPPST	X'08'	OPTCD=APPSTAT	96	60
		RPLINQST	X'04'	OPTCD=STATUS	96	60
		RPLCIDE	X'02'	OPTCD=CIDXLATE	96	60
		RPLTOPL	X'01'	OPTCD=TOPLOGON	96	60
	RPLOPT10	RPLSPARM	X'20'	OPTCD=SESSPARM	97	61
		RPLTSKY	X'10'	OPTCD=SESSKEY	97	61
		RPLUNBND	X'04'	OPTCD=UNBIND	97	61
		RPLSONOP	X'02'	OPTCD=SONCODE	97	61
		RPLSENOP	X'01'	OPTCD=SENSE	97	61
	RPLOPT11	RPLQUIES	X'80'	OPTCD=QUIESCE	98	62
		RPLSTART	X'40'	OPTCD=START	98	62
		RPLSTOP	X'20'	OPTCD=STOP	98	62
		RPLHOLD	X'10'	OPTCD=HOLD	98	62
		RPLMTS	X'04'	OPTCD=MTS	98	62
		RPLTERMQ	X'02'	OPTCD=TERMQ	98	62
		RPLKPSRB	X'01'	OPTCD=KEEPSRB	98	62
	RPLOPT12	RPLRSPQD	X'80'	OPTCD=RSPQUED	99	63
		RPLKEEP	X'40'	OPTCD=KEEP	99	63
		RPLTRUNC	X'20'	OPTCD=TRUNC	99	63
		RPLNIBTK	X'10'	OPTCD=NIBTK	99	63
		RPLQSESS	X'08'	OPTCD=QSESSLIM	99	63
		RPLQNOTE	X'04'	OPTCD=QNOTENAB	99	63
		RPLQALL	X'02'	OPTCD=QALL	99	63
		RPLFMHDR	X'01'	OPTCD=FMHDR	99	63
PARMS	RPLSONCD		XL1	PARMS=SONCODE=*code* UNBIND type code (Son code)	31	1F
POST	RPVTFL1	RPLRSPNM	X'02'	At least one response on normal flow inbound response queue	19	13
		RPLRSPQR	X'01'	At least one QRI response on normal flow inbound data queue.	19	13
POST	RPLVTFL2	RPLSCHED	X'80'	POST=SCHED	20	14

Table 119. RPL DSECT (IFGRPL) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
RECLEN	RPLRLEN	—	A	RECLEN value	48	30
REQ	RPLREQ	RPLQUISE	X'15'	SETLOGON	2	2
		RPLSMLGO	X'16'	SIMLOGON	2	2
		RPLOPNDS	X'17'	OPNDST	2	2
		RPLCHNG	X'19'	CHANGE	2	2
		RPLINQIR	X'1A'	INQUIRE	2	2
		RPLINTPT	X'1B'	INTRPRET	2	2
		RPLCLOSE	X'1F'	CLSDST	2	2
		RPLSNDCD	X'22'	SEND	2	2
		RPLRCVCD	X'23'	RECEIVE	2	2
		RPLRSRCD	X'24'	RESETSR	2	2
		RPLSSCCD	X'25'	SESSIONC	2	2
		RPLSDCMD	X'27'	SENDCMD	2	2
		RPLRVCMD	X'28'	RCVCMD	2	2
		RPLTREQS	X'29'	REQSESS	2	2
		RPLTOPNS	X'2A'	OPNSEC	2	2
		RPLTRMS	X'2C'	TERMSESS	2	2
		RPL6APPC	X'62'	APPCCMD	2	2
RESPOND	RPLVTFL2	RPLQRI	X'08'	RESPOND=QRESP	20	14
		RPLEX	X'04'	RESPOND=EX	20	14
		RPLNFME	X'02'	RESPOND=NFME	20	14
		RPLRRN	X'01'	RESPOND=RRN	20	14
RPLLEN	RPLLEN	—	XL1	RPL length	3	3
RTNCD	RPLRTNCD	—	XL1	RPL return code (see Appendix B, "Return codes and sense fields for RPL-based macroinstructions," on page 651 and ISTUSFBC DSECT in this appendix)	13	D
RTYPE	RPLSRTYP	RPLRRESP	X'08'	RTYPE=RESP	17	11
		RPLNFSYN	X'04'	RTYPE=NDFSYN	17	11
		RPLDFASY	X'02'	RTYPE=DFASY	17	11
SEQNO	RPLSEQNO	—	XL2	Sequence number	66	42
	RPLEXTDS	RPLXSRV	X'01'	Entire XBUFLST accepted by VTAM	68	44
SIGDATA	RPLSIGDA	—	XL4	Signal data	108	6C
SSENSEI	RPLSSEI	RPLPATHI	X'80'	SSENSEI=PATH	88	58
		RPLCPMI	X'40'	SSENSEI=CPM	88	58
		RPLSTATI	X'20'	SSENSEI=STATE	88	58
		RPLFII	X'10'	SSENSAI=FI	88	58
		RPLRRI	X'08'	SSENSEI=RR	88	58
SSENSEO	RPLSSEO	RPLCPMO	X'40'	SSENSEO=CPM	100	64

Table 119. RPL DSECT (IFGRPL) (continued)

Parameter on RPL-based macro	DSECT label	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Offset	
					Dec	Hex
		RPLSTATO	X'20'	SSENSEO=STATE	100	64
		RPLFIO	X'10'	SSENSEO=FI	100	64
		RPLRRO	X'08'	SSENSEO=RR	100	64
SSENSMI	RPLSSMI	—	XL1	System sense modifier input	89	59
SSENSMO	RPLSSMO	—	XL1	System sense modifier output	101	65
STYPE	RPLSRTYP	RPLSRESP	X'80'	STYPE=RESP	17	11
USENSEI	RPLUSNSI	—	XL2	User sense input	90	5A
USENSEO	RPLUSNSO	—	XL2	User sense output	102	66
USER	RPLUSFLD	—	XL4	User data field	92	5C
	RPLURH	—	XL3	User RH	28	1C
	RPLACTIV	—	X'FF'	RPL is active (cleared by CHECK macro)	69	45

RPL RTNCD-FDB2-FDBK (ISTUSFBC)

Table 120. RPL's RTNCD-FDB2-FDBK DSECT (ISTUSFBC)

Field to which byte values apply	DSECT RTNCD value	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Displacement in RPL	
					Dec	Hex
RTNCD (RPLRTNCD in IFGRPL)	—	USFAOK	X'00'	Normal or conditional completion	13	D
		USFXORDC	X'04'	Exception condition	13	D
		USFRESSU	X'08'	Retriable completion	13	D
		USFDAMGE	X'0C'	Data integrity damage	13	D
		USFENVER	X'10'	Environment error	13	D
		USFLOGIC	X'14'	Logic error	13	D
		USFRLGIC	X'18'	Logic error; RPL not valid	13	D
FDB2 (RPLFDB2 in IFGRPL)	X'00'	USFAOOK	X'00'	Normal completion or request accepted	14	E
		USFRCWNP	X'01'	Reset conditional was deactivated	14	E
		USFRCDPR	X'02'	Reset conditional was successful– Read-ahead data present	14	E
		USFYTCTN	X'03'	Yielded to contention	14	E
		USFYTCTL	X'04'	Yielded to contention–Error Lockset	14	E
		USFATSFI	X'05'	Input area too small	14	E
		USFNOIN	X'06'	No input available	14	E

Table 120. RPL's RTNCD-FDB2-FDBK DSECT (ISTUSFBC) (continued)

Field to which byte values apply	DSECT RTNCD value	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Displacement in RPL	
					Dec	Hex
		USFIIINA	X'07'	INQUIRE information not available	14	E
		USFDSTIU	X'08'	OPNDST OPTCD=ACQUIRE SIMLOGON, or CLSDST OPTCD=PASS failed	14	E
		USFNLGFA	X'09'	OPNDST OPTCD=ACCEPT denied–no queued CINITs or OPNDST OPTCD=RESTORE denied–no sessions restored	14	E
		USFANC	X'0A'	Application program not connectable	14	E
		USFINQPS	X'0D'	Input area too small	14	E
		USFEXRQ	X'03'	Exception request received	14	E
		USFEXRS	X'04'	Negative response received	14	E
		USFNQN	X'05'	Symbolic name known by network-qualified name only	14	E
	X'08'	USFSTALF	X'00'	Temporary storage shortage	14	E
	X'0C'	USFDVUNS	X'01'	Reserved	14	E
		USFUNTRM	X'02'	Reserved	14	E
		USFRECIP	X'07'	Reserved	14	E
		USFRTRAF	X'08'	Reserved	14	E
		USFQOPDC	X'09'	Reserved	14	E
		USFUSRES	X'0A'	Request canceled by RESETSR	14	E
		USFCLOCC	X'0B'	Request canceled because the session has been terminated	14	E
		USFCLRED	X'0C'	Request canceled by CLEAR request	14	E
		USFPREXC	X'0D'	Prior exception in chain detected	14	E
		USFPOQLE	X'0E'	Request cancelled - POA queue limit exceeded	14	E
	X'10'	USFTANAV	X'00'	LU not available, application program status not available, or queued — BIND not available	14	E
		USFSBFAL	X'01'	OPNDST failed	14	E
		USFTAPUA	X'02'	LU inhibited for sessions	14	E

Table 120. RPL's RTNCD-FDB2-FDBK DSECT (ISTUSFBC) (continued)

Field to which byte values apply	DSECT RTNCD value	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Displacement in RPL	
					Dec	Hex
		USFVTHAL	X'03'	HALT issued	14	E
		USFILRS	X'04'	Reserved	14	E
		USFPCF	X'05'	Request or Response encryption failure	14	E
		USFANS	X'06'	Reserved	14	E
		USFVOFOC	X'07'	Request canceled by VARY command	14	E
		USFUTSCR	X'09'	Unconditional Terminate or character-coded logoff received	14	E
		USFSYERR	X'0A'	VTAM error	14	E
		USFVTMNA	X'0D'	VTAM inactive for your ACB	14	E
		USFABNDO	X'0E'	Request abnormally terminated	14	E
		USFVTBFO	X'0F'	Buffers filled	14	E
		USFCTERM	X'10'	Reserved	14	E
		USFOSDTF	X'11'	SDT failure on OPNDST	14	E
		USFMFF	X'12'	Macroinstruction failure, sense included	14	E
		USF6APRJ	X'13'	Attempt to start 6.2 session; request rejected	14	E
		USA6APST	X'14'	Attempt to start 6.2 session; pending session terminated	14	E
		USF6APIS	X'15'	Must issue APPCCMD	14	E
		USFNONSW	X'16'	Switched operation attempted on non-switched device	14	E
		USFNOCRY	X'17'	Reject encryption request; session does not support cryptography	14	E
		USFNOSES	X'18'	XES not accessible	14	E
		USFNOSYS	X'19'	Application not resident in sysplex node	14	E
		USFXMEMS	X'1A'	Suspend failure	14	E
		USFXMEMR	X'1B'	Resume failure	14	E
		USFOSLVL	X'1C'	Operating system level does not support function	14	E
		USFSECME	X'1D'	Security manager error	14	E

Table 120. RPL's RTNCD-FDB2-FDBK DSECT (ISTUSFBC) (continued)

Field to which byte values apply	DSECT RTNCD value	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Displacement in RPL	
					Dec	Hex
	X'14'	USFNONVR	X'00'	VSAM request	14	E
		USFNOTAS	X'01'	Reserved	14	E
		USFEXTAZ	X'02'	Zero EXIT field	14	E
		USFEXTEZ	X'03'	Zero ECB field	14	E
		USFCRPLN	X'04'	Inactive RPL checked	14	E
		USFCBERR	X'10'	Control block not valid	14	E
		USFRNORT	X'11'	RTYPE not valid	14	E
		USFCLSIP	X'12'	CLSDST in progress	14	E
		USFCIDNG	X'13'	CID not valid	14	E
		USFIDA	X'1E'	Data address or length not valid	14	E
		USFJTOJ	X'20'	Reserved	14	E
		USFRILCP	X'22'	Reserved	14	E
		USFCRIRT	X'23'	Request type not valid	14	E
		USFASIDE	X'24'	Request not valid for address space	14	E
		USFIDAEL	X'31'	Reserved	14	E
		USFDFIBH	X'38'	Reserved	14	E
		USFQSCIE	X'3A'	Reserved	14	E
		USFREXAL	X'3B'	NFME-NRRN response	14	E
		USFSDNP	X'3C'	Previous macroinstruction outstanding	14	E
		USFSCEM	X'3D'	Reserved	14	E
		USFSCEF	X'3F'	Reserved	14	E
		USFSINVC	X'40'	CONTROL not valid	14	E
		USFSDFR	X'41'	Data traffic not allowed	14	E
		USFSNOS	X'42'	STYPE for SESSIONC not valid	14	E
		USFSNOUT	X'43'	Reserved	14	E
		USFLIMEX	X'44'	RESPLIM exceeded	14	E
		USFSSEQ	X'45'	Reserved	14	E
		USFSINVS	X'46'	Reserved	14	E
		USFSINVR	X'47'	3270 SEND option not valid	14	E
		USFINVRT	X'48'	Session control protocol violation	14	E
		USFACINV	X'49'	STSN action/result code not valid	14	E
		USFICNDN	X'4A'	Installation-wide exit routine not available	14	E

Table 120. RPL's RTNCD-FDB2-FDBK DSECT (ISTUSFBC) (continued)

Field to which byte values apply	DSECT RTNCD value	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Displacement in RPL	
					Dec	Hex
		USFILSIN	X'4B'	INTRPRET sequence or LOGMODE not valid, or cryptographic incompatibility	14	E
		USFIICBE	X'4C'	Search argument for INQUIRE or INTRPRET not valid	14	E
		USFINTNA	X'4D'	No interpret table	14	E
		USFILNBL	X'4E'	Use of an NIB list not valid	14	E
		USFINVOT	X'4F'	OPTCD setting not valid	14	E
		USFINVAP	X'50'	RPL field not valid	14	E
		USFAPNAC	X'51'	OPNDST OPTCD=ACCEPT and SIMLOGON not allowed	14	E
		USFINVNB	X'52'	NIB not valid	14	E
		USFSYMNU	X'53'	LU not found	14	E
		USFDSTUO	X'54'	Reserved	14	E
		USFNOPAU	X'55'	One of the following applies: • Application program not authorized • Application program name not available • Task association not specified • You must issue a SEND RPL.	14	E
		USFMDINC	X'56'	Reserved	14	E
		USFINVMD	X'57'	MODE field not valid	14	E
		USFBHSUN	X'58'	Reserved	14	E
		USFMDNAU	X'59'	Reserved	14	E
		USFMBHSS	X'5A'	Reserved	14	E
		USFINVLA	X'5B'	Reserved	14	E
		USFDUPND	X'5C'	Reserved	14	E
		USFNPSAU	X'5E'	CLSDST OPTCD=PASS not authorized	14	E
		USFRSCNO	X'5F'	Reserved	14	E
		USFRSCNC	X'60'	LU name for CLSDST, SESSIONC, or OPNSEC not valid	14	E
		USFINVSL	X'61'	SETLOGON not valid	14	E
		USFMCNVD	X'62'	Reserved	14	E

Table 120. RPL's RTNCD-FDB2-FDBK DSECT (ISTUSFBC) (continued)

Field to which byte values apply	DSECT RTNCD value	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Displacement in RPL	
					Dec	Hex
		USFRNOEL	X'6C'	Exceeded limit on outstanding RCVCMD requests	14	E
		USFRNONA	X'6D'	Application program not authorized	14	E
		USFRNOSE	X'6E'	Syntax error in reply to VTAM operator message	14	E
		USFRNOIA	X'6F'	SENDCMD/RCVCMD processor inactive	14	E
		USFRNOCL	X'70'	Program operator closing ACB with requests outstanding	14	E
		USFRNOCE	X'71'	Operator command not valid	14	E
		USFPCIT	X'72'	Reserved	14	E
		USFINVSD	X'73'	Send parameters not valid for CNM	14	E
		USFNRNBD	X'74'	Negotiable response to non-negotiable BIND		
		USFINBRP	X'75'	Negotiable BIND response parameters not valid	14	E
		USFINBSZ	X'76'	Negotiable BIND response size not valid	14	E
		USFNFMDQ	X'77'	FM data RU required	14	E
		USFCHINV	X'78'	Chain specification not valid	14	E
		USFBLINV	X'79'	Buffer list length not valid	14	E
		USFINVRH	X'7B'	User RH not valid	14	E
		USFSCINV	X'7C'	OPTCD=USERRH not valid for SESSIONC	14	E
		USFHPINV	X'7D'	XRF protocol error	14	E
		USFCOMR	X'7E'	Conflicting OPTCD on a macroinstruction request	14	E
		USFPRINV	X'80'	SETLOGON OPTCD=PERSIST and NPERSIST not valid	14	E
		USFTSPND	X'81'	TERMSESS without OPTCD=UNBIND with session in a pending state	14	E
		USFPARML	X'82'	Parameter length not valid	14	E
		USFSFERR	X'83'	Subfield or combination of subfields not valid	14	E
		USFASDAZ	X'84'	NIBASDPA = 0; address of the application's dial parameter list not valid	14	E

Table 120. RPL's RTNCD-FDB2-FDBK DSECT (ISTUSFBC) (continued)

Field to which byte values apply	DSECT RTNCD value	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Displacement in RPL	
					Dec	Hex
		USFSMBRS	X'85'	Session currently pending recovery	14	E
		USFSESSA	X'86'	Sessions active or affinities exist	14	E
		USFSNAME	X'87'	Resource name and generic name are same	14	E
		USFNOSPT	X'88'	No LU-to-application association meets the criteria for CHANGE OPTCD=ENDAFFIN or INQUIRE OPTCD=SESSNAME issued	14	E
		USFNSECM	X'89'	Not authorized for generic resources	14	E
		USFDIFNM	X'8A'	Already registered with different generic name	14	E
		USFNOMAP	X'8B'	Not registered as generic resource	14	E
		USFNETID	X'8C'	Already registered with different network ID	14	E
		USFCPNAM	X'8D'	Mapping exists already on other sysplex node	14	E
		USFCNFAC	X'8E'	Conflicting APPC capability	14	E
		USFVTAMO	X'8F'	Lu-to-application association owned by VTAM	14	E
		USFUSVAR	X'90'	Generic name conflict with existing USERVAR	14	E
		USFGNUNA	X'91'	TSO generic name conflict	14	E
		USFGGNMA	X'92'	SETLOGON GNAMESUB failure	14	E
		USFSTKNV	X'93'	STOKEN not valid	14	E
		USFNONAM	X'94'	No Telnet LU name was passed	14	E
		USFNORDT	X'95'	No applicable RDTE was found	14	E
		USFCNFNM	X'96'	Conflict with found RDTE	14	E
FDBK (RPLFDB3 in the ISTRPL DSECT)	X'00'	USFIACT	X'00'	APPL is active	15	F
		USFIINA	X'04'	APPL is inactive	15	F
		USFINA	X'08'	APPL ACB has MACRF=NLOGON	15	F

Table 120. RPL's RTNCD-FDB2-FDBK DSECT (ISTUSFBC) (continued)

Field to which byte values apply	DSECT RTNCD value	DSECT EQU label	Field or EQU value	For EQU: Meaning when bit setting on For DS: Meaning when byte value set	Displacement in RPL	
					Dec	Hex
		USFITNA	X'0C'	APPL has issued SETLOGON STOP	15	F
		USFIQUIE	X'10'	APPL has issued SETLOGON QUIESCE	15	F
		USFILACT	X'80'	LU is active	15	F
		USFILINA	X'84'	LU is not active	15	F

Access-method-support vector list (ISTAMSVL)

```
  Loc    Source Statement
000000 ISTAMSVL DSECT                      MAPPING FOR RESOURCE INFORMATION
       *                                   VECTOR LIST POINTED TO BY ACVAMSVL
000000 AMSLLEN  DS    HL2                  TOTAL LENGTH OF VECTORS
000002 AMSLDATA DS    0X                   VECTOR DATA
       *
       ************************************************************************
       ***  GENERALIZED MAPPING FOR EXAMINING COMMON FIELDS IN ALL ACB      **
       ***  VECTORS IN THE VECTOR LIST POINTED TO BY ACBAMSVL               **
       ************************************************************************
000000 ISTAMSVT DSECT                      VECTOR FIELDS
000000 AMSVTLEN DS    X                    VECTOR LENGTH
000001 AMSVTKEY DS    X                    VECTOR KEY
000002 AMSVTDAT DS    0X                   VECTOR DATA
       *
       *
       ************************************************************************
       *** ISTAMS01 - maps the RELEASE LEVEL vector.                        **
       ***          - Contains identification codes for the access method   **
       ***            product and its version, release, and modification    **
       ***            level.                                               **
       ************************************************************************
000000 ISTAMS01 DSECT                      RELEASE LEVEL VECTOR
000000 AMS01LEN DS    X                    VECTOR LENGTH
000001 AMS01KEY DS    X                    VECTOR KEY
       AMS01KYC EQU   X'01'                KEY IS X'01'
000002 AMS01DTA DS    0CL4                 VECTOR DATA
000002 AMS01PRD DS    CL1                  PRODUCT CODE
       AMS01VTM EQU   C'0'                 VTAM PRODUCT CODE
000003 AMS01VER DS    CL1                  VERSION CODE
000004 AMS01REL DS    CL1                  RELEASE CODE
000005 AMS01MDF DS    CL1                  MODIFICATION CODE
       *
       ************************************************************************
       *** ISTAMS04 - maps the COMPONENT IDENTIFICATION vector.             **
       ***          - This vector may be repeated.                         **
       ***          - Each component identification vector contains product **
       ***            identification information about a major component or **
       ***            feature of the VTAM licensed program.  When multiple  **
       ***            component identification vectors are present, the     **
       ***            first one designates the base VTAM product and later  **
       ***            vectors are features or other major VTAM components.  **
       ***          - The vector data is in the form: C'xxxx-xxxxx-xxx'.    **
       ************************************************************************
000000 ISTAMS04 DSECT                      COMPONENT IDENTIFICATION VECTOR
000000 AMS04LEN DS    X                    VECTOR LENGTH
000001 AMS04KEY DS    X                    VECTOR KEY
       AMS04KYC EQU   X'04'                KEY IS X'04'
```

```
000002 AMS04DTA DS    CL14                VECTOR DATA
       *
       ************************************************************************
       *** ISTAMS05 - maps the FUNCTION LIST vector.                        **
       ***          - The vector data is a variable-length bit string, in   **
       ***            which each bit corresponds to a particular VTAM       **
       ***            function.  If a bit is on, the corresponding function **
       ***            is present in the executing release of VTAM.  If a    **
       ***            bit is off, the function is not available.  If the    **
       ***            vector is not present or if the bit string is shorter **
       ***            than expected, you may assume that the missing bits   **
       ***            are zero and their corresponding functions are not    **
       ***            available.                                            **
       ***          - These indicator bits correspond to the compile-time   **
       ***            global indicator bits in the ISTGLBAL macro.          **
       ************************************************************************
000000 ISTAMS05 DSECT                      FUNCTION LIST VECTOR
000000 AMS05LEN DS    X                    VECTOR LENGTH
000001 AMS05KEY DS    X                    VECTOR KEY
       AMS05KYC EQU   X'05'                KEY IS X'05'
000002 AMS05DTA DS    0X                   VECTOR DATA
000002 AMS05DT0 DS    X                    BYTE 0 OF INDICATORS
       AMS05B00 EQU   X'80'                NIB ENCR AND RPL CRYPT
       *                                   (CRYPTOGRAPHY)
       AMS05B01 EQU   X'40'                ACB PARMS=NIB (COMMUNICATION
       *                                   NETWORK MANAGEMENT INTERFACE)
       AMS05B02 EQU   X'20'                MULTIPLE-ADDRESS-SPACE
       *                                   APPLICATIONS PROGRAMS
       AMS05B03 EQU   X'10'                AUTHORIZED PATH FOR
       *                                   COMMUNICATIONS MACROS
       AMS05B04 EQU   X'08'                AUTHORIZED PATH FOR ALL
       *                                   RPL-BASED MACROS
       AMS05B05 EQU   X'04'                SRBEXIT (ON APPL DEFINITION
       *                                   STATEMENT)
       AMS05B06 EQU   X'02'                SONSCIP (ON APPL DEFINITION
       *                                   STATEMENT)
       AMS05B07 EQU   X'01'                VTAMFRR (ON APPL DEFINITION
       *                                   STATEMENT)
       *
000003 AMS05DT1 DS    X                    BYTE 1 OF INDICATORS
       AMS05B10 EQU   X'80'                SSCP TRACKING OF DEVICE-LU
       *                                   SESSION CAPABILITY VIA NOTIFY
       *                                   (ENABLED/DISABLED/INHIBITED)
       AMS05B11 EQU   X'40'                RPL OPTCD=LMPEO
       AMS05B12 EQU   X'20'                RPL OPTCD=BUFFLST
       AMS05B13 EQU   X'10'                RPL OPTCD=USERRH
       AMS05B14 EQU   X'08'                ACB PARMS=USERFLD
       AMS05B15 EQU   X'04'                RPL BRACKET=CEB
       AMS05B16 EQU   X'02'                APPLICATION PROGRAM ASSIGNMENT OF
       *                                   SEQUENCE NUMBERS FOR EXPEDITED DFC
       AMS05B17 EQU   X'01'                RESOURCE-IDENTIFICATION VECTOR LIST
       *
000004 AMS05DT2 DS    X                    BYTE 2 OF INDICATORS
       AMS05B20 EQU   X'80'                ACCESS-METHOD-SUPPORT VECTOR LIST
       AMS05B21 EQU   X'40'                RETURN OF SYSTEM RESPONSE BYTE AND
       *                                   EXTENDED RESPONSE BYTE FOR BSC 3270
       *                                   TERMINALS ATTACHED TO ACF/NCP
       AMS05B22 EQU   X'20'                INTRPRET
       AMS05B23 EQU   X'10'                VTAM API IS XRF CAPABLE
       AMS05B24 EQU   X'08'                SENSE ON -RSP(CINIT). CLSDST
       *                                   OPTCD=(RELEASE,SENSE)
       AMS05B25 EQU   X'04'                UNBIND SON CODE AND SENSE.
       *                                   CLSDST OPTCD=(RELEASE,SONCODE),
       *                                   TERMSESS OPTCD=(UNBIND,SONCODE)
       AMS05B26 EQU   X'02'                HOLD/RELEASE LOGON/SCIP EXIT FOR
       *                                   SESSION SETUP.
       *                                   SETLOGON OPTCD=(START|HOLD)
```

```
       AMS05B27 EQU   X'01'               CINIT - NETWORK ADDRESSES IN
       *                                  VECTOR KEY X'15'
       *
000005 AMS05DT3 DS    X                   BYTE 3 OF INDICATORS
       AMS05B30 EQU   X'80'               31-BIT API
       AMS05B31 EQU   X'40'               NOTIFICATION OF QUEUED RESPONSES
       *                                  SUPPORTED.  SEND OPTCD=(RSPQUED)
       AMS05B32 EQU   X'20'               APPC IS SUPPORTED
       AMS05B33 EQU   X'10'               ADD SUPPORT FOR USERVAR
       AMS05B34 EQU   X'08'               VCNS API SUPPORT FOR X.25
       AMS05B35 EQU   X'04'               VCNS API SUPPORT FOR TOKEN BUS,
       *                                  TOKEN RING,
       AMS05B36 EQU   X'02'               CROSS-MEMORY API IS SUPPORTED
       AMS05B37 EQU   X'01'               KEEPFRR SUPPORT (ON ACB STATEMENT)
       *
000006 AMS05DT4 DS    X                   BYTE 4 OF INDICATORS
       AMS05B40 EQU   X'80'               SRBEXIT SUPPORT (ON ACB STATEMENT)
       AMS05B41 EQU   X'40'               PERSISTENT LU-LU SESSIONS
       AMS05B42 EQU   X'20'               V.25BIS SUPPORT
       AMS05B43 EQU   X'10'               VTAM/NPM INTERFACE SUPPORT
       AMS05B44 EQU   X'08'               LU6 PLUS TRACKING SUPPORTED
       AMS05B45 EQU   X'04'               EXITFRR SUPPORT (ON ACB STATEMENT)
       *                                                                 @N1C
       AMS05B46 EQU   X'02'               BYTE 4, BIT 6: RESERVED
       AMS05B47 EQU   X'01'               NETWORK QUALIFIED NAMES SUPPORTED
       *
000007 AMS05DT5 DS    X                   BYTE 5 OF INDICATORS
       AMS05B50 EQU   X'80'               MS TRANSPORT SUPPORTED
       AMS05B51 EQU   X'40'               PERFORMANCE MONITOR INTERFACE
       *                                  SUPPORTED
       AMS05B52 EQU   X'20'               QUEUED SESSION TERMINATION
       *                                  SUPPPORTED
       AMS05B53 EQU   X'10'               VTAM AGENT SUPPORTED
       AMS05B54 EQU   X'08'               GENERIC RESOURCES SUPPORTED
       AMS05B55 EQU   X'04'               OPTCD=KEEPSRB FOR SYNC SRB
       *                                  SUSPEND/RESUME
       AMS05B56 EQU   X'02'               APPLICATION VECTORS SUPPORTED ON
       *                                  ACB MACRO
       AMS05B57 EQU   X'01'               SETLOGON GNAMESUB SUPPORTED    @D2C
       *
000008 AMS05DT6 DS    X                   BYTE 6 OF INDICATORS
       AMS05B60 EQU   X'80'               FORCETKO Support (on ACB and the
       *                                  SETLOGON statements)
       AMS05B61 EQU   X'40'               USER CV Support
       AMS05B62 EQU   X'20'               BYTE 6, BIT 2: RESERVED
       AMS05B63 EQU   X'10'               BYTE 6, BIT 3: RESERVED
       AMS05B64 EQU   X'08'               BYTE 6, BIT 4: RESERVED
       AMS05B65 EQU   X'04'               BYTE 6, BIT 5: RESERVED
       AMS05B66 EQU   X'02'               BYTE 6, BIT 6: RESERVED
       AMS05B67 EQU   X'01'               BYTE 6, BIT 7: RESERVED
       *
000009 AMS05DT7 DS    X                   BYTE 7 OF INDICATORS
       AMS05B70 EQU   X'80'               BYTE 7, BIT 0: RESERVED
       AMS05B71 EQU   X'40'               BYTE 7, BIT 1: RESERVED
       AMS05B72 EQU   X'20'               BYTE 7, BIT 2: RESERVED
       AMS05B73 EQU   X'10'               BYTE 7, BIT 3: RESERVED
       AMS05B74 EQU   X'08'               BYTE 7, BIT 4: RESERVED
       AMS05B75 EQU   X'04'               BYTE 7, BIT 5: RESERVED
       AMS05B76 EQU   X'02'               BYTE 7, BIT 6: RESERVED
       AMS05B77 EQU   X'01'               BYTE 7, BIT 7: RESERVED
       *
       ************************************************************************
       *** ISTAMS06 - maps the LU6.2 SUPPORT FUNCTION LIST vector.          **
       ***          - The vector data is a variable-length string of byte   **
       ***            indicators, each byte corresponding to a particular   **
       ***            LU6.2 function.  Each byte will have a value showing  **
       ***            that its corresponding function is either supported,  **
```

```
       ***           not supported, or supported on a "pass-through" basis.**
       ***           (Pass-through functions are those which VTAM does not **
       ***           directly provide but provides the ability for the     **
       ***           application to create the support itself.)            **
       ***           If the vector is not present or if the byte string    **
       ***           is shorter than expected, you may assume that the     **
       ***           missing bytes are zero and their corresponding        **
       ***           functions are not supported.                          **
       ***         - These indicator bytes correspond to the compile-time  **
       ***           global indicators in the ISTGAPPC macro.              **
       ************************************************************************
000000 ISTAMS06 DSECT                    LU6.2 SUPPORT FUNCTION LIST VECTOR
000000 AMS06LEN DS    X                  VECTOR LENGTH
000001 AMS06KEY DS    X                  VECTOR KEY
       AMS06KYC EQU   X'06'              KEY IS X'06'
000002 AMS06DTA DS    0X                 VECTOR DATA
000002 AMS06D01 DS    X                  01. CONVERSATIONS BETWEEN TPS
       *                                     AT SAME LU
000003 AMS06D02 DS    X                  02. DELAYED SESSION
       *                                     ALLOCATION
000004 AMS06D03 DS    X                  03. IMMEDIATE SESSION
       *                                     ALLOCATION
000005 AMS06D04 DS    X                  04. SYNC POINT SERVICES
000006 AMS06D05 DS    X                  05. PROGRAM RECONNECT
000007 AMS06D06 DS    X                  06. RESERVED
000008 AMS06D07 DS    X                  07. SESSION-LEVEL LU-LU
       *                                     VERIFICATION
000009 AMS06D08 DS    X                  08. USERID VERIFICATION
00000A AMS06D09 DS    X                  09. PROGRAM SUPPLIED USERID
       *                                     AND PASSWORD
00000B AMS06D10 DS    X                  10. USERID AUTHORIZATION
00000C AMS06D11 DS    X                  11. PROFILE VERIFICATION AND
       *                                     AUTHORIZATION
00000D AMS06D12 DS    X                  12. RESERVED
00000E AMS06D13 DS    X                  13. PROFILE PASSTHROUGH
00000F AMS06D14 DS    X                  14. PROGRAM-SUPPLIED PROFILE
000010 AMS06D15 DS    X                  15. SEND PERSISTENT
       *                                     VERIFICATION
000011 AMS06D16 DS    X                  16. RECEIVE PERSISTENT
       *                                     VERIFICATION
000012 AMS06D17 DS    X                  17. PIP DATA
000013 AMS06D18 DS    X                  18. LOGGING OF DATA IN SYSTEM
       *                                     LOG
000014 AMS06D19 DS    X                  19. FLUSH LU SEND BUFFER
000015 AMS06D20 DS    X                  20. LUW IDENTIFIER
000016 AMS06D21 DS    X                  21. PREPARE TO RECEIVE
000017 AMS06D22 DS    X                  22. LONG LOCKS
000018 AMS06D23 DS    X                  23. POST ON RECEIPT WITH WAIT
000019 AMS06D24 DS    X                  24. POST ON RECEIPT WITH TEST
       *                                     FOR POSTING
00001A AMS06D25 DS    X                  25. RECEIVE-IMMEDIATE
00001B AMS06D26 DS    X                  26. TEST FOR REQUEST-TO-SEND
       *                                     RECEIVED
00001C AMS06D27 DS    X                  27. DATA MAPPING
00001D AMS06D28 DS    X                  28. FMH APPLICATION-DATA
00001E AMS06D29 DS    X                  29. GET ATTRIBUTES
00001F AMS06D30 DS    X                  30. GET CONVERSATION-TYPE
000020 AMS06D31 DS    X                  31. MAPPED CONVERSATION LU
       *                                     SERVICES COMPONENT
000021 AMS06D32 DS    X                  32. CHANGE_SESSION_LIMIT VERB
000022 AMS06D33 DS    X                  33. MIN_CONWINNERS_TARGET
       *                                     PARAMETER
000023 AMS06D34 DS    X                  34. RESPONSIBLE(TARGET)
       *                                     PARAMETER
000024 AMS06D35 DS    X                  35. DRAIN_TARGET(NO) PARAMETER
000025 AMS06D36 DS    X                  36. FORCE PARAMETER
000026 AMS06D37 DS    X                  37. ACTIVATE_SESSION VERB
```

```
000027 AMS06D38 DS    X                  38. DEACTIVATE_SESSION VERB
000028 AMS06D39 DS    X                  39. LU-PARAMETER VERBS
000029 AMS06D40 DS    X                  40. LU-LU SESSION LIMIT
00002A AMS06D41 DS    X                  41. LOCALLY-KNOWN LU NAMES
00002B AMS06D42 DS    X                  42. UNINTERPRETED LU NAMES
00002C AMS06D43 DS    X                  43. SINGLE-SESSION
       *                                     RE-INITIATION
00002D AMS06D44 DS    X                  44. ALTERNATE CODE PROCESSING
00002E AMS06D45 DS    X                  45. MAXIMUM RU SIZE BOUNDS
00002F AMS06D46 DS    X                  46. SESSION-LEVEL MANDATORY
       *                                     CRYPTOGRAPHY
000030 AMS06D47 DS    X                  47. CONTENTION WINNER
       *                                     AUTOMATIC ACTIVATION LIMIT
000031 AMS06D48 DS    X                  48. CONWINNER SESSION
       *                                     ALLOCATION
000032 AMS06D49 DS    X                  49. ENHANCED SECURITY (SAME)
000033 AMS06D50 DS    X                  50. SESSION-LEVEL SELECTIVE
       *                                     CRYPTOGRAPHY
000034 AMS06D51 DS    X                  51. CONVERSATION GROUP SUPPORT
000035 AMS06D52 DS    X                  52. ALLOCATE WHEN SESSION FREE
000036 AMS06D53 DS    X                  53. FULL-DUPLEX
000037 AMS06D54 DS    X                  54. APPCCMD VECTOR LISTS
000038 AMS06D55 DS    X                  55. QUEUED RCVFMH5              @D1A
000039 AMS06D56 DS    X                  56. HIGH PERFORMANCE DATA
       *                                     TRANSFER                  @D3A
00003A AMS06D57 DS    X                  57. APPCCMD SENDRCV             @D1A
00003B AMS06D58 DS    X                  58. INTRA-LU CONVERSATIONS      @D4A
00003C AMS06D59 DS    X                  59. PASSWORD SUBSTITUTION       @D5A
00003D AMS06D60 DS    X                  60. EXTENDED SECURITY SENSE     @D5A
00003E AMS06D61 DS    X                  61. DCE SECURITY SERVICES       @D6A
```

Resource-information vector list (ISTRIVL)

```
  Loc    Source Statement
000000 ISTRIVL  DSECT                    MAPPING FOR RESOURCE INFORMATION
       *                                 VECTOR LIST POINTED TO BY ACBRIVL
000000 RIVLLEN  DS    HL2                TOTAL LENGTH OF VECTORS
000002 RIVLDATA DS    0X                 VECTOR DATA
       *
       ************************************************************************
       ***  GENERALIZED MAPPING FOR EXAMINING COMMON FIELDS IN ALL ACB      **
       ***  VECTORS IN THE VECTOR LIST POINTED TO BY ACBRIVL                **
       ************************************************************************
000000 ISTRIVVT DSECT                    VECTOR TEMPLATE                 @Y3A
000000 RIVVTLEN DS    X                  VECTOR LENGTH                   @Y3A
000001 RIVVTKEY DS    X                  VECTOR KEY                      @Y3A
000002 RIVVTDAT DS    0X                 VECTOR DATA                     @Y3A
       *
       *
       ************************************************************************
       *** ISTRIV02 - maps the application's network name vector.           **
       ***          - The name is specified by the name field of the        **
       ***            application definition statement.                     **
       ***          - This is obtained from the NAME ON APPL STATEMENT.     **
       ************************************************************************
000000 ISTRIV02 DSECT                    APPLICATION NETWORK NAME VECTOR
       *                                 (FROM NAME ON APPL STATEMENT)
000000 RIV02LEN DS    X                  VECTOR LENGTH
000001 RIV02KEY DS    X                  VECTOR KEY
       RIV02KYC EQU   X'02'              KEY IS X'02'
000002 RIV02DTA DS    CL8                VECTOR DATA
       *
       ************************************************************************
       *** ISTRIV03 - maps the application's ACB name vector.               **
       ***          - This is supplied by the APPLID operand on the ACB     **
       ***            statement or can be supplied by the operating         **
       ***            system.  During OPEN ACB, VTAM will search for the    **
```

```
       ***            application's characteristics by matching the ACB     **
       ***            APPLID value to an RDTE with the application's        **
       ***            ACBNAME.  If ACBNAME was not coded for the            **
       ***            application, VTAM will search for a match with an     **
       ***            RDTE containing the application's network name.       **
       ***          - This is obtained from the APPLID on ACB MACRO.        **
       ************************************************************************
000000 ISTRIV03 DSECT                      APPLICATION ACB NAME VECTOR
       *                                   (FROM APPLID ON ACB MACRO)
000000 RIV03LEN DS    X                    VECTOR LENGTH
000001 RIV03KEY DS    X                    VECTOR KEY
       RIV03KYC EQU   X'03'                KEY IS X'03'
000002 RIV03DTA DS    CL8                  VECTOR DATA
       *
       ************************************************************************
       *** ISTRIV06 - maps the network name in which the host resides.      **
       ***          - This is obtained from the NETID START OPTION.         **
       ***            If NETID start option is not specified, this value    **
       ***            will be blanks.                                       **
       ************************************************************************
000000 ISTRIV06 DSECT                      NETWORK NAME VECTOR
       *                                   (FROM NETID START OPTION)
000000 RIV06LEN DS    X                    VECTOR LENGTH
000001 RIV06KEY DS    X                    VECTOR KEY
       RIV06KYC EQU   X'06'                KEY IS X'06'
000002 RIV06DTA DS    CL8                  VECTOR DATA
       *
       ************************************************************************
       *** ISTRIV07 - maps the SSCP Name vector.                           **
       ***          - This is obtained from the SSCPNAME START OPTION       **
       ************************************************************************
000000 ISTRIV07 DSECT                      SSCP NAME VECTOR
       *                                   (FROM SSCPNAME START OPTON)
000000 RIV07LEN DS    X                    VECTOR LENGTH
000001 RIV07KEY DS    X                    VECTOR KEY
       RIV07KYC EQU   X'07'                KEY IS X'07'
000002 RIV07DTA DS    CL8                  VECTOR DATA
       *                                   (DEFAULT IS 'VTAM')
       *
       ************************************************************************
       *** ISTRIV08 - maps the Host Subarea PU Network Name vector.         **
       ***          - This is obtained from the HOSTPU START OPTION         **
       ***            If HOSTPU start option is not specified, the name     **
       ***            will default to 'ISTPUS  '.                           **
       ************************************************************************
000000 ISTRIV08 DSECT                      HOST SUBAREA PU NETWORK NAME VECTOR
       *                                   (FROM HOSTPU START OPTION)
000000 RIV08LEN DS    X                    VECTOR LENGTH
000001 RIV08KEY DS    X                    VECTOR KEY
       RIV08KYC EQU   X'08'                KEY IS X'08'
000002 RIV08DTA DS    CL8                  VECTOR DATA
       *                                   (DEFAULT IS 'ISTPUS')
       *
       ************************************************************************
       *** ISTRIV09 - maps the Host Subarea PU network address vector.      **
       ***          - It contains the network address of the host           **
       ***            subarea PU.                                           **
       ************************************************************************
000000 ISTRIV09 DSECT                      HOST SUBAREA PU NETWORK ADDRESS
       *
000000 RIV09LEN DS    X                    VECTOR LENGTH
000001 RIV09KEY DS    X                    VECTOR KEY
       RIV09KYC EQU   X'09'                KEY IS X'09'
000002 RIV09DTA DS    XL6                  VECTOR DATA
       *
       ************************************************************************
       *** ISTRIV0A - maps the maximum subarea vector.                      **
```

```
       ***          - Contains the maximum subarea number that is valid     **
       ***            for the host's domain.                                **
       ***          - This is obtained from the MAXSUBA START OPTION        **
       ************************************************************************
000000 ISTRIV0A DSECT                      MAXIMUM SUBAREA NUMBER VECTOR
       *                                   (FROM MAXSUBA START OPTION)
000000 RIV0ALEN DS    X                    VECTOR LENGTH
000001 RIV0AKEY DS    X                    VECTOR KEY
       RIV0AKYC EQU   X'0A'                KEY IS X'0A'
000002 RIV0ADTA DS    X                    VECTOR DATA
       *
       ************************************************************************
       *** ISTRIV0B - maps the LU 6.2 application definition vector.        **
       ***            After the LU 6.2 application program has issued an    **
       ***            open ACB, the LU 6.2 application program may use      **
       ***            this vector to determine the values coded on the      **
       ***            APPL definition statement.                            **
       ***          - This is obtained from the APPL STATEMENT PARAMETERS   **
       ************************************************************************
000000 ISTRIV0B DSECT                      LU 6.2 APPL DEFINITION VECTOR
       *                                   (FROM APPL STATEMENT PARAMETERS)
000000 RIV0BLEN DS    X                    VECTOR LENGTH
000001 RIV0BKEY DS    X                    VECTOR KEY
       RIV0BKYC EQU   X'0B'                KEY IS X'0B'
000002 RIV0BDTA DS    0X                   VECTOR DATA
000002          DS    X                    RESERVED
       RIV0BSLV EQU   X'C0'                SESSION-LEVEL LU-LU VERIFICATION
       *                                   BIT MASK
       RIV0BSLR EQU   X'80'                REQUIRED
       RIV0BSLO EQU   X'40'                OPTIONAL
       RIV0BSLN EQU   X'00'                NONE
000003 RIV0BCLS DS    X                    CONVERSATION SECURITY ACCEPTANCE
       RIV0BCLN EQU   X'01'                NONE
       RIV0BCLC EQU   X'02'                CONV
       RIV0BCLA EQU   X'03'                ALREADYV
       RIV0BCLP EQU   X'04'                PERSISTV
       RIV0BCLV EQU   X'05'                AVPV
000004 RIV0BFLG DS    X                    MISCELLANEOUS FLAGS
       RIV0BDDL EQU   X'80'                DDRAINL=ALLOW
       RIV0BDRL EQU   X'40'                DRESPL=ALLOW
       RIV0BATA EQU   X'20'                ATNLOSS=ALL
       RIV0BSYP EQU   X'10'                SYNCLVL=SYNCPT
       RIV0BOPC EQU   X'08'                OPERCNOS=ALLOW
000005          DS    X                    RESERVED
000006 RIV0BDSL DS    HL2                  DSESLIM VALUE
000008 RIV0BDML DS    HL2                  DMINWNL VALUE
00000A RIV0BDMR DS    HL2                  DMINWNR VALUE
00000C RIV0BAUT DS    HL2                  AUTOSES VALUE
       *
       ************************************************************************
       *** ISTRIV0C - maps the common application definition vector.        **
       ***            After the application program has issued an open for  **
       ***            its ACB, the application may examine this vector to   **
       ***            determine the values coded on the APPL definition     **
       ***            statement for common application defination keywords. **
       ***          - This is obtained from the APPL STATEMENT PARAMETERS   **
       ************************************************************************
000000 ISTRIV0C DSECT                      APPLICATION DEFINITION VECTOR
       *                                   FOR ALL APPLICATION PROGRAMS   @N1A
       *                                   (FROM APPL STATEMENT PARAMETERS)
000000 RIV0CLEN DS    X                    VECTOR LENGTH                  @N1A
000001 RIV0CKEY DS    X                    VECTOR KEY                     @N1A
       RIV0CKYC EQU   X'0C'                KEY IS X'0C'                   @N1A
000002 RIV0CDTA DS    0X                   VECTOR DATA                    @N1A
000002 RIV0CAUT DS    X                    AUTHORIZATION SETTINGS         @N1A
       RIV0CACQ EQU   X'80'                AUTH=ACQ                       @N1A
       RIV0CASD EQU   X'40'                AUTH=ASDP                      @N1A
```

```
       RIV0CCNM EQU   X'20'               AUTH=CNM                        @N1A
       RIV0CPAS EQU   X'10'               AUTH=PASS                       @N1A
       RIV0CPPO EQU   X'08'               AUTH=PPO                        @N1A
       RIV0CSPO EQU   X'04'               AUTH=SPO                        @N1A
       RIV0CTSO EQU   X'02'               AUTH=TSO                        @N1A
       RIV0CVPA EQU   X'01'               AUTH=VPACE                      @N1A
000003 RIV0CFL1 DS    X                   MISCELLANEOUS FLAGS 1           @N1A
       RIV0CAPC EQU   X'80'               APPC=YES                        @N1A
       RIV0CAUX EQU   X'40'               AUTHEXIT=YES                    @N1A
       RIV0CCER EQU   X'20'               CERTIFY=YES                     @N1A
       RIV0CDSW EQU   X'10'               DSPLYWLD=YES                    @N1A
       RIV0CFSP EQU   X'08'               FASTPASS=YES                    @N1A
       RIV0CHAV EQU   X'04'               HAVAIL=YES                      @N1A
       RIV0CPAR EQU   X'02'               PARSESS=YES                     @N1A
       RIV0CPRS EQU   X'01'               PERSIST=MULTI                   @N1A
000004 RIV0CFL2 DS    X                   MISCELLANEOUS FLAGS 2           @N1A
       RIV0CSSL EQU   X'80'               SESSLIM=YES                     @N1A
       RIV0CSON EQU   X'40'               SONSCIP=YES                     @N1A
       RIV0CSRX EQU   X'20'               SRBEXIT=YES                     @N1A
       RIV0CVCN EQU   X'10'               VCNS=YES                        @N1A
       RIV0CVFR EQU   X'08'               VTAMFRR=YES                     @N1A
000005 RIV0CLTM DS    X                   LOSTERM SETTING                 @N1A
       RIV0CLTN EQU   X'00'               LOSTERM=NORMAL                  @N1A
       RIV0CLTI EQU   X'01'               LOSTERM=IMMED                   @N1A
       RIV0CLTS EQU   X'02'               LOSTERM=SECOND                  @N1A
000006 RIV0CCMI DS    X                   CMPAPPLI VALUE                  @N1A
000007 RIV0CCMO DS    X                   CMPAPPLO VALUE                  @N1A
000008 RIV0CENC DS    X                   ENCR VALUE                      @N1A
       RIV0CECN EQU   X'00'               ENCR=NONE                       @N1A
       RIV0CECO EQU   X'01'               ENCR=OPT                        @N1A
       RIV0CECC EQU   X'02'               ENCR=COND                       @N1A
       RIV0CECS EQU   X'03'               ENCR=SEL                        @N1A
       RIV0CECR EQU   X'04'               ENCR=REQD                       @N1A
000009 RIV0CVPC DS    X                   VPACING VALUE                   @N1A
00000A          DS    XL4                 RESERVED                        @N1A
       *
       ********************************************************************
       *** ISTRIV11 - maps the APPCCMD vector area length vector.          **
       ***          - It contains the absolute minimum length and the      **
       ***            recommended minimum length for full use of the       **
       ***            APPCCMD vector area.                                 **
       ********************************************************************
000000 ISTRIV11 DSECT                     APPCCMD VECTOR AREA LENGTH VECTOR
       *                                                                 @L3C
000000 RIV11LEN DS    X                   VECTOR LENGTH
000001 RIV11KEY DS    X                   VECTOR KEY
       RIV11KYC EQU   X'11'               KEY IS X'11'
000002 RIV11AML DS    XL4                 ABSOLUTE MINIMUM APPCCMD VECTOR
       *                                  AREA LENGTH                  @L3A
000006 RIV11RML DS    XL4                 RECOMMENDED MINIMUM APPCCMD
       *                                  VECTOR AREA LENGTH           @L3C
       *
       ********************************************************************
       *** ISTRIV12 - maps the application to VTAM vector keys vector.     **
       ***          - It contains a list of all ACB vector keys that       **
       ***            VTAM will process. Constants for the ACB vectors are **
       ***            located in ISTVACBV.                                 **
       ********************************************************************
000000 ISTRIV12 DSECT                     APPLICATION TO VTAM VECTOR KEYS
       *                                  FOR ACB MACRO
000000 RIV12LEN DS    X                   VECTOR LENGTH
000001 RIV12KEY DS    X                   VECTOR KEY
       RIV12KYC EQU   X'12'               KEY IS X'12'
000002 RIV12DTA DS    0CL1                VECTOR DATA
       *
       ********************************************************************
       *** ISTRIV13 - maps the Performance Monitor vector.                 **
```

```
      ***           Identifies a table of Performance Data vector fields  **
      ***           (within ISTXPL) that have been retired by the         **
      ***           Performance Monitor Interface since its inception.    **
      ************************************************************************
000000 ISTRIV13 DSECT                     PERFORMANCE MONITOR VECTOR   @L1A
      *
000000 RIV13LEN DS    X                   VECTOR LENGTH                @L1A
000001 RIV13KEY DS    X                   VECTOR KEY                   @L1A
      RIV13KYC EQU   X'13'               KEY IS X'13'                 @L1A
000002 RIV13ENT DS    HL2                 NUMBER OF ENTRIES IN TABLE   @L2A
      *                                   (ZERO IF NONE RETIRED)
000004 RIV13RFT DS    AL4                 RETIRED FIELDS TABLE ADDRESS @L2A
      *                                   (ZERO IF NONE RETIRED)
000008 RIV13ELN DS    HL2                 LENGTH OF EACH ENTRY         @L2A
      *
000000 RIV13TBL DSECT                     RETIRED FIELDS TABLE ENTRY   @L1A
      *                                   (MAPS ENTRIES IN TABLE ADDRESSED
      *                                   BY RIV13RFT)
000000 RIV13VID DS    0CL6                ID OF AFFECTED VECTOR        @L1A
000000 RIV13MAJ DS    CL2                 MAJOR CATEGORY               @L1A
000002 RIV13SUB DS    CL2                 SUBCATEGORY                  @L1A
000004 RIV13REC DS    CL2                 RECORD TYPE                  @L1A
000006 RIV13FLD DS    0CL4                FIELD POSITION WITHIN VECTOR @L1A
000006 RIV13OFF DS    HL2                 FIELD OFFSET                 @L1A
000008 RIV13FLG DS    BL1                 FLAG BYTE                    @L1A
      RIV13BIT EQU   X'01'               DATA TYPE INDICATOR
      *                                   (1= BITSTRING, 0= OTHER)     @L1A
000009 RIV13LNG DS    XL1                 FIELD LENGTH IF NOT BITSTRING,
      *                                   MASK FOR BITS RETIRED WITHIN BYTE
      *                                   FOR BITSTRING FIELD
      *
```

Application-ACB vector list (ISTVACBV)

```
  Loc    Source Statement
      ************************************************************************
      ***                                                                  **
      ***       DATA FIELDS PASSED FROM THE APPLICATION TO VTAM.           **
      ***                                                                  **
      ***                                                                  **
      *** Addressability: ACBAPID, ACBPASSW.                              **
      ***                                                                  **
      ************************************************************************
000000 ISTVACAP  DSECT                    APPLID MAPPING
      *
000000 VACAPLEN DS    X                   MAP LENGTH
000001 VACAPDTA DS    0X                  MAP DATA
      *
000000 ISTVACPW DSECT                     PASSWORD MAPPING
      *
000000 VACPWLEN DS    X                   MAP LENGTH
000001 VACPWDTA DS    0X                  MAP DATA
      *
      ************************************************************************
      ***                                                                  **
      ***          VECTORS PASSED FROM THE APPLICATION TO VTAM.           **
      ***                                                                  **
      ***                                                                  **
      *** Addressability: ACBAVPTR.                                       **
      ***                                                                  **
      *** Note: Highorder bit in vector key is on for all vectors sent     **
      ***       from application to VTAM.                                  **
      ***                                                                  **
      ************************************************************************
      *
      ***     MAPPING FOR VECTORLIST HEADER (LENGTH FIELD)                 **
000000 ISTVACAV DSECT                     APPLICATION VECTORLIST
```

```
       *                                POINTED TO BY ACBAPVTR
       *                                WHEN PARMS=(APPLVCTR=address)
000000 VACAVLEN DS    HL2               TOTAL LENGTH OF APPL VECTORS
000002 VACAVDTA DS    0X                VECTOR DATA
       *
       ********************************************************************
       ***  GENERALIZED MAPPING FOR EXAMINING OR BUILDING COMMON FIELDS IN  **
       ***  ALL ACB VECTORS IN THE VECTOR LIST POINTED TO BY ACBAVPTR       **
       ********************************************************************
000000 ISTVACVT DSECT                  VECTOR TEMPLATE
000000 VACVTLEN DS    HL2              VECTOR LENGTH
000002 VACVTKEY DS    X                VECTOR KEY
000003 VACVTDAT DS    0X               VECTOR DATA
       *
       *
       ********************************************************************
       *** ISTVAC81 - Application Capabilities vector                     **
       ***          - Passed to VTAM by the application at OPEN invocation  **
       ***            for the ACB.                                        **
       ***          - Bit indicators which enable/disable application use   **
       ***            of certain VTAM functions.                          **
       ********************************************************************
000000 ISTVAC81 DSECT                  APPLICATION CAPABILITIES VECTOR
000000 VAC81LEN DS    HL2              VECTOR LENGTH
000002 VAC81KEY DS    X                VECTOR KEY
       VAC81KYC EQU   X'81'            KEY IS X'81'
000003 VAC81CAP DS    0XL4             APPLICATION CAPABILITIES DATA
       VAC81MLE EQU   X'80'            APPLICATION SUPPORTS HAVING ITS
       *                               LOGON EXIT DRIVEN MULTIPLE TIMES
       *                               PER SESSION REQUEST.  APPLICATIONS
       *                               WITH LOGON EXITS MUST SET THIS
       *                               INDICATOR TO BENEFIT FROM
       *                               VERIFICATION REDUCTION
       VAC81FPR EQU   X'40'            APPLICATION INDICATES THAT IT WILL
       *                               USE HPDT INTERFACE PROVIDED
       *                               VIA THE OPTCD=XBUFLST FIELD ON THE
       *                               APPCCMD RECEIVE MACROINSTRUCTION
       *                                                              @D1A
       VAC81PWS EQU   X'20'            APPLICATION INDICATES THAT IT
       *                               IS PASSWORD SUBSTITUTION
       *                               CAPABLE                  @D2A
       VAC81ESS EQU   X'10'            APPLICATION INDICATES THAT IT
       *                               IS CAPABLE OF EXTENDED
       *                               SECURITY SENSE CODES     @D2A
       VAC81FPS EQU   X'08'            APPLICATION INDICATES THAT IT
       *                               WILL USE HPDT INTERFACE
       *                               PROVIDED BY THE OPTCD=XBUFLST
       *                               FIELD ON AN APPCCMD
       *                               MACROINSTRUCTION THAT SENDS
       *                               DATA                     @D1A
       VAC81EOM EQU   X'04'            APPLICATION INDICATES THAT IT
       *                               IS CAPABLE TO SUPPORT END OF
       *                               MESSAGES MESSAGE FOR DISPLAY,
       *                               MODIFY AND VARY COMMANDS     @J1A
       VAC81ACO EQU   X'02'            APPLICATION INDICATES THAT IT
       *                               REQUESTS AUTOSES FOR CNOS
       *                               ONLY
       VAC81FAA EQU   X'01'            APPLICATION INDICATES THAT IT
       *                               REQUESTS ATNLOSS=ALL
       VAC81UCV EQU   X'0080'          APPLICATION INDICATES THAT IT
       *                               WOULD LIKE TO SPECIFY A USER
       *                               CONTROL VECTOR ON SETLOGON START
       *
       *
       ********************************************************************
       *** ISTVAC82 - Local Application's DCE Capability Vector          **
       ***          - Passed to VTAM by the application at OPEN invocation  **
```

```
       ***            for the ACB.                                          **
       ***          - Contains the Security Mechanisms data for the Local   **
       ***            LU.                                                   **
       ************************************************************@D3A**
000000 ISTVAC82 DSECT                     LOCAL APPLICATION'S DCE
       *                                  CAPABILITY VECTOR MAPPING    @D3A
000000 VAC82LEN DS    HL2                 LENGTH OF VECTOR (INCLUDING
       *                                  LENGTH OF THIS FIELD).       @D3A
000002 VAC82KEY DS    X                   VECTOR KEY                   @D3A
       VAC82KYC EQU   X'82'               VECTOR KEY X'82'             @D3A
000003 VAC82DTA DS    0X                  ISTVAC82 DATA                @D3A
```

Appendix F. Specifying a session parameter

A BIND request and response contain parameters that specify certain protocols used by the PLU and SLU for the session. This appendix describes the format of the session parameter as seen by a VTAM application program. The macroinstructions used to establish a session and specify session parameters are described in "Establishing parameters for sessions" on page 122. For a detailed description of the session parameters themselves and how to use them, see:

- Chapter 5, "Establishing and terminating sessions with logical units," on page 81, as well as the
- *SNA Format and Protocol Reference Manual: Architectural Logic, SNA Formats, SNA Sessions between Logical Units,* and
- *SNA Concepts and Products*

VTAM supplies a default logon mode table containing examples of session parameters for certain devices. This table is described in detail in the z/OS Communications Server: SNA Resource Definition Reference.

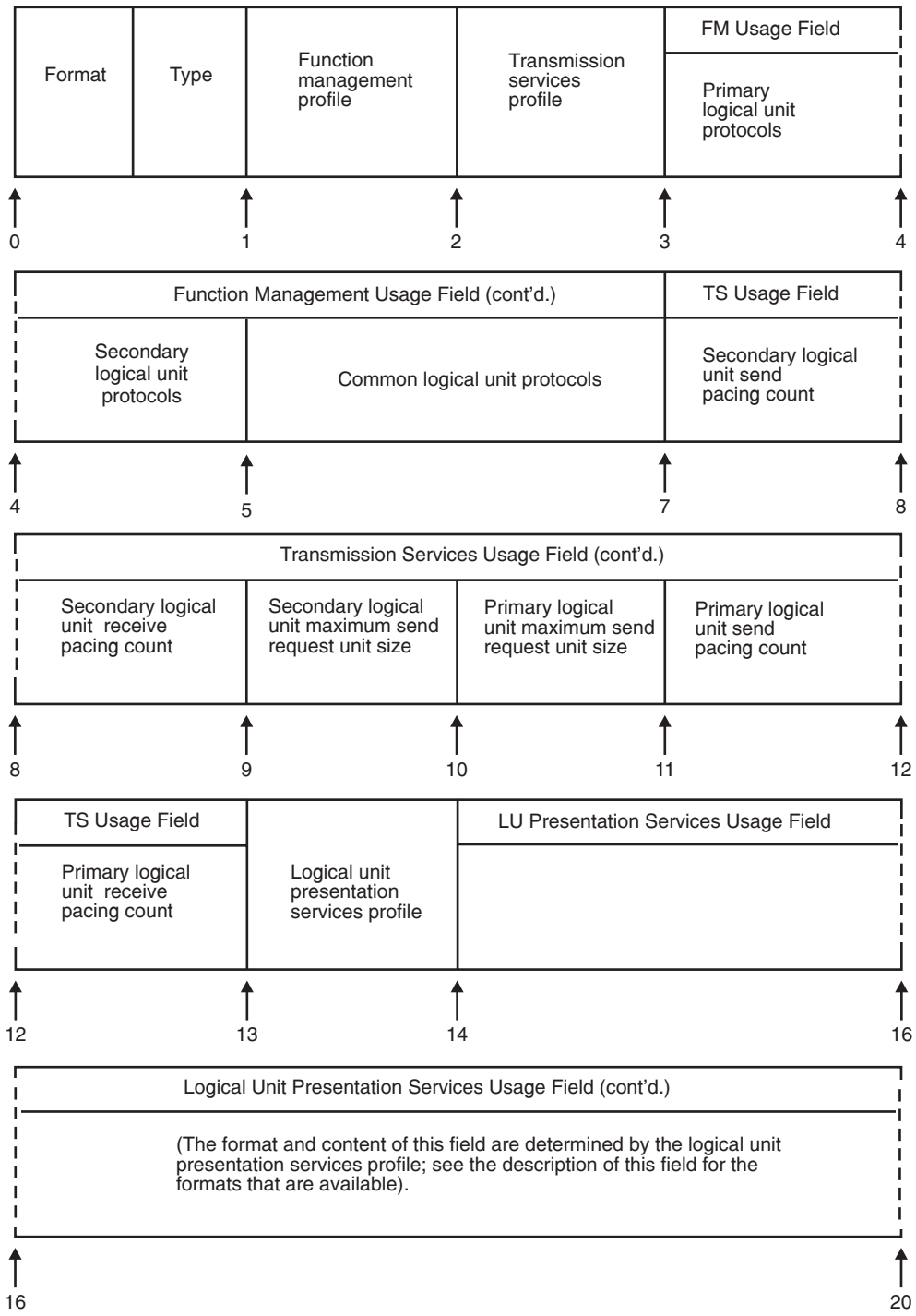
Note: This appendix describes fields applicable to session parameters and is not to be used to locate fields in an SNA BIND request. Because the SNA BIND request is created by VTAM and not the application program, the BIND request contains fields set by VTAM that have no significance to the application program. Also, corresponding fields in the session parameters and the BIND request can have different formats and be located in different positions.

Session parameter fields (BIND image)

The displacements indicated in Figure 171 on page 794 refer to the ISTDBIND DSECT. The displacements are 1 greater in the actual BIND RU, because the BIND RU contains a request code that is not contained in the ISTDBIND DSECT. Note that the BIND image field in the CINIT RU also does not contain the BIND request code.

The format of the session parameter fields is shown in Figure 171 on page 794. A description of each field follows.

© Copyright IBM Corp. 2000, 2013



Note: The dashed vertical line indicates continuation of the field to the next line.

Figure 171. Format of session parameters area (BIND image)

Note: When session parameters are part of the BIND or CINIT RU, the primary-logical-unit-name length field can have a value ranging 1–8, thus changing the displacements of the user-data-length and user-data fields. Similarly, for these RUs, the cryptographic-control field can be a variable-length field and thus alter succeeding displacements. For session parameters received in INQUIRE OPTCD=SESSPARM or in the SCIP exit routine (pointed to by word 4 of the exit

parameter list), the primary-logical-unit name field is always 8 bytes long. The associated length field is set to 8 (or 0). Also, in these cases, the cryptographic-control field is always 1 byte long. Therefore, the displacements shown in this figure are correct.

Although the PLU name can be 1-17 bytes in the BIND RU, the ISTDBIND DSECT requires it to be exactly 8 bytes. This causes the offset of the user data field to vary in the BIND RU but not in the BIND DSECT. The area provided by VTAM to the SCIP exit will have an 8 in the PLU name length field and the name may be padded with blanks or truncated, if necessary. VTAM requires the bind area to be in this format when it is supplied by the application on an OPNDST or OPNSEC macroinstruction. Also note that in the case of a BIND error, the offset contained in sense code 0835*nnnn* is adjusted, if necessary, when given to the application to parse the BIND DSECT. See "BIND area format and DSECT" on page 816 for information about these session parameters.

Format

This field specifies the format of the BIND request or response. Only format 0 is supported.

Type

This field indicates whether the BIND request or response is negotiable (type 0) or non-negotiable (type 1). See "BIND request" on page 91 for further information on negotiable or non-negotiable BIND requests.

Function management profile

The function management profile identifies a predefined set of protocol rules. Each profile represents a different set of rules. Within each profile, some of the protocol rules are mandatory and some are optional. The profiles that have optional rules require the use of the function management usage field to indicate how the optional rules are implemented. For a detailed description of the function management profiles, see *SNA Formats* The following profiles are available to the VTAM application program and can be specified in the function management profile field (using the FMPROF operand of the MODEENT macroinstruction).

Bit setting 0123 4567
(Byte 1) Meaning

0000 0010
Function management profile 2 is to be used. (This profile applies only to LU type 0 3270 terminals.)

0000 0011
Function management profile 3 is to be used.

0000 0100
Function management profile 4 is to be used.

0000 0111
Function management profile 7 is to be used.

Transmission services profile

The transmission services profile identifies a predefined set of session-control requests that can be used in a session. Each profile represents a different set of session-control requests and other transmission services attributes. For a detailed description of the transmission services profiles, see *SNA Formats* The following

profiles are available to the VTAM application program and can be specified in the transmission services field (using the TSPROF operand of the MODEENT macroinstruction).

Bit setting 0123 4567
(Byte 2) Meaning

0000 0010
Transmission services profile 2 is to be used. (This profile applies only to LU type 0 3270 sessions.)

0000 0011
Transmission services profile 3 is to be used.

0000 0100
Transmission services profile 4 is to be used.

0000 0111
Transmission services profile 7 is to be used.

The session-control requests that apply to each of these profiles are shown in Table 121.

In this table:

Req This request must be used. The required requests can be issued by VTAM on behalf of the application program.

Opt This request can be used.

N/A Not applicable. This request must not be used.

Table 121. Session-control requests for each transmission services profile

Session- control request	Profile 2	Profile 3	Profile 4	Profile 7
CLEAR	Opt	Opt	Opt	N/A
RQR	N/A	N/A	Opt	N/A
SDT	N/A	Req	Req	N/A
STSN	N/A	N/A	Opt	N/A

The function management usage field is used to supplement the protocol rules defined by the function management profile. The function management usage field is divided into three parts:

- The primary logical unit protocols (specified with the PRIPROT operand of the MODEENT macroinstruction) apply to the primary end of the session.
- The secondary logical unit protocols (specified with the SECPROT operand of the MODEENT macroinstruction) apply to the secondary end of the session.
- The common logical unit protocols (specified with the COMPROT operand of the MODEENT macroinstruction) apply to both the primary and secondary ends of the session.

The 2 bytes describing the primary and secondary logical unit protocols have the same format; however, different bits can be on or off depending upon the protocol used and how it applies to the specific end of the session being described by the byte.

The format of the function management usage field is described in *SNA Technical Overview* and is also shown in Table 124 on page 821 (see the primary, secondary, and common protocol fields). An overview of the use of many of the indicators in this field is given in Chapter 6, "Communicating with logical units," on page 151. More detailed information about the function management usage field can be found in *SNA Sessions Between Logical Units* and *SNA Concepts and Products*

Transmission services usage field

The Transmission Services Usage (TSU) field indicates the send pacing count and maximum send request unit size for both the primary and secondary logical unit. See Figure 171 on page 794 for a description of this field.

Request unit size

The session parameters specify the maximum length of request units that can be sent by the logical units. This information is divided into two parts: the secondary logical unit send request unit size (byte 9) and the primary logical unit send request unit size (byte 10). Both bytes have the same format; however, they can be set to different values depending upon the requirements of the logical units. The bits that can be set (using the RUSIZES operand of the MODEENT macroinstruction) follow:

Bit setting 0123 4567	**(Bytes 9 and 10) Meaning**
`0... ....`	(byte 9) 6K bytes is the default maximum length of the request unit that can be sent by the SLU. VTAM will terminate a session on behalf of the PLU if a larger RU is received.
`0... ....`	(byte 10) There is no limit specified for the size of the request unit that can be sent by the PLU.
`1000 ....` through `1111 ....`	This is the mantissa (m) of the formula m X 2^n used to determine the maximum length of request units that can be sent by the primary or the secondary logical unit.
`.... 0000` through `.... 1111`	This is the exponent (n) of the formula m X 2^n used to determine the maximum length of request units that can be sent by the primary or secondary logical unit.

For example, if byte 9 or 10 is set to hex 85, the mantissa is 8 and the exponent is 5; the associated RU size is 256 bytes (8×2^5). For a discussion of maximum RU size when using SEND OPTCD=LMPEO, refer to "LMPEO operating considerations" on page 183.

The maximum size of request units is shown in Table 122.

Table 122. Maximum size of request unit (in decimal)

Decimal value of bits 4–7	**Decimal value of bits 0–3**							
	8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	8	9	10	11	12	13	14	15
1	16	18	20	22	24	26	28	30
2	32	36	40	44	48	52	56	60
3	64	72	80	88	96	104	112	120
4	128	144	160	176	192	208	224	240

Table 122. Maximum size of request unit (in decimal) (continued)

Decimal value of bits 4–7	Decimal value of bits 0–3							
	8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
5	256	288	320	352	384	416	448	480
6	512	576	640	704	768	832	896	960
7	1024	1152	1280	1408	1536	1664	1792	1920
8	2048	2304	2560	2816	3072	3328	3584	3840
9	4096	4608	5120	5632	6144	6656	7168	7680
10	8192	9216	10240	11264	12288	13312	14336	15360
11	16384	18432	20480	22528	24576	26624	28672	30720
12	32768	36864	40960	45056	49152	53248	57344	61440
13	65536	73728	81920	90112	98304	106496	114688	122880
14	131072	147456	163840	180224	196608	212992	229376	245760
15	262144	294912	327680	360488	393216	425984	458752	491520

Note: The bits this table refers to are contained in the RU size bytes (bytes 9 and 10).

Pacing count

The pacing count fields permit a user to control the rate of data flow through the network path joining a VTAM application program and a logical unit (including another VTAM application program). The four types of pacing count information are:

- The secondary logical unit send pacing count (byte 7) specified with the SSNDPAC operand of the MODEENT macroinstruction.
- The secondary logical unit receive pacing count (byte 8) specified with the SRCVPAC operand of the MODEENT macroinstruction.
- The primary logical unit send pacing count (byte 11) specified with the PSNDPAC operand of the MODEENT macroinstruction.
- The primary logical unit receive pacing count (byte 12).

Bit setting 0123 4567	**(Byte 7) Meaning**
`0... ....`	Bit off = one-stage pacing.
`1... ....`	Bit on = two-stage pacing.
`.x.. ....`	Reserved.
`..00 0000 through ..11 1111`	Secondary send pacing count.

Bit setting 0123 4567	**(Byte 8) Meaning**
`1... ....`	Adaptive pacing.
`.x.. ....`	Reserved.
`xx.. ....`	Reserved.
`..00 0000 through ..11 1111`	Secondary receive pacing count.

Bit setting 0123 4567	**(Byte 11) Meaning**
`0... ....`	Bit off = two-stage pacing.
`1... ....`	Bit on = one-stage pacing.
`.x.. ....`	Reserved.
`..00 0000 through ..11 1111`	Primary send pacing count.

Bit setting 0123 4567	**(Byte 12) Meaning**
`xx.. ....`	Reserved.

Bit setting 0123 4567	(Byte 12) Meaning
..00 0000 through ..11 1111	Primary receive pacing count.

Logical unit presentation services profile

The logical unit presentation services profile identifies a predefined type of logical unit and the format of the associated logical unit presentation services usage field. Each type of logical unit uses a unique subset of the SNA-defined protocols and data streams for its operation. The logical unit presentation services usage field is used in conjunction with the presentation services profile field. The profile field determines the format of the usage field; the usage field is used to specify optional protocols or data streams that are applicable to the LU type specified in the profile. For detailed information on LU types and on LU presentation services usage field indicators, see *SNA Sessions Between Logical Units* and *SNA Concepts and Products* The logical unit presentation services profiles that can be specified (using the PSERVIC operand of the MODEENT macroinstruction) follow:

Bit setting 0123 4567
(Byte 13) Meaning

0000 0000
Logical unit presentation service profile 0 is used (LU type 0).

0000 0001
Logical unit presentation service profile 1 is used (LU type 1).

0000 0010
Logical unit presentation service profile 2 is used (LU type 2).

0000 0011
Logical unit presentation service profile 3 is used (LU type 3).

0000 0100
Logical unit presentation service profile 4 is used (LU type 4).

0000 0110
Logical unit presentation service profile 6 is used (LU type 6).

The subsets of protocols and data streams that apply to each of these logical unit types are:

LU type 0

Use of a set of protocols and data streams agreed upon by the logical units involved. SNA does not define them. Note that byte 24 is an exception to this rule.

LU type 1

- Use of predefined function management headers to identify a variety of data set types and controls that can be sent and received.
- Use of an SNA character string (SCS) and a defined set of SNA protocols for a keyboard/printer device.

LU type 2

Use of a 3270 data stream and a defined set of SNA protocols for a keyboard/display device.

LU type 3

Use of a 3270 data stream and a defined set of SNA protocols for a printer.

LU type 4

Use of an SNA character string (SCS) and a defined set of SNA protocols for a batch input/output LU having associated with it a printer and a magnetic strip reader/recorder.

LU type 6 for LU 6.2

Use of a set of protocols and data streams agreed upon by the logical units involved which is characterized by:

- A peer relationship between session partners
- Efficient utilization of a session for multiple transactions
- Comprehensive end-to-end error processing.

Logical unit presentation services usage field

The logical unit presentation services usage field specifies (using the PSERVIC operand of the MODEENT macroinstruction) certain optional SNA protocols and data streams for the type of logical unit identified by the logical unit profile field. See the description of the logical unit presentation services profile field in the preceding section for an explanation of the profiles. The format of the logical unit presentation services usage field that corresponds to each profile is as follows:

Profile 0

Figure 172 shows the profile 0 presentation services usage field.

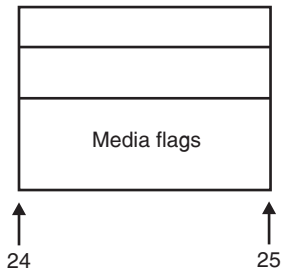


Figure 172. Profile 0 presentation services usage field

Media flags

The bit settings for byte 24 are:

Bit setting 0123 4567
(Byte 24) Meaning

xxxx xx..
Reserved.

.... ..00
No compression

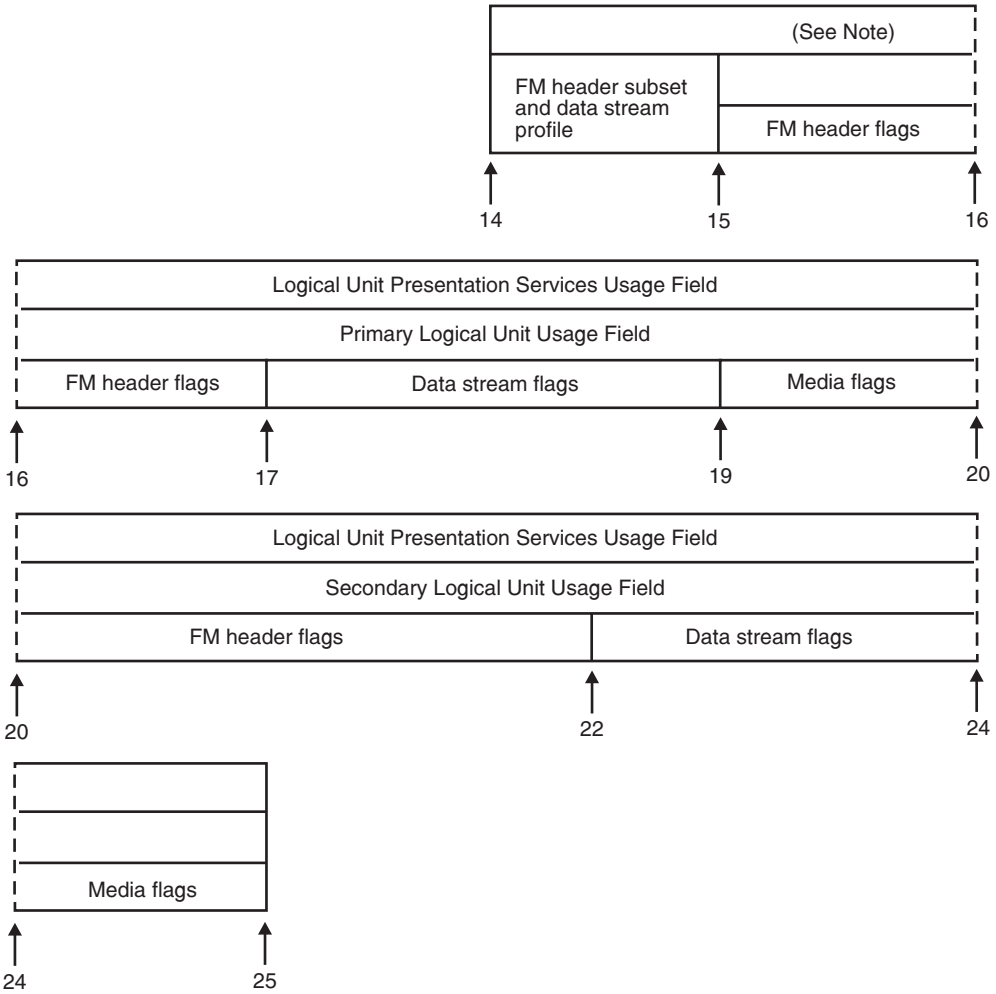
.... ..01
Compression bid

.... ..10
Reserved

.... ..11
Compression required

Profile 1

Figure 173 shows the profile 1 presentation services usage field.



Note: The dashed vertical line indicates continuation of the field to the next line.

Figure 173. Profile 1 presentation services usage field

Function management header subset and data stream profile

This field identifies a predefined set of function management header and data stream subsets. Each subset specifies the types of data sets and data set controls and the types of data streams that can be handled. The following subset profiles are available:

Bit setting 0123 4567
(Byte 14) Meaning

0000 ....
Function management header subset 0 is used.

0001 ....
Function management header subset 1 is used.

0010 ....
Function management header subset 2 is used.

0011 ....
Function management header subset 3 is used.

.... 0000
Data stream subset 0 is used.

.... 0001
Data stream subset 1 is used.

Function management header subset flags

These flags identify the types of data sets and data set controls that can be used. They are specified for the primary logical unit (bytes 15 and 16) and for the secondary logical unit (bytes 20 and 21). Both sets of bytes have the same format; however, different bits can be set for the primary and secondary logical units. If the function management header usage bit in the common protocols area of the function management usage field is set to 0, these flags cannot be used and should be set to 0. The following function management header subset flags can be set:

Bit setting 0123 4567
(Bytes 15 or 20) Meaning

Function management header subset 0

xxxx xxxx
Reserved.

Function management header subset 1, 2, or 3

1... ....
The transmission can be interrupted twice to send to another logical unit (three destinations are outstanding).

0... ....
The transmission can be interrupted once to send to another logical unit (two destinations are outstanding).

.1.. ....
The primary and the secondary LUs can send compacted data.

.0.. ....
The primary and the secondary LUs must not send compacted data.

..1. ....
The primary and the secondary LUs can send PDIR.

..0. ....
The primary and the secondary LUs must not send PDIR.

...x xxxx
Reserved (for subsets 1 and 2 only).

Function management header subset 3

...1 ....
The primary or the secondary LU can send keyed direct data sets.

...0 ....
The primary or the secondary LU does not send keyed direct data sets.

.... 1...
The primary or the secondary LU can send sequential data sets.

.... 0...
The primary or the secondary LU does not send sequential data sets.

.... .1..
The primary or the secondary LU can send addressed direct data sets that are accessed sequentially with Add, Note, and Note Reply in the opposite direction.

.... .0..
The primary or the secondary LU does not send addressed direct data sets that are accessed sequentially with Add, Note, and Note Reply in the opposite direction.

.... ..1.
The primary or the secondary LU supports series IDs (with status in reply).

.... ..0.
The primary or the secondary LU does not support series IDs (with status in reply).

....1
The primary or the secondary LU supports Add Replicate and Replace Replicate.

....0
The primary or the secondary LU does not support Add Replicate and Replace Replicate.

Bit setting 0123 4567
(Bytes 16 or 21) Meaning

Function management header subset 0, 1, or 2

.... 1...
The primary or the secondary LU can use structured fields for error recovery processing.

.... 0...
The primary or the secondary LU does not use structured fields for error recovery processing.

....1
The primary or the secondary LU supports query of structured fields.

....0
The primary or the secondary LU does not support query of structured fields.

xxxx .xx.
Reserved.

Function management header subset 3

.1.. ....
The primary or the secondary LU supports query for data sets.

.0.. ....
The primary or the secondary LU does not support query for data sets.

..1. ....
The primary or the secondary LU supports Create, Scratch, and Scratch All data sets.

..0. ....
The primary or the secondary LU does not support Create, Scratch, and Scratch All data sets.

...1 ....
The primary or the secondary LU supports Execute Program Offline.

...0 ....
The primary or the secondary LU does not support Execute Program Offline.

.... 1...
The primary or the secondary LU can use structured fields for error recovery processing.

.... 0...
The primary or the secondary LU does not use structured fields for error recovery processing.

....1
The primary or the secondary LU supports query of structured fields.

....0
The primary or the secondary LU does not support query of structured fields.

x... .xx.
Reserved.

Data stream subset flags

These flags identify the type of data stream that can be used. They are specified for the primary logical unit (bytes 17 and 18) and for the secondary logical unit (bytes 22 and 23). Both bytes have the same format; however, different bits can be set for the primary and secondary logical units. The following data stream subset flags can be set:

Bit setting 0123 4567
(Bytes 17 or 22) Meaning

1... ....
The primary or the secondary LU can send an interactive data stream (BS, CR, LF, HT, VT, ENP, and INF).

0... ....
The primary or the secondary LU does not send an interactive data stream (BS, CR, LF, HT, VT, ENP, and INP).

.1.. ....
The primary or the secondary LU can send a horizontal format data stream (SHF) with parameters.

.0.. ....
The primary or the secondary LU does not send a horizontal format data stream (SHF).

..1. ....
The primary or the secondary LU can send a vertical format data stream (SVF) with parameters.

..0. ....
The primary or the secondary LU does not send a vertical format data stream (SVF).

...1 ....
The primary or the secondary LU can send vertical channel, including vertical format and vertical channel select (VCS) with parameters.

...0 ....
The primary or the secondary LU does not send vertical channel, including vertical format and vertical channel select (VCS) with parameters.

.... 1...
SLD (Set Line Density) is supported.

.... 0...
SLD (Set Line Density) is not supported.

.... ..1.
The primary or the secondary LU can send BEL.

.... ..0.
The primary or the secondary LU does not send BEL.

....1
The primary or the secondary LU can send a transparency data stream (TRN and IRS).

....0
The primary or the secondary LU does not send a transparency data stream (TRN and IRS).

.... .x..
Reserved.

Bit setting 0123 4567
(Bytes 18 or 23) Meaning

1... ....
The secondary LU initiates unattended (byte 18 only).

0... ....
The secondary LU initiates attended (byte 18 only).

.1.. ....
During the session, the secondary LU can alternate between attended and unattended (byte 18 only).

.0.. ....
During the session, the secondary LU does not alternate between attended and unattended (byte 18 only).

..xx xxxx
Reserved (byte 18 only).

xxxx xxxx
Reserved (byte 23 only).

Media flags

These flags identify the types of physical recording media for which the data stream can be formatted. They are specified for the primary logical unit (byte 19) and for the secondary logical unit (byte 24). Both bytes have the same format; however, different bits can be set for the primary and secondary logical units. The following media flags can be set:

Bit setting 0123 4567
(Bytes 19 or 24) Meaning

1... ....
The primary or the secondary LU can use document output.

0... ....
The primary or the secondary LU does not use document output.

.1.. ....
The primary or the secondary LU can use card format.

.0.. ....
The primary or the secondary LU does not use card format.

..1. ....
The primary or the secondary LU can use exchange media format.

..0. ....
The primary or the secondary LU does not use exchange media format.

...1 ....
The primary or the secondary LU can use disk data management format.

...0 ....
The primary or the secondary LU does not use disk data management format.

.... 1...
The primary or secondary LU can use extended card format.

.... 0...
The primary or secondary LU does not use extended card format.

.... .1..
The primary or secondary LU can use extended document format.

.... .0..
The primary or secondary LU does not use extended document format.

.... ..1.
(Byte 19) The secondary LU must send CD every EDS.

.... ..0.
(Byte 19) The secondary LU can send CD every EDS.

....x
(Byte 19) Reserved.

.... ..00
(Byte 24) No compression

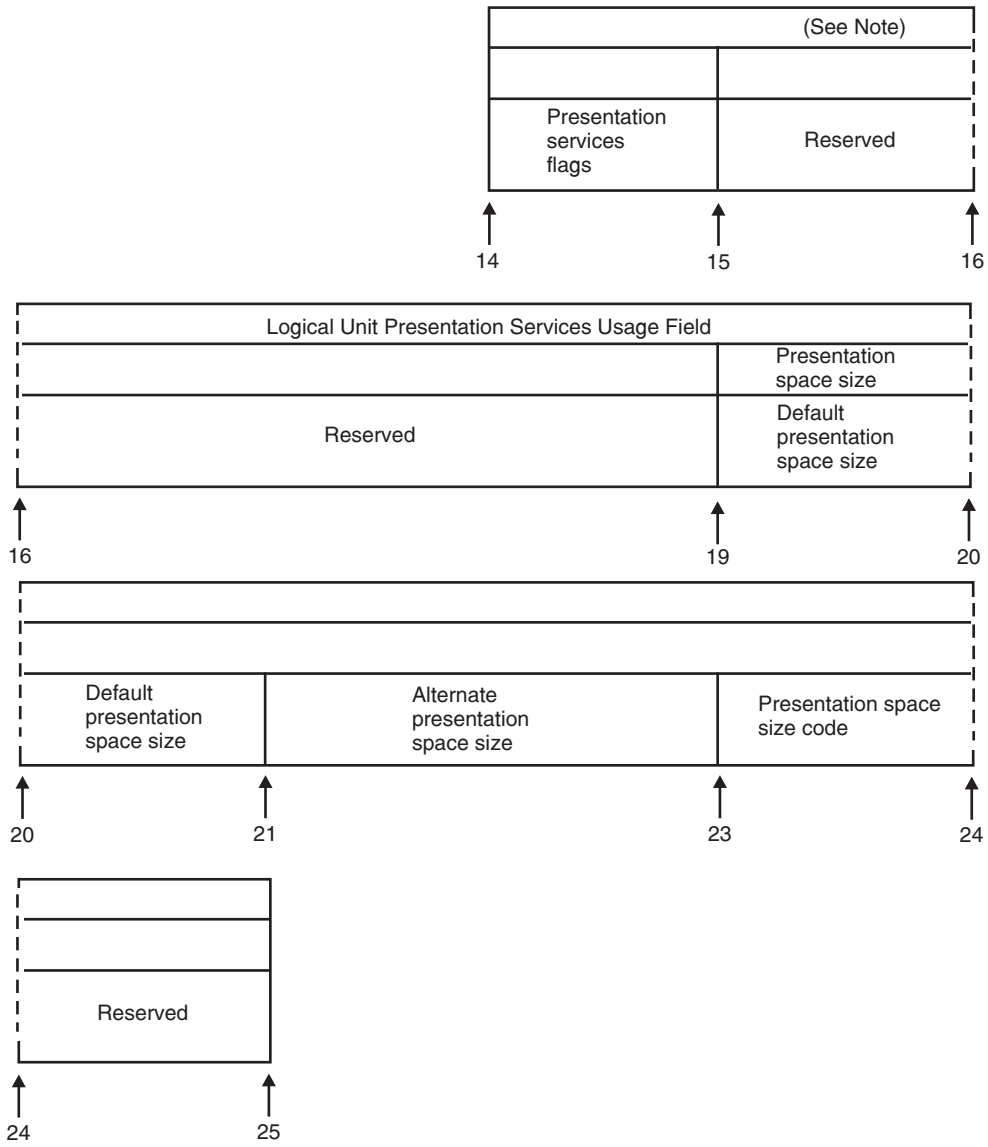
.... ..01
(Byte 24) Compression bid

.... ..10
(Byte 24) Reserved

.... ..11
(Byte 24) Compression required

Profiles 2 and 3

Figure 174 on page 807 shows the profile 2 and 3 presentation services usage fields.



Note: The dashed vertical line indicates continuation of the field to the next line.

Figure 174. Profile 2 and 3 presentation services usage fields

Presentation services flags

Bit setting 0123 4567
(Byte 14) Meaning

0... ....
Query (for example, for a 3270 extended data stream) is not supported.

1... ....
Query is supported.

.xxx xxxx
Reserved.

Presentation space size

This field (bytes 19–23) contains the default and alternate presentation space sizes and a presentation space size code field. These fields are shown in Figure 174 on page 807.

Bytes 19 and 21 contain the number of rows in the default presentation space and the alternate presentation space respectively. Bytes 20 and 22 contain the number of columns in the default presentation space and the alternate presentation space respectively.

Byte 23 identifies the presentation space size or indicates to use either the default or alternate presentation space size. The sizes that can be specified are:

Bit setting 0123 4567
(Byte 23) Meaning

0000 0000
A presentation space is not defined.

0000 0001
A presentation space of 12 by 40 bytes is used.

0000 0010
A presentation space of 24 by 80 bytes is used.

0111 1110
A presentation space as described by the default field is used.

0000 0011
The default presentation space is 24 by 80 bytes, and the alternate presentation space is specified in the Query Reply (implicit partition size) structured field. If there is no implicit partition size specified, use usable area.

0111 1111
Either the default or the alternate presentation space size is used.

Bit setting 0123 4567
(Byte 24) Meaning

xxxx xx..
Reserved

.... ..00
No compression

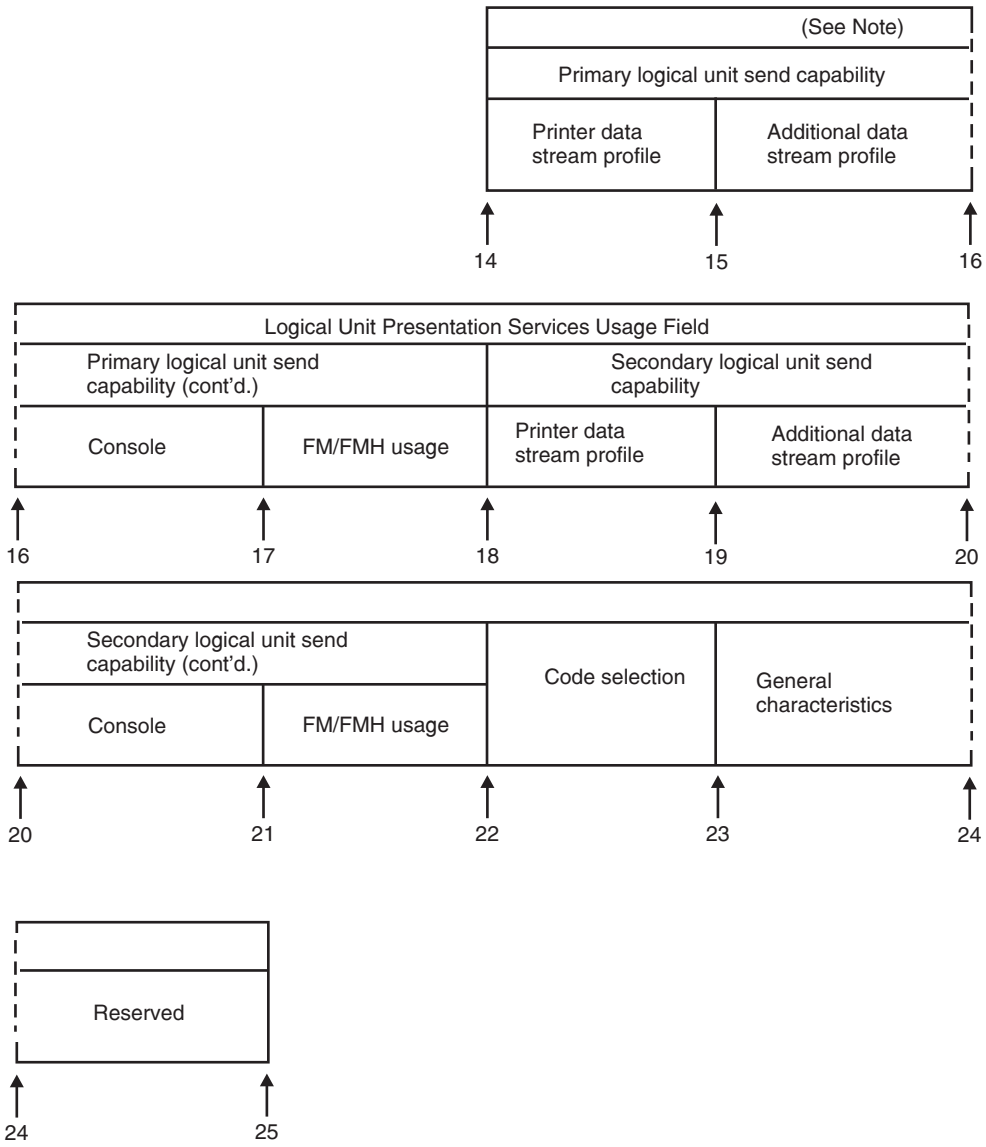
.... ..01
Compression bid

.... ..10
Reserved

.... ..11
Compression required

Profile 4

Figure 175 on page 809 shows the profile 4 presentation services usage field.



Note: The dashed vertical line indicates continuation of the field to the next line.

Figure 175. Profile 4 presentation services usage field

Logical unit send capability

These 4-byte fields identify a predefined set of protocols used by either the primary logical unit (bytes 14–17) or the secondary logical unit (bytes 18–21) to send data through the VTAM network. See Figure 175. These send protocols are:

Bit setting 0123 4567
(Bytes 14 or 18) Meaning

Printer data stream profile

1... ....
Base data stream profile is supported.

0... ....
Base data stream profile is not supported.

.1.. ....
General data stream is supported.

.0.. ....
General data stream is not supported.

..1. ....
Job SNA character string (SCS) subset is supported.

..0. ....
Job SNA character string (SCS) is not supported.

.... 1...
Word processing in raw form is supported.

.... 0...
Word processing in raw form is not supported.

...x .xxx
Reserved.

Bit setting 0123 4567
(Bytes 15 or 19) Meaning

Additional data stream profile

.1.. ....
Card format is supported.

.0.. ....
Card format is not supported.

...1 ....
Basic exchange format is supported.

...0 ....
Basic exchange format is not supported.

.... .1..
Word processing exchange diskette format is supported.

.... .0..
Word processing exchange diskette format is not supported.

x.x. x.xx
Reserved.

Bit setting 0123 4567
(Bytes 16 or 20) Meaning

Console

1... ....
Base data stream profile is supported.

0... ....
Base data stream profile is not supported.

.1.. ....
General data stream profile is supported.

.0.. ....
General data stream profile is not supported.

..1. ....
Job SNA character string (SCS) subset is supported.

..0. ....
Job SNA character string (SCS) subset is not supported.

...x xxxx
Reserved.

Bit setting 0123 4567
(Bytes 17 or 21) Meaning

FM or FM header usage

.00. ....
A one-level destination suspension stack is supported.

.01. ....
A two-level destination suspension stack is supported.

.10. ....
Reserved.

.11. ....
A three-level destination suspension stack is supported.

...1 ....
Compaction is supported.

...0 ....
Compaction is not supported.

.... 1...
PDIR for all media is supported.

.... 0...
PDIR for all media is not supported.

.... ..1.
Query for data set FM header type 2 is supported.

.... ..0.
Query for data set FM header type 2 is not supported.

....1
(Byte 17 only) The secondary logical unit needs to receive a CD with every EDS.

....0
(Byte 17 only) The secondary logical unit does not need to receive a CD with every EDS.

....1
(Byte 21 only) The primary logical unit needs to receive a CD with every EDS.

....0
(Byte 21 only) The primary logical unit does not need to receive a CD with every EDS.

x... .x..
Reserved.

Code selection

This 1-byte field specifies the type of data stream code that is used in the session. These codes are:

Bit setting 0123 4567
(Byte 22) Meaning

1... ....
Session is capable of EBCDIC code transmission.

.1.. ....
Session is capable of ASCII/ISCII/ITA #5 code transmission.

.... 00..
EBCDIC is the main code.

.... 01..
ASCII/ISCII/ITA #5 is the main code.

.... ..00
EBCDIC is the alternate code.

.... ..01
ASCII/ISCII/ITA #5 is the alternate code.

..xx ....
Reserved.

General characteristics

This 1-byte field defines certain general session characteristics. These characteristics are:

Bit setting 0123 4567
(Byte 23) Meaning

..0. ....
The primary logical unit can send data first.

..1. ....
The secondary logical unit must send data first.

.... 0...
The secondary logical unit initiates attended.

.... 1...
The secondary logical unit initiates unattended.

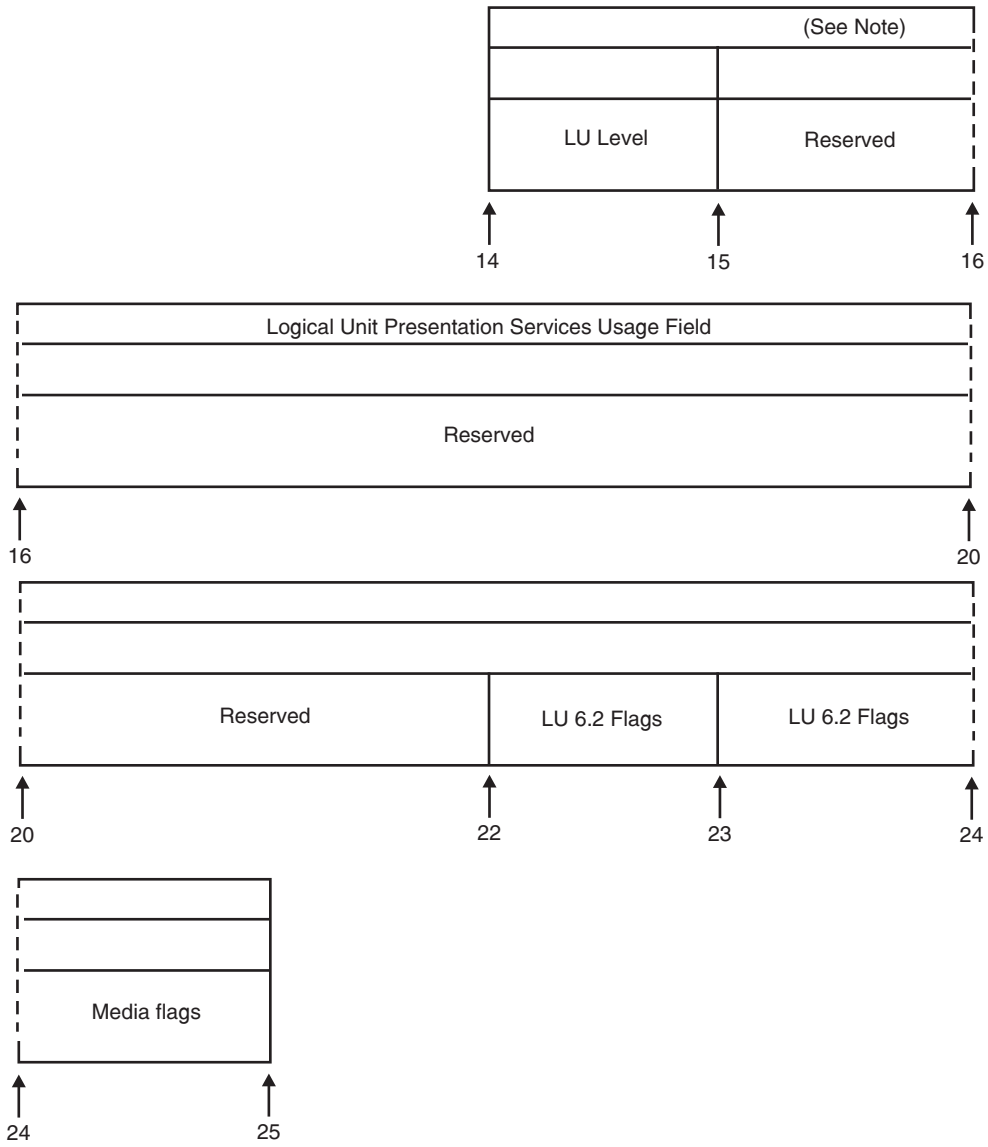
.... .0..
The secondary logical unit does not alternate between attended and unattended.

.... .1..
The secondary logical unit alternates between attended and unattended.

xx.x ..xx
Reserved.

Profile 6 for LU 6.2

Figure 176 on page 813 shows the profile 6 for LU 6.2 presentation services usage field.



Note: The dashed vertical line indicates continuation of the field to the next line.

Figure 176. Profile 6 for LU 6.2 presentation services usage field

LU level

This one-byte field indicates the LU6 level.

Bit setting 0123 4567
(Byte 14) Meaning

00000010
LU 6.2

LU 6.2 flags

These indicators show the capabilities supported for the LU 6.2 session.

Bit setting 0123 4567
(Byte 22) Meaning

xxx.....
Reserved

...1....
Access security information field will be accepted on incoming FMH-5s.

...0 ...
Access security information field will not be accepted on incoming FMH-5s.

....1...
Session level security protocol

.....x..
Reserved

......1.
Already-verified function will be accepted on incoming FMH-5s.

.... .0.
Already-verified function will not be accepted on incoming FMH-5s.

.......1
Persistent verification function will be accepted on incoming FMH-5s.

.......0
Persistent verification function will not be accepted on incoming FMH-5s.

Bit setting 0123 4567
(Byte 23) Meaning

x.......
Reserved

.xx.....
Synchronization level

.10.....
Confirm, syn point, and backout supported

.01.....
Confirm supported

...1....
Reconnect supported

...0....
Reconnect not supported

....xx..
Responsibility for session reinitiation (reserved when parallel sessions are supported)

....00..
Operator controlled

....01..
Primary will reinitiate

....10..
Secondary will reinitiate

....11..
Either can reinitiate

......1.
Parallel sessions are supported

.....0.
Parallel sessions are not supported

.....1
Change number of sessions (CNOS) GDS variable flow is supported

.....0
Change number of sessions (CNOS) GDS variable flow is not supported

LU 6.2 flags

The bit settings for byte 24 are:

Bit setting 0123 4567
(Byte 24) Meaning

x... ....
Reserved

.1.. ....
Limited resource exists

..xx xx..
Reserved

.... ..00
No compression

.... ..01
Compression bid

.... ..10
Reserved

.... ..11
Compression required

Cryptographic control

This field specifies the private-level and session-level cryptographic capabilities of this session.

CAUTION:
Session-level cryptography can interfere with private-level cryptography.

Bit setting 0123 4567
(Byte 25) Meaning

00.. ....
No private cryptography protocol.

01.. ....
Private cryptography protocol.

10.. ....
Reserved.

11.. ....
Reserved.

..00 ....
No session-level cryptography.

..01 ....
Selective session-level cryptography.

..10 ....
Reserved.

..11 ....
Required session-level cryptography (called mandatory session-level cryptography in SNA).

.... 0000
SLU is not capable of cryptography.

.... 1001
SLU is capable of cryptography (a session cryptographic option byte and key are included in the BIND RU; however, they are not included in the fixed length part of BIND).

Primary logical unit name length

This field can contain the length of the PLU name field. See the note in Figure 171 on page 794.

Primary logical unit name

This field contains the uninterpreted name of the primary logical unit. Refer to "OPNSEC macroinstruction" on page 96 for a warning about the use of the uninterpreted name.

User data length

This field specifies the length of the user data field. If hex 00 is specified, there is no user data.

Note: When sending a BIND request or a BIND response, SNA requires that the total BIND RU length (including the user data field) not exceed 256. The BIND RU consists of the preceding fields plus some additional fields supplied by VTAM; see *SNA Formats* for the BIND RU format.

User data

This field can be used to send data to the secondary logical unit as a part of the BIND request.

BIND area format and DSECT

The ISTDBIND DSECT can be used to set up or examine a set of session parameters. The situations in which the DSECT might be useful are listed here. For further information about the individual macroinstructions, see Chapter 13, "Conventions and descriptions of VTAM macroinstructions," on page 371. For a general discussion of DSECTs, see the introductory section of Appendix E, "Control block formats and DSECTs," on page 739. The format maps and DSECT descriptions for the BIND area are shown in Figure 177 on page 818 through Table 129 on page 840.

The term "fixed-length portion of BIND" refers to bytes 0–25 of a BIND area (only fixed-length fields). See Figure 177 on page 818 for an example of the fixed-length fields of a BIND area. In general, the fields in BIND after the fixed-length portion are variable-length fields, so they cannot be examined by a DSECT. However, in some instances, the primary logical unit name field is padded by VTAM to be a fixed-length (8-byte) field. In this case, the full ISTDBIND DSECT can be used.

The ISTDBIND DSECT can be used:

- To examine the session parameters obtained by INQUIRE OPTCD=SESSPARM. The full ISTDBIND DSECT can be used.
- To set up the session parameters (in the area pointed to by the NIB's BNDAREA field) sent by OPNDST in a BIND request, or sent by OPNSEC in a negotiable BIND response. The full ISTDBIND DSECT can be used.
- To examine the session parameters pointed to by the fourth word of the SCIP exit routine parameter list when a BIND is received. The full ISTDBIND DSECT can be used.
- To examine the BIND image portion of a CINIT request unit received in a LOGON exit routine. Only the fixed-length portion (bytes 0–25) of the BIND area can be examined with ISTDBIND. Note that this portion of CINIT can also be accessed by INQUIRE OPTCD=SESSPARM.
- To examine bytes 1–26 of a BIND request received in an SCIP exit routine. Byte 0 of BIND is the BIND request code and is not included in ISTDBIND. The BIND request is pointed to by the AREA field of the SCIP exit routine read-only RPL.
- To examine bytes 1–26 of a negotiable BIND response received in the AAREA field specified by OPNDST. Byte 0 of AAREA is the BIND request code and is not included in ISTDBIND.

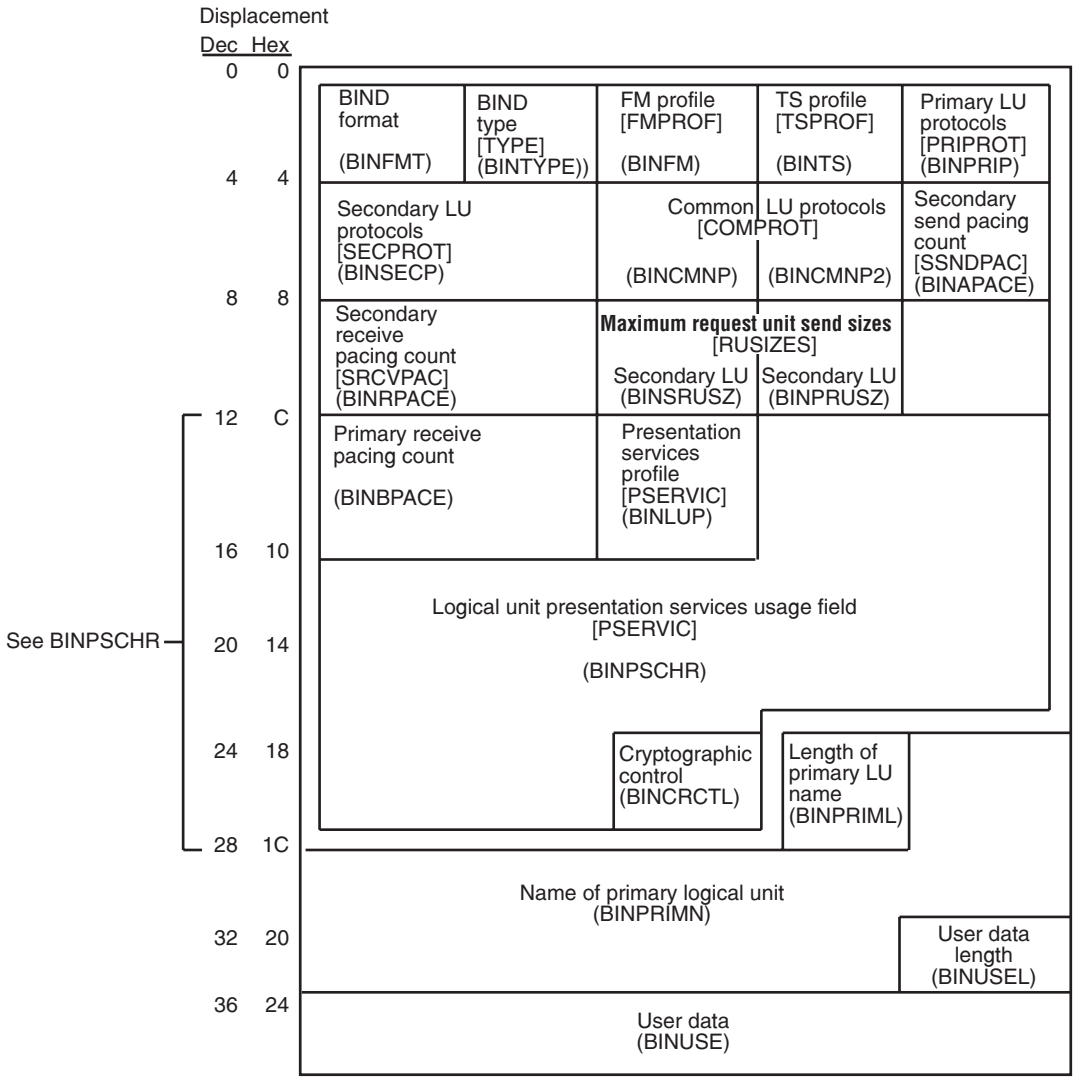


Figure 177. Format of BNDAREA (ISTDBIND).

For BINPSCHR, see Table 126 on page 829 Table 127 on page 838, Table 128 on page 839, and Table 129 on page 840.

Note: The double outlined section is the fixed-length portion of BIND. The names in brackets are the operands of the MODEENT macro used to build the corresponding fields in a logon mode table entry (refer to the z/OS Communications Server: SNA Resource Definition Reference for information on the logon mode table.) The names in parentheses are the ISTDBIND DSECT labels for the field.

Although the PLU name can be 1-17 bytes in the BIND RU, the ISTDBIND DSECT requires it to be exactly 8 bytes. This causes the offset of the user data field to vary in the BIND RU but not in the BIND DSECT. The area provided by VTAM to the SCIP exit will have an 8 in the PLU name length field and the name may be padded with blanks or truncated, if necessary. VTAM requires the bind area to be in this format when it is supplied by the application on an OPNDST or OPNSEC macroinstruction. Also note that in the case of a BIND error, the offset contained in sense code 0835*nnnn* is adjusted, if necessary, when given to the application to parse the BIND DSECT.

Table 123. Session parameter fields: How they are made available and who can change them

ISTDBIND offset	ISTDBIND field description	Information available in SCIP exit session parameters (See note 10)	Information available to INQUIRE OPTCD= SESSPARM (See note 3)	Application program can specify in OPNDST or OPNSEC (in NIB BNDAREA)	VTAM sets at OPNDST or OPNSEC (NIB BNDAREA information superseded)
–	BIND RU code (X'31')[2]	No	No	No	Yes
0	BIND format and type	Yes	Yes	No[8]	Yes[8]
1	FM profile	Yes	Yes	Yes	No
2	TS profile	Yes	Yes	Yes	No[12]
3	PLU protocols	Yes	Yes	Yes	No
4	SLU protocols	Yes	Yes	Yes	No
5–6	Common protocols	Yes	Yes	Yes	No[16]
7	SLU send pacing count	Yes	Yes	No[14]	Yes
8	SLU receive pacing count	Yes	Yes	No for OPNDST.[14] Yes for OPNSEC.[13]	Yes for OPNDST. No for OPNSEC.[12]
9	SLU maximum send RU size-mantissa/ exponent	Yes	Yes	Yes[1]	No
10	PLU maximum send RU size-mantissa/ exponent	Yes	Yes	Yes[1]	No
11	PLU send pacing count	Yes	Yes	No[14]	Yes
12	PLU receive pacing count	Yes	Yes	Yes for OPNDST. No for OPNSEC.[14]	No for OPNDST.[4] Yes for OPNSEC.
13–24	LU presentation services	Yes	Yes	Yes	No
25	Cryptographic control	Yes[6]	Yes[6]	Yes[9]	Yes[7] [12]
26	PLU name length	Yes[15]	No	No	Yes
27–34	PLU name	Yes[15]	No	No	Yes
35	User data length[5]	Yes	Yes	Yes	No
36–n[11]	User data[5]	Yes	Yes	Yes	No

Table 123. Session parameter fields: How they are made available and who can change them (continued)

ISTDBIND offset	ISTDBIND field description	Information available in SCIP exit session parameters (See note 10)	Information available to INQUIRE OPTCD= SESSPARM (See note 3)	Application program can specify in OPNDST or OPNSEC (in NIB BNDAREA)	VTAM sets at OPNDST or OPNSEC (NIB BNDAREA information superseded)

Note:

1. The RU size specified should be no larger than the size specified in the received CINIT or BIND RU.
2. The BIND request unit code (X'31') is not included in the ISTDBIND DSECT.
3. Certain information is available to an INQUIRE OPTCD=SESSPARM for a received CINIT RU which is not available for other types of INQUIREs for session parameters. This information is the BIND type and user data.
4. The PLU receive pacing count is not overlaid unless the application program sets BINBPACE to zero in BNDAREA (meaning the application program wants the CINIT value to be used), or unless the value in CINIT for this field is 0 (meaning the SLU to PLU direction is not to be paced). The value specified should be no larger than in the associated received CINIT RU.
5. The user data and user-data length have different meanings depending on when they are examined:
   - For INQUIRE OPTCD=SESSPARM for a received CINIT RU, they represent the user-data field specified in the original session-initiation request. For other types of INQUIRE, the length field is zero and there is no user data.
   - For OPNDST ACCEPT or ACQUIRE specifying a BNDAREA, they specify the user-data field to be sent in a BIND to the secondary logical unit. If BNDAREA is not specified, no user data is sent to the logical unit as part of BIND.
   - For an SCIP exit routine scheduled by the receipt of a BIND request, they specify the data received in that BIND request.
   - For OPNSEC sending a negotiable BIND response in a BNDAREA, you must specify the user data-length field (and user-data field if the length is not zero) when returning the BIND response to the PLU. If BNDAREA is not specified, no user data is sent to the logical unit.
6. Note that the cryptographic length field in this byte is always set to 0. Some of the variable-length cryptographic information is available through use of INQUIRE OPTCD=SESSKEY.
7. VTAM changes the session level cryptography part of this field (if possible) if a higher level of cryptography is required for session establishment.
8. See NIB PROC option for NEGBIND to see how the application program can set the type field indirectly. The type field of BNDAREA is ignored by VTAM; the format field is tested for 0 by VTAM.
9. See NIB ENCR parameter to see how the application program can set the session level cryptography part of this field indirectly; that part of the cryptographic-control field in BNDAREA is ignored by VTAM. The private-level-cryptography setting can be changed by the application program in BNDAREA and is not subsequently changed by VTAM.
10. These are the session parameters pointed to by word 4 of the SCIP exit parameter list, not those in the BIND RU pointed to by the SCIP exit read-only RPL.
11. The total BIND RU length (request or response) can be no larger than 256 bytes.
12. This field is checked for valid values by VTAM. The macroinstruction fails if a value that is not valid is found.
13. The application program can decrease this value, but cannot set it to 0 unless the field received in BIND is 0.
14. VTAM determines the pacing value from either the definition statement for the LU or the logon mode table entry.
15. Although the PLU name can be 1-17 bytes in the BIND RU, the ISTDBIND DSECT requires it to be exactly 8 bytes. This causes the offset of the user data field to vary in the BIND RU but not in the BIND DSECT. The area provided by VTAM to the SCIP exit will have an 8 in the PLU name length field and the name may be padded with blanks or truncated, if necessary. VTAM requires the bind area to be in this format when it is supplied by the application on an OPNDST or OPNSEC macroinstruction.
16. VTAM sets byte 6 bit 6 (Control vectors included indicator) as needed and includes any control vectors appropriate for the session.

Table 124. BNDAREA DSECT (ISTDBIND)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
BIND Format and Type	BINFMTY	BINFMT	X'F0'	BIND format	0	0
		BINFMT0	X'00'	Format 0	0	0
		BINTYPE	X'0F'	BIND type	0	0
		BINNEGO	X'00'	Negotiable	0	0
		BINONEGO	X'01'	Non-negotiable	0	0
		BINCOLD	X'01'	Non-negotiable	0	0
FM Profile	BINFM	BINFM19	X'13'	FM Profile 19	1	1
TS Profile	BINTS	BINTS2	X'02'	Sequence numbers and no reset state	2	2
		BINTS3	X'03'	Sequence numbers and reset state	2	2
		BINTS4	X'04'	Sequence numbers and reset state (STSN and RQR)	2	2
		BINTS7	X'07'	Sequence numbers and no reset state	2	2
Primary LU Protocol	BINPRIP	BINPCHN	X'80'	**Bit on =** multiple request chains **Bit off =** only single request chain used	3	3
		BINPMCH	X'40'	**Bit on =** multiple outstanding chains; delayed request mode **Bit off =** single outstanding chains only; immediate request mode	3	3
		BINPCHNR	X'30'	Primary chain response protocol	3	3
		BINNYRSP	X'30'	Either definite or exception response	3	3
		BINDFRSP	X'20'	Definite response	3	3
		BINEXRSP	X'10'	Exception response	3	3
		BINNORSP	X'00'	No response	3	3
		BINNORSP	X'00'	No response	3	3
			X'0C'	Reserved	3	3

Table 124. BNDAREA DSECT (ISTDBIND) (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINPCMP	X'02'	**Bit on =** compression can be used **Bit off =** compression must not be used	3	3
		BINPSEB	X'01'	**Bit on =** primary can send end bracket indicator **Bit off =** primary will not send end bracket indicator	3	3
Secondary LU Protocol	BINSECP	BINSCHN	X'80'	**Bit on =** multiple request chains **Bit off =** only single request chains used	4	4
		BINSMCH	X'40'	**Bit on =** multiple outstanding chains; delayed request mode **Bit off=** single outstanding chains only; immediate request mode	4	4
		BINSCHNR	X'30'	Secondary chain response protocol	4	4
		BINNYRSP	X'30'	Either definite or exception responses	4	4
		BINDFRSP	X'20'	Definite response	4	4
		BINEXRSP	X'10'	Exception response	4	4
			X'0C'	Reserved	4	4
		BINSCMP	X'02'	**Bit on =** compression can be used **Bit off =** compression must not be used	4	4

Table 124. BNDAREA DSECT (ISTDBIND) (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINSSEB	X'01'	**Bit on =** secondary can send end bracket indicator **Bit off =** secondary will not send end bracket indicator	4	4
Common LU Protocol	BINCMNP	BINWBREQ	X'80'	Whole BIUs-required indicator **Bit on =** sending LU nodes does not support receipt of segments on this session **Bit off =** sending LU nodes supports receipt of segments on this session	5	5
		BINFMHD	X'40'	**Bit on =** function management headers can be used **Bit off =** function management headers must not be used	5	5
		BINBRAK	X'20'	**Bit on =** brackets are used and the reset state is between brackets **Bit off =** either brackets are not used or brackets are used and the reset state is in brackets	5	5

Table 124. BNDAREA DSECT (ISTDBIND) (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINBKTR	X'10'	**Bit on =** conditional bracket termination (termination rule one) **Bit off =** unconditional bracket termination (termination rule two)	5	5
		BINALT	X'08'	**Bit on =** alternate code can be used **Bit off =** alternate code must not be used	5	5
			X'06'	Reserved	5	5
		BINQUE	X'01'	BIND queueing indicator **Bit on =** BIND sender allows the BIND receiver to queue the BIND for an indefinite period **Bit off =** BIND cannot be held/queued	5	5
	BINCMNP2	BINFMTRM	X'C0'	Send/Receive mode as follows:	6	6
		BINHDXFF	X'80'	Half-duplex flip-flop mode	6	6
		BINHDXC	X'40'	Half-duplex contention mode	6	6
		BINFLDPX	X'00'	Full-duplex mode	6	6
		BINRCVR	X'20'	**Bit on =** symmetric responsibility for recovery **Bit off =** contention loser is responsible for recovery	6	6

Table 124. BNDAREA DSECT (ISTDBIND) (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINBKFS	X'10'	**Bit on =** primary is first speaker in bracket mode and contention winner; secondary is brackets bidder and contention loser **Bit off =** secondary is first speaker in bracket mode and contention winner; primary is brackets bidder and contention loser	6	6
		BINASCC	X'0C'	**00 =** alternate code selection is ASCII 7 **01 =** alternate code selection is ASCII 8	6	6
		BINCTLV	X'02'	Control vectors included after SLU name.	6	6
		BINCONR	X'01'	**Bit on =** for half-duplex flip-flop mode, primary sends first when the data traffic reset state is left **Bit off =** for half-duplex flip-flop mode, secondary sends first when the data traffic reset is left	6	6
Secondary LU Send Pacing Count	BINAPACE	:c	:c	Secondary logical unit send pacing. See Table 123 on page 819 for availability.	7	7

Table 124. BNDAREA DSECT (ISTDBIND) (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINSP2ST	X'80'	Number of pacing stages from the secondary logical unit to the primary logical unit: **Bit on =** one stage **Bit off =** two stages	7	7
			X'40'	Reserved	7	7
		BINAPACM	X'3F'	Secondary logical unit send pacing count	7	7
Secondary LU Receive Pacing Count	BINRPACE			Secondary logical unit receive pacing. See Table 123 on page 819 for availability.	8	8
		BINASPI	X'80'	Adaptive pacing	8	8
			X'40'	Reserved	8	8
		BINRPACM	X'3F'	Secondary logical unit receive pacing count	8	8
Request Unit Sizes	BINSRUSZ			Maximum size of the request unit that can be sent by the secondary logical unit	9	9
		BINSRUSS	X'80'	RU size is specified	9	9
		BINRU256	X'85'	256-byte request unit (8×2^5)	9	9
		BINRU1K	X'87'	1024-byte request unit (8×2^7)	9	9
		BINRUSZM	X'F0'	Mantissa (M) mask	9	9
		BINRUSZE	X'0F'	Exponent (E) mask Size = M x 2E	9	9
	BINPRUSZ			Maximum size of the request unit that can be sent by the primary logical unit	10	A
		BINPRUSS	X'80'	RU size is specified	9	9
		BINRU256	X'85'	256-byte request unit (8×2^5)	10	A
		BINRU1K	X'87'	1024-byte request unit (8×2^7)	10	A
		BINRUSZM	X'F0'	Mantissa (M) mask	10	A
		BINRUSZE	X'0F'	Exponent (E) mask Size = M x 2E	10	A
Primary LU Send Pacing Count	BINSPACE			Primary logical unit send placing. See Table 123 on page 819 for availability.	11	B

Table 124. BNDAREA DSECT (ISTDBIND) (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINPS1ST	X'80'	Number of pacing stages from the primary logical unit to the secondary logical unit: **Bit on =** one stage **Bit off =** two stages	11	B
			X'40'	Reserved	11	B
		BINBPACM	X'3F'	Primary logical unit send pacing count	11	B
Primary LU Receive Pacing Count	BINBPACE			Primary logical unit receive pacing. See Table 123 on page 819 for availability.	12	C
			X'C0'	Reserved	12	C
		BINBPACM	X'3F'	Primary logical unit receive pacing count. PS usage field format. **0 =** Basic format **1 =** Reserved	12	C
Logical Unit Presentation Services Profile	BINLUP	BINPSFMT	X'80'	PS usage field format	13	D
		BINLUTYP	X'7F'	LU type	13	D
		BINLUP0C	X'00'	Logical unit profile 0	13	D
		BINLUP1C	X'01'	Logical unit profile 1	13	D
		BINLUP2C	X'02'	Logical unit profile 2	13	D
		BINLUP3C	X'03'	Logical unit profile 3	13	D
		BINLUP4C	X'04'	Logical unit profile 4	13	D
		BINLUP6C	X'06'	Logical unit profile 6	13	D
Logical Unit Presentation Services Usage	BINPSCHR			Logical unit presentation services. See Table 126 on page 829, Table 127 on page 838, Table 128 on page 839, and Table 129 on page 840.	14	E
Crypto-graphic Control	BINCRCTL			Cryptographic control byte	25	19
		BINNOCRY	X'00'	No cryptography	25	19
		BINCRYCA	X'09'	Capable of cryptography	25	19
		BINCRYSL	X'19'	Selective cryptography	25	19
		BINCRYRQ	X'39'	Required cryptography	25	19

Table 124. BNDAREA DSECT (ISTDBIND) (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINCEUMP	X'C0'	Private cryptography flags	25	19
		BINCEUPS	X'80'	System key, private protocol	25	19
		BINCEUPP	X'40'	Private key, private protocol	25	19
		BINCEUNP	X'00'	No private protocol	25	19
		BINCSESS	X'30'	Session level cryptography flags	25	19
		BINCSENP	X'00'	No cryptography	25	19
		BINCSESP	X'10'	Selective cryptography	25	19
		BINCSESR	X'30'	Required cryptography	25	19
		BINCLEN	X'0F'	Cryptographic field length mask	25	19
Primary LU Name Length	BINPRIML			Primary logical unit name length	26	1A
Primary LU Name	BINPRIMN			Primary logical unit name	27	1B
User-Data Length	BINUSEL			User-data length	35	23
		BINUSERD	X'00'	User-data length default	35	23
User Data	BINUSE			User data	36	24

Note:

1. Refer to the z/OS Communications Server: SNA Programmer's LU 6.2 Guide for more information on the BNDAREA DSECT (ISTDBIND).

Table 125. BINPSCHR field of BNDAREA DSECT for logical unit profile 0

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
	BINDFLAG	BINSEDS	X'80'	3270 extended data stream	14	E
			X'7F'	Reserved	14	E
				Reserved	15	F
				Reserved	16	10
				Reserved	17	11
				Reserved	18	12
	BINSCRSZ			Presentation space size (next 4 bytes)	19	13
	BINSPRIR			Default number of rows	19	13
	BINSPRIC			Default number of columns	20	14
	BINSALTR			Alternate number of rows	21	15
		BINSALTC		Alternate number of columns	22	16

Table 125. BINPSCHR field of BNDAREA DSECT for logical unit profile 0 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
Presentation Space Size	BINPRESZ			Presentation space size	23	17
		BINPSZ0	X'00'	Undefined row and column format	23	17
		BINPSZ1	X'01'	12 rows 40 columns format	23	17
		BINPSZ2	X'02'	24 rows 80 columns format	23	17
		BINPSZ3	X'03'	24 rows 80 columns default. The alternate presentation space is specified in the Query Reply structured field.	23	17
		BINPSFX	X'7E'	Presentation space is a fixed size as defined by default values	23	17
		BINPSZRC	X'7F'	Presentation space has both default and alternate sizes as defined in the DEFAULT and ALTERNATE fields	23	17
Device Type			X'C0'	Device Type: **00 =** unspecified device type **01 =** printer device **10 =** display device **11 =** reserved	24	18
			X'3C'	Reserved	24	18
		BINCMP1 BINCMP2	X'02' X'01'	BIND Request: **00 =** no compression **01 =** compression bid **10 =** reserved **11 =** compression required BIND Response: **00 =** reserved **01 =** reserved **10 =** compression used **11 =** reserved	24	18

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
FM Header Subset and Data Stream Profile	BINLUP1	BINFMHS1	X'F0'	Function management header subset	14	E

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINFMS3C	X'30'	FM header subset 3 used–Data management subset	14	E
		BINFMS2C	X'20'	FM header subset 2 used–Type 1 headers	14	E
		BINFMS1C	X'10'	FM header subset 1 used–Type 1 headers with restrictions	14	E
		BINFMS0C	X'00'	FM header subset 0 used–No function management headers allowed	14	E
		BINDSP1	X'0F'	Data stream profile	14	E
		BINDSP1C	X'01'	Basic controls, cards can span RUs (data stream subset 1)	14	E
		BINDSP0C	X'00'	Basic controls (data stream subset 0)	14	E
Primary Logical Unit FM Header Subset Flags	BINPFMB1	BINDESTS	X'80'	**Bit on =** three destinations might be outstanding **Bit off =** two destinations might be outstanding	15	F
		BINCMPCT	X'40'	**Bit on =** can send compaction table/can be queried for compaction tables **Bit off =** will not send compaction table/will not be queried for compaction tables	15	F
		BINPDIR	X'20'	**Bit on =** PDIR can be sent **Bit off =** PDIR will not be sent	15	F
			X'1F'	Reserved	15	F
		BINKDDSI	X'10'	**Bit on =** keyed direct data sets can be sent **Bit off =** keyed direct data sets will not be sent	15	F
		BINSDSI	X'08'	**Bit on =** sequential data sets can be sent **Bit off =** sequential data sets will not be sent	15	F

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINSAI	X'04'	**Bit on =** sequential access to addressed direct data sets can be sent **Bit off =** sequential access to addressed direct data sets will not be sent	15	F
		BINSIDS	X'02'	**Bit on =** series IDs (with status in reply) are supported **Bit off =** series IDs are not supported	15	F
		BINARRR	X'01'	**Bit on =** Add Replicate, Replace Replicate are supported **Bit off =** Add Replicate, Replace Replicate are not supported	15	F
		BINPFMB2	X'80'	Reserved	16	10
		BINQDSI	X'40'	**Bit on =** query for destination selection is supported **Bit off =** query for destination selection is not supported	16	10
		BINCSDS	X'20'	**Bit on =** CREATE, SCRATCH, and SCRATCH ALL are permitted **Bit off =** CREATE, SCRATCH, and SCRATCH ALL are not permitted	16	10
		BINXFPD	X'10'	**Bit on =** Execute Program Offline is permitted **Bit off =** Execute Program Offline is not permitted	16	10
			X'0F'	Reserved	17	10

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
Primary Logical Unit Data Stream Flags	BINPDSB1	BININTR	X'80'	**Bit on =** full base set data stream (BS, CR, LF, ENP, INP HT, VT) can be sent **Bit off =** full base set data stream will not be sent	17	11
		BINHFDS	X'40'	**Bit on =** horizontal format data stream (SHF) can be sent **Bit off =** horizontal format data stream will not be sent	17	11
		BINVTDS	X'20'	**Bit on =** vertical format data stream (SVF) can be sent **Bit off =** vertical format data stream will not be sent	17	11
		BINVSDS	X'10'	**Bit on =** vertical channel (includes vertical format and vertical channel select (VCS) with parameters) can be sent **BIN off =** vertical channel will not be sent	17	11
		BINSLD	X'08'	**Bit on =** SLD can be sent **Bit off =** SLD will not be sent	17	11
			X'06'	Reserved	17	11
		BINTRNDS	X'01'	**Bit on =** transparency data stream (TRN, IRS) can be sent **Bit off =** transparency data stream will not be sent	17	11
	BINPDSB2	BINUAINT	X'80'	**Bit on =** secondary logical unit will initiate unattended **Bit off =** secondary logical unit will initiate attended	18	12

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINUAALT	X'40'	**Bit on =** in session, the secondary logical unit will alternate between attended and unattended **Bit off =** in session, secondary logical unit will not alternate between attended and unattended	18	12
			X'3F'	Reserved	18	12
Primary Logical Unit Media Flags	BINPMED1	BINDOCMT	X'80'	**Bit on =** document format can be sent **Bit off =** document format will not be sent	19	13
		BINCARD	X'40'	**Bit on =** card format can be sent **Bit off =** card format will not be sent	19	13
		BINXCHNG	X'20'	**Bit on =** exchange media can be sent **Bit off =** exchange media will not be sent	19	13
		BINDISK	X'10'	**Bit on =** disk data management can be sent **Bit off =** disk data management will not be sent	19	13
		BINXCDF	X'08'	**Bit on =** extended card format can be sent **Bit off =** extended card format will not be sent	19	13

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINXDOCF	X'04'	**Bit on =** extended document format can be sent **Bit off =** extended document format will not be sent	19	13
		BINCDEDS	X'02'	**Bit on =** secondary logical unit must send Change Direction every EDS **Bit off =** secondary logical unit can send Change Direction every EDS	19	13
			X'01'	Reserved	19	13
Secondary Logical Unit FM Header Subset Flags	BINSFMB1	BINDESTS	X'80'	**Bit on =** three destinations might be outstanding **Bit off =** two destinations might be outstanding	20	14
		BINCMPCT	X'40'	**Bit on =** can send compaction table/can be queried for compaction tables **Bit off =** will not send compaction table/will not be queried for compaction tables	20	14
		BINPDIR	X'20'	**Bit on =** PDIR can be sent **Bit off =** PDIR will not be sent	20	14
		BINKDDSI	X'10'	**Bit on =** keyed direct data sets can be sent **Bit off =** keyed direct data sets will not be sent	20	14
		BINSDSI	X'08'	**Bit on =** sequential data sets can be sent **Bit off =** sequential data sets will not be sent	20	14

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINSAI	X'04'	**Bit on =** sequential access to addressed direct data set can be sent **Bit off =** these data sets will not be sent	20	14
		BINSIDS	X'02'	**Bit on =** series IDs (with status in reply) are supported **Bit off =** series IDs are not supported	20	14
		BINARRR	X'01'	**Bit on =** Add Replicate, Replace Replicate are supported **Bit off =** these are not supported	20	14
	BINSFMB2		X'80'	Reserved	21	15
		BINQDSI	X'40'	**Bit on =** query for destination selection is supported **Bit off =** query for destination selection is not supported	21	15
		BINCSDS	X'20'	**Bit on =** CREATE, SCRATCH, and SCRATCH ALL are permitted **Bit off =** these are not permitted	21	15
		BINXFPD	X'10'	**Bit on =** Execute Program Offline is permitted **Bit off =** Execute Program Offline is not permitted	21	15
			X'0F'	Reserved	21	15

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
Secondary Logical Unit Data Stream Flags	BINSDSB1	BININTR	X'80'	**Bit on =** full base set data stream (BS, CR, LF, ENP, INP, HT, VT) can be sent **Bit off =** full base set data stream will not be sent	22	16
		BINHFDS	X'40'	**Bit on =** horizontal format data stream (SHF) can be sent **Bit off =** horizontal format data stream will not be sent	22	16
		BINVTDS	X'20'	**Bit on =** vertical format data stream (SVF) can be sent **Bit off =** vertical format data stream will not be sent	22	16
		BINVSDS	X'10'	**Bit on =** vertical channel (includes vertical format and vertical channel select (VCS) with parameters) can be sent **Bit off =** vertical channel will not be sent	22	16
		BINSLD	X'08'	**Bit on =** SLD can be sent **Bit off =** SLD will not be sent	22	16
			X'06'	Reserved	22	16
		BINTRNDS	X'01'	**Bit on =** transparency data stream (TRN, IRS) can be used **Bit off =** transparency data stream will not be sent	22	16
	BINSDSB2		X	Reserved	23	17

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
Secondary Logical Unit Media Flags	BINSMED1	BINDOCMT	X'80'	**Bit on =** document format can be sent **Bit off =** document format will not be sent	24	18
		BINCARD	X'40'	**Bit on =** card format can be sent **Bit off =** card format will not be sent	24	18
		BINXCHNG	X'20'	**Bit on =** exchange media can be sent **Bit off =** exchange media will not be sent	24	18
		BINDISK	X'10'	**Bit on =** disk data management can be sent **Bit off =** disk data management will not be sent	24	18
		BINXCDF	X'08'	**Bit on =** extended card format can be sent **Bit off =** extended card format will not be sent	24	18
		BINXDOCF	X'04'	**Bit on =** extended document format can be sent **Bit off =** extended document format will not be sent	24	18

Table 126. BINPSCHR field of BNDAREA DSECT for logical unit profile 1 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINCMP1 BINCMP2	X'02' X'01'	BIND Request: **00 =** no compression **01 =** compression bid **10 =** reserved **11 =** compression required BIND Response: **10 =** compression used **xx =** reserved	24	18

Table 127. BINPSCHR field of BNDAREA DSECT for logical unit profile 2

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
	BINDFLAG	BINSEDS	X'80'	3270 extended data stream	14	E
			X'7F'	Reserved	14	E
				Reserved	15	F
				Reserved	16	10
				Reserved	17	11
				Reserved	18	12
	BINSCRSZ			Presentation space size (next 4 bytes)	19	13
	BINSPRIR			Default number of rows	19	13
	BINSPRIC			Default number of columns	20	14
	BINSALTR			Alternate number of rows	21	15
	BINSALTC			Alternate number of columns	22	16
Presentation Space Size	BINPRESZ			Presentation space size	23	17
		BINPSZO	X'00'	Undefined row and column format	23	17
		BINPSZ1	X'01'	12 rows 40 columns format	23	17
		BINPSZ2	X'02'	24 rows 80 columns format	23	17
		BINPSZ3	X'03'	24 rows 80 columns default to undefined alternate. Do not write structural field query to identify alternate.	23	17
		BINPSFX	X'7E'	Presentation space is a fixed size as defined by default values	23	17
		BINPSZRC	X'7F'	Presentation space has both default and alternate sizes as defined in the DEFAULT and ALTERNATE fields	23	17
			X'FC'	Reserved	24	18

Table 127. BINPSCHR field of BNDAREA DSECT for logical unit profile 2 (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
		BINCMP1 BINCMP2	X'02' X'01'	BIND Request: **00 =** no compression **01 =** compression bid **10 =** reserved **11 =** compression required BIND Response: **10 =** compression used **xx =** reserved	24	18

Table 128. BINPSCHR field of BNDAREA DSECT for logical unit profile 3

DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
BINDFLAG	BINSEDS	X'80'	3270 extended data stream	14	E
		X'7F'	Reserved	14	E
			Reserved	15	F
			Reserved	16	10
			Reserved	17	11
			Reserved	18	12
BINBFRSZ			Presentation space size (next 4 bytes)	19	13
BINBFRDR			Default number of rows	19	13
BINBFRDC			Default number of columns	20	14
BINBFRAR			Alternate number of rows	21	15
BINBFRAC			Alternate number of columns	22	16
BINBDESC			Code for presentation size: **0** maximum **1** 480 characters **2** 1920 characters **X'7E'** fixed size as defined by the default values **X'7F'** variable size as defined by the default and alternate values	23	17
		X'FC'	Reserved	24	18

Table 128. BINPSCHR field of BNDAREA DSECT for logical unit profile 3 (continued)

DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
	BINCMP1 BINCMP2	X'02' X'01'	BIND Request: **00 =** no compression **01 =** compression bid **10 =** reserved **11 =** compression required BIND Response: **10 =** compression used **xx =** reserved	24	18

Table 129. BINPSCHR field of BNDAREA DSECT for logical unit profile 4

DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
BINPSNDO			Primary logical unit send capability (next 4 bytes)	14	E
BINPDSPP			Printer data stream profile	14	E
	BINPBDSP	X'80'	Base data stream profile **Bit on =** supported **Bit off =** not supported	14	E
	BINPJOB	X'20'	Job SNA character string subset **Bit on =** supported **Bit off =** not supported	14	E
	BINWPRAW	X'08'	Word processing raw form **Bit on =** supported **Bit off =** not supported	14	E
		X'57'	Reserved	14	E
BINADSPP			Additional data stream profile	15	F
	BINADSCD	X'40'	**Bit on =** card supported **Bit off =** card not supported	15	F
		X'5F'	Reserved	15	F
BINCSLP			Console data stream profile	16	10

Table 129. BINPSCHR field of BNDAREA DSECT for logical unit profile 4 (continued)

DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
	BINCBDSP	X'80'	Base data stream profile **Bit on =** supported **Bit off =** not supported	16	10
	BINCJOB	X'20'	Job SNA character string subset **Bit on =** supported **Bit off =** not supported	16	10
		X'5F'	Reserved	16	10
BINFMHUP			Function management/function management header usage	17	11
	BINDSSTO	X'60'	**00** 1 level of destination selection suspension stack **01** 2 levels of destination selection suspension stack **10** Reserved **11** 3 levels of destination selection suspension stack	17	11
	BINKIXS	X'01'	**Bit on =** secondary logical unit must receive a Change Direction on every End Data Set command **Bit off =** secondary logical unit need not receive a Change Direction on every End Data Set command	17	11
		X'9E'	Reserved	17	11
BINSSNDO			Secondary logical unit send capability (next 4 bytes)	18	12
BINPDSPS			Printer data stream profile	18	12
	BINPBDSP	X'80'	Base data stream profile **Bit on =** supported **Bit off =** not supported	18	12
	BINPJOB	X'20'	Job SNA character string subset **Bit on =** supported **Bit off =** not supported	18	12

Table 129. BINPSCHR field of BNDAREA DSECT for logical unit profile 4 (continued)

DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
	BINWPRAW	X'08'	Word processing raw form **Bit on =** supported **Bit off =** not supported	18	12
		X'57'	Reserved	18	12
BINADSPS			Additional data stream profile	19	13
	BINADSCD	X'40'	**Bit on =** card supported **Bit off =** card not supported	19	13
		X'BF'	Reserved	19	13
BINSCLS			Console data stream profile	20	14
	BINCBDSP	X'80'	Base data stream profile **Bit on =** supported **Bit off =** not supported	20	14
	BINCJOB	X'20'	Job SNA character string subset **Bit on =** supported **Bit off =** not supported	20	14
		X'5F'	Reserved	20	14
BINFMHUS			Function management/function management header usage	21	15
	BINDSSTO	X'60'	**00** 1 level of destination selection suspension stack **01** 2 levels of destination selection suspension stack **10** Reserved **11** 3 levels of destination selection suspension stack	21	15
	BINKIXS	X'01'	**Bit on =** secondary logical unit must receive a Change Direction on every End Data Set command **Bit off =** secondary logical unit need not receive a Change Direction on every End Data Set command	21	15
		X'9E'	Reserved	21	15

Table 129. BINPSCHR field of BNDAREA DSECT for logical unit profile 4 (continued)

DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
BINSCO			Code selection	22	16
	BINCSOR	X'F0'	Repertoire	22	16
	BINCSOE	X'80'	EBCDIC	22	16
	BINCSOA1	X'40'	ASCII/ISCII/ITA #5	22	16
		X'30'	Reserved	22	16
	BINCSOC1	X'0C'	**00 =** code 0 (main code) selection is EBCDIC **01 =** code 0 (main code) selection is ASCII/ISCII/ITA #5	22	16
	BINCSOC2	X'03'	**00 =** code 1 (alternate code) selection is EBCDIC **01 =** code 1 (alternate code) selection is ASCII/ISCII/ITA #5	22	16
BINGENCO			General characteristics	23	17
		X'C0'	Reserved	23	17
	BINWSDF	X'20'	**Bit on =** secondary logical unit must send data first **Bit off =** primary logical unit can send data first	23	17
		X'10'	Reserved	23	17
	BINIAO	X'08'	**Bit on =** secondary logical unit will initiate unattended **Bit off =** secondary logical unit will initiate attended	23	17
	BINAAO	X'04'	**Bit on =** secondary logical unit can alternate between attended and unattended **Bit off =** secondary logical unit will not alternate between attended and unattended	23	17
		X'03'	Reserved	24	18

Table 130. BINPSCHR field of BNDAREA DSECT for logical unit profile 6

DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
BINLULEV			LU 6 Level	14	E

Table 130. BINPSCHR field of BNDAREA DSECT for logical unit profile 6 (continued)

DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
	BINLV02	X'02'	Level 2	14	E
BINFLG0			LU 6.2 flags, byte 0	21	15
	BINDSSSP	X'80'	Distributed Systems Security supported	21	15
	BINDESS	X'40'	Extended security sense codes supported	21	15
BINFLG1			LU 6.2 flags, byte 1	22	16
	BINCLSS	X'10'	Access security information support	22	16
	BINDPWS	X'04'	Password substitution support	22	16
	BINSLAPS	X'08'	Session level security protocol	22	16
	BINAVFS	X'02'	Already-verified function support	22	16
	BINPV	X'01'	Persistent verification function support	22	16
BINFLG2			LU 6.2 flags, byte 2	23	17
	BINSYNCH	X'60'	Synchronization level field	23	17
	BINCONF	X'20'	Confirm supported	23	17
	BINCSBK	X'40'	Confirm, sync point, and backout supported	23	17
	BINRS	X'10'	Reconnect support	23	17
	BINRSR	X'0C'	Responsibility for session, reinitiation field (reserved when parallel sessions are supported)	23	17
	BINOPRC	X'00'	Operator controlled	23	17
	BINPRIMH	X'04'	Primary will reinitiate	23	17
	BINSECNH	X'08'	Secondary will reinitiate	23	17
	BINETHR	X'0C'	Either may reinitiate	23	17
	BINPSS	X'02'	Parallel session support for LU-LU pair: 0= PSS not supported, 1= PSS supported	23	17
	BINGDSVF	X'01'	Change number of sessions GDS variable flow support: 0= not supported, 1= supported	23	17
BINFLG3		X	LU 6.2 Flags, byte 3	24	18
	BINRSV37	X'80'	Reserved	24	18

Table 130. BINPSCHR field of BNDAREA DSECT for logical unit profile 6 (continued)

DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Dec offset	Hex offset
	BINLTDRC	X'40'	1 = limited resource exists	24	18
	BINRSV38	X'3C'	Reserved	24	18
	BIN6CMP1	X'02'	Applies only to BINSMED1	24	18
	BIN6CMP2	X'01'	(See BINCMP1 and BINCMP2)	24	18
	BINCMP1 BINCMP2	X'02' X'01'	BIND Request: **00 =** no compression **01 =** compression bid **10 =** reserved **11 =** compression required BIND Response: **10 =** compression used **xx =** reserved	24	18

XRF session activation control vector

The XRF session control vector consists of:

1. A key (hex 27) which identifies the type of vector.
2. The length of the vector data field.
3. The vector data which consists of usage indicators to indicate whether the BIND request is for a primary XRF session or a backup XRF session.

   Note: The XRF session request type (primary or backup) is the initial XRF state of this session. Subsequent SWITCH commands can be issued to change this status after the session establishment has been completed.
4. BIND correlation ID, used to relate the BIND for primary XRF session to a subsequent BIND for the backup XRF sessions.

   Note: When the XRF session activation control vector specifies that a backup XRF session is to be activated, the application program must ensure compatible session parameters between this backup XRF session and the related primary XRF session.

Table 131 on page 846 shows the structure of the XRF Vector hex 27. This XRF session activation control vector must be appended to the user data field of the BIND for XRF sessions.

Table 131. Structure of the XRF vector hex 27

Byte	Description
0	Key: X'27'
1	Length (n-1), in binary, of vector data field
2–n	Vector data
2	Usage indicators bit 0, session type: **0** XRF primary: the BIND request is for a primary XRF session which might become related to one or more backup sessions **1** XRF backup: the SLU will relate this LU-LU session with the previously activated session
3–n	Session correlator field
3	Length of session correlator
4–n	Session correlator

Appendix G. RPL fields associated with VTAM macroinstructions

VTAM can alter RPL fields prior to posting an RPL-based macroinstruction complete. Figure 178 on page 848, Figure 179 on page 849, and Figure 180 on page 850 show fields modified by the SEND and SESSIONC macroinstructions. Figure 181 on page 851 and Figure 182 on page 852 show fields modified by the RECEIVE macroinstruction.

© Copyright IBM Corp. 2000, 2013

	SEND POST=SCHED					SEND POST=RESP				SESSIONC	
	DFSYN Data		DFSYN DFC		DFASY	DFSYN Data	DFSYN DFC	DFSYN DFC	DFASY		
Applicable RPL Fields:	**Rq**	**Rsp**	**for (ORDRESP) Rq**	**Rsp**	**Rsp**	**Rq**	**for (ORDRESP) Rq**	**for (NORDRESP) Rq**	**Rq**	**Rq**	**Rsp**
ACB	A	A	A	A	A	A	A	A	A	A	A
ARG/NIB (when ARG specified)	A	A	A	A	A	AV[1]	AV[1]	AV[1]	AV[1]	AV[1]	A
ARG/NIB (when NIB specified)	---	---	---	---	---	---	---	---	---	---	A[2]
AREA	A					A					
RECLEN	A					A					
BRANCH	A	A	A	A	A	A	A	A	A	A	A
EXIT/ECB (when ECB specified)	A	A	A	A	A	A	A	A	A	A	A
EXIT/ECB (when EXIT specified)	◄ A for ASY; AV for SYN ►										
EXIT/ECB (when internal ECB is used)	V	V	V	V	V	V	V	V	V	V	V
REQ	V	V	V	V	V	V	V	V	V	V	V
RTNCD[3]	V	V	V	V	V	V	V	V	V	V	V
FDBK2[3]	V	V	V	V	V	V	V	V	V	V	V
FDBK[4]	V	V	V	V	V	V	V	V	V	V	V
USER	V	V	V	V	V	V	V	V	V	V	V
SEQNO	V	AV[1]	V	AV[1]	A	V	V	V	AV[1]	V	AV[1]
POST	A(SCHED)	(SCHED)	A(SCHED)[5]	(SCHED)	(SCHED)	A(RESP)	A(RESP)	A(RESP)	(RESP)	(RESP)	(SCHED)
RESPOND:											
EX/NEX	A	A	A	A	A	A	A	(NEX)	(NEX)	(NEX)	A
FME/NFME	A	A	A	A	(FME)	A	A	(FME)	(FME)	(FME)	(FME)
RRN/NRRN	A	A	A	A	(NRRN)	A	A	(NRRN)	(NRRN)	(NRRN)	(NRRN)
QRESP/NQRESP	A	A	A	A	(NQRESP)	A	A	(NQRESP)	(NQRESP)	(NQRESP)	(NQRESP)
Notes: Applicable to RESPOND	6	7	6	7		7, 8, 9	7, 8, 9	9	9	9	
CONTROL	A	A	A	A	A	AV[1]	AV[1]	AV[1]	AV[1]	AV[1]	A
CHAIN	A(F|M|L|O)[10]	(O)[11]	(O)	(O)	(O)	A(F|M|L|O)	(O)	(O)	(O)	(O)	(O)
CHNGDIR CMD	A	(NCMD)	A[12]	(NCMD)	(NCMD)	A	A[12]	A[12]	(NCMD)	(NCMD)	(NCMD)
CHNGDIR REQ	A	A	A	A	A	AV	AV	AV	AV	(NREQ)V	(NREQ)
RTYPE	A	A	A	A	A	A	A	A	A		
STYPE	A	A	A	A	A	A	A	A	A	A	A
SSENSEO[14]		A	A[13]	A	A		A[13]	A[13]			A
SSENSMO[14]		A	A[13]	A	A		A[13]	A[13]			A
USENSED[14]		A	A[13]	A	A		A[13]	A[13]			A
SSENSEI[3]						V	V	V	V	V	

Figure 178. RPL fields associated with the SEND and SESSIONC macroinstructions for various modes of operation (Part 1 of 3)

	SEND POST=SCHED					SEND POST=RESP				SESSIONC	
	DFSYN Data		DFSYN DFC		DFASY	DFSYN Data	DFSYN DFC	DFSYN DFC	DFASY		
Applicable RPL Fields:	**Rq**	**Rsp**	**for (ORDRESP) Rq**	**Rsp**	**Rsp**	**Rq**	**for (ORDRESP) Rq**	**for (NORDRESP) Rq**	**Rq**	**Rq**	**Rsp**
SSENSMI[3]						V	V	V	V	V	
USENSEI[3]						V	V	V	V	V	
CRYPT	A	(NO)	(NO)	(NO)	(NO)	A	(NO)	(NO)	(NO)	(NO)	(NO)
RPLURH[21]	A	A	A	A	A	A	A	A	A		
BRACKET BB	A	(NBB)	A[15]	(NBB)	(NBB)	A	A[15]	A[15]	(NBB)	(NBB)	(NBB)
BRACKET EB	A	(NEB)	A[16]	(NEB)	(NEB)	A	A[16]	A[16]	(NEB)	(NEB)	(NEB)
BRACKET CEB	A	(NCEB)	A[22]	(NCEB)	(NCEB)	A	A[22]	A[22]	(NCEB)	(NCEB)	(NCEB)
IBSQAC										AV[17]	A[17]
OBSQAC										AV[17]	A[17]
IBSQVAL										AV[17]	A[17]
OBSQVAL	V[23]					V[23]				AV[17]	A[17]
SIGDATA									A[20]		
CODESEL	A	(S)[18]	(S)	(S)	(S)	A	(S)	(S)	(S)	(S)	(S)
OPTCD:											
FMHDR-NFMHDR	A	A	(F)[19]	(F)	(F)	A	(F)	(F)	(F)	(F)	(F)
SYN-ASY	A	A	A	A	A	A	A	A	A	A	A
CS-CA	A	A	A	A	A	A	A	A	A		
LMPEO-NLMPEO	A					A					
CONTCHN-NOCNTCHN	A					A					
BUFFLST-NBUFFLST	A					A					
USERRH-NUSERRH	A	A	A	A	A	A	A	A	A		

Figure 179. RPL fields associated with the SEND and SESSIONC macroinstructions for various modes of operation (Part 2 of 3)

[A] Application program specifies (by setting in RPL or on RPL-based macroinstruction). VTAM uses the RPL file (for example, to set up the RH field).

[---] Cannot specify; results in error.

[ ] A blank box means the application program can specify (for example, change RPL); however, VTAM does not look at the RPL field and does not change the RPL field when posting the macroinstruction complete.

[V] VTAM changes the field when posting the macroinstruction complete.

[(x)] No matter what the application program specifies in the RPL, VTAM acts as if x had been specified. VTAM does not change the RPL unless [V] is also listed.

1 The setting made by VTAM ([V]) will be the same as the setting made by the application program ([A]) except when a VTAM or network error occurs, or if the logical unit does not obey SNA protocols.

2 [A] , if SESSIONC is used to send a negative response to a BIND request; otherwise, [---] .

3 VTAM sets this field to zero when the macroinstruction is accepted. A non-zero value may be set by VTAM when the macroinstruction is posted complete.

4 VTAM sets this field to zero when the macroinstruction is accepted.

5 For NIB PROC=ORDRESP; if NIB PROC=NORDRESP, [(RESP)] is assumed.

6 Check is made for (EX, NFME, NRRN), which is not allowed; if found, (NEX, NFME, NRRN) is assumed.

7 Check is made for (NEX, NFME, NRRN) and (EX, NFME, NRRN); if found SEND is rejected with (RTNCD, FDBK2)=(20,59).

8 If EX is specified, POST=SCHED is assumed. If (NEX, NFME, NRRN) is specified, POST=SCHED is assumed.

9 On posting complete, the RPL RESPOND field is changed to whatever was received.

10 [A(F|M|L|O)] means [A (FIRST or MIDDLE or LAST or ONLY)] .

11 [(O)] means [(ONLY)] .

12 Only for CONTROL=CANCEL, CHASE, QC, or LUS; otherwise, [(NCMD)] .

13 For CONTROL=LUS.

14 VTAM sets this field to zero when the macroinstruction is posted complete.

15 For CONTROL=LUS; otherwise, [(NBB)] .

16 Only for CONTROL=CANCEL, CHASE, QC, or LUS; otherwise, [(NEB)] .

17 Only for CONTROL=STSN; otherwise, ignored (for [A]) and not changed by VTAM.

18 [(S)] means [(STANDARD)] .

19 [(F)] means [(FMHDR)] .

20 Only for SIGNAL; otherwise, the RPL field is ignored by VTAM.

21 RPLURH is a label in the ISTRH DSECT (Appendix E), rather than a field name. There is no RPL operand for this field.

22 For CONTROL=LUS; otherwise, [(NCEB)] .

23 This field is set by VTAM when SEND OPTCD=LMPEO is posted complete.

Figure 180. RPL fields associated with the SEND and SESSIONC macroinstructions for various modes of operation (Part 3 of 3)

Applicable RPL Fields:	RECEIVE SPEC					RECEIVE ANY	RESP Exit	DFASY Exit	SCIP Exit	LOGON Exit	NSEXIT Exit
	DFSYN Data		DFSYN DFC		DFASY						
	Rq	Rsp	Rq	Rsp	Rq						
ACB	A	A	A	A	A	A	V	V	V	V	V
ARG/NIB (when ARG specified)	AV [1]	AV [1]	AV [1]	AV [1]	AV [1]	V	V	V	V	V	V [2]
ARG/NIB (when NIB specified)	---	---	---	---	---	---					
AREA	A		A [3]			A [4]			V [5]	V	V
AREALEN	A		A [3]			A [4]			V [5]	V	V
RECLEN	V					V [4]			V [5]	V	V
BRANCH	A	A	A	A	A	A					
EXIT/ECB (when ECB specified)	A	A	A	A	A	A					
EXIT/ECB (when EXIT specified)	◄ (A for ASY; AV for SYN) ►										
EXIT/ECB (when internal ECB is used)	V	V	V	V	V	V					
REQ	V	V	V	V	V	V					
RTNCD [6]	V	V	V	V	V	V					
FDB2 [6]	V	V	V	V	V	V					
FDBK [7]	V	V	V	V	V	V					
USER	V	V	V	V	V	V	V	V	V		
SEQNO	V	V	V	V	V	V	V	V	V		
POST											
RESPOND	V	V	V	V	V	V	V	V	V	V	V
CONTROL	V	V	V	V	V	V	V	V	V	V	V
CHAIN	V		V			V [4]			V [5]	V	V
CHNGDIR CMD	V		V		V	V [4]		V	V		
CHNGDIR REQ	V	V	V	V	V	V	V	V	V		
RTYPE	AV	AV	AV	AV	AV	AV	V	V	V		
SSENSEI [6]	V	V	V	V	V	V	V	V	V		
SSENSMI [6]	V	V	V	V	V	V	V	V	V		
USENSEI [6]	V	V	V	V	V	V	V	V	V		
CRYPT	V					V [4]					
BRACKET BB	V		V		V	V [4]		V	V		
BRACKET EB	V		V		V	V [4]		V	V		
BRACKET CEB	V		V		V	V [4]		V	V		
RPLURH [11]	V	V	V	V	V	V	V	V	V		V
IBSQAC									V [9]		
OBSQAC									V [9]		

Figure 181. RPL fields associated with the RECEIVE macroinstruction for various modes of operation (Part 1 of 2)

Applicable RPL Fields:	RECEIVE SPEC					RECEIVE ANY	RESP Exit	DFASY Exit	SCIP Exit	LOGON Exit	NSEXIT Exit
	DFSYN Data		DFSYN DFC		DFASY						
	Rq	Rsp	Rq	Rsp	Rq						
IBSQVAL									V [9]		
OBSQVAL									V [9]		
SIGDATA					V [10]	V [10]		V [10]			
CODESEL	V		V		V	V [4]		V	V		
OPTCD:											
TRUNC-KEEP-NIBTK	A					A [4]					
FMHDR-NFMHDR	V	V	V	V	V	V	V	V	V	V	V
SPEC-ANY	A(SPEC)	A(SPEC)	A(SPEC)	A(SPEC)	A(SPEC)	A(ANY)					
SYN-ASY	A	A	A	A	A	A					
CS-CA	A	A	A	A	A	A					
Q-NQ	A	A	A	A	A	A					

A Application program specifies (by setting in RPL or on RPL-based macroinstruction). VTAM uses the RPL field (for example, to set up the RH field).

--- Cannot specify; results in error.

A blank means the application program can specify (for example, change RPL); however, VTAM does not look at the RPL field and does not change the RPL field when posting the macroinstruction complete.

V VTAM changes the field when posting the macroinstruction complete.

(x) No matter what the application specifies in the RPL, VTAM acts as if x had been specified. VTAM does not change the RPL unless V is also listed.

1 The setting made by VTAM (V) will be the same as the setting made by the application program (A) except when a VTAM or network error occurs, or if the logical unit does not obey SNA protocols.

2 ARG is set only for a CLEANUP RU.

3 Application program must specify this field for RTYPE=DFSYN because it is required for DFSYN CONTROL=DATA. However, the RPL field is not used if the RU actually received is DFSYN DFC.

4 Used (A) or set (V) if meaningful for the type of RU that completes the RECEIVE. See the corresponding RECEIVE SPEC column.

5 When an RU (BIND or UNBIND) is made available in the exit routine.

6 VTAM sets this field to zero when the macroinstruction is accepted. A non-zero value may be set by VTAM when the macroinstruction is posted complete.

7 VTAM sets this field to zero when the macroinstruction is accepted.

8 VTAM sets this field to zero when the macroinstruction is posted complete.

9 Only for CONTROL=STSN; otherwise, ignored (for A) and not changed by VTAM.

10 Set only for CONTROL=SIGNAL RU received.

11 RPLURH is a label in the ISTRH DSECT (Appendix E), rather than a field name. There is no RPL operand for this field.

Figure 182. RPL fields associated with the RECEIVE macroinstruction for various modes of operation (Part 2 of 2)

Appendix H. Summary of register usage

The following table shows what VTAM does with the general-purpose registers before it returns control to the application program at the next sequential instruction. It indicates which registers are left unchanged by the VTAM macroinstructions and which ones can be modified between the time the macroinstruction is executed and control is returned to the application program. The table also shows the disposition of the registers when any of the exit routines receive control. Refer to Chapter 9, "Handling errors and special conditions," on page 277 for further details on how to handle macroinstruction errors.

Table 132. Register contents upon return of control

	Register 0	Register 1	Register 2-12	Register 13	Register 14	Register 15
Upon return from OPEN and CLOSE macroinstructions	Unpredictable	Unpredictable	Unmodified	Unmodified[1]	Unpredictable	Return code
Upon return from RPL-based macroinstructions, including CHECK	See footnote[2]	Address of RPL	Unmodified	Unmodified[1]	Unpredictable	See footnote[2]
Upon return from GENCB	Error return code or control block length[3] [4]	Control block address[3] [4]	Unmodified	Unmodified[1]	Unpredictable	General return code
Upon return from SHOWCB, MODCB, or TESTCB	Error return code[4]	Unpredictable	Unmodified	Unmodified[1]	Unpredictable	General return code
Upon invocation of LERAD or SYNAD exit routines	Recovery action return code	Address of RPL	Unmodified	Unmodified[1]	Return address	Address of exit routine
Upon invocation of other EXLST exit routines	Unpredictable	Address of VTAM-supplied parameter list	Unpredictable	Unpredictable	Return address	Address of exit routine
Upon invocation of RPL-based exit routines	Unpredictable	Address of RPL	Unpredictable	Unpredictable	Return address	Address of exit routine

Notes:

1. Register 13 must indicate the address of an 18-word save area when the macroinstruction is executed.
2. If the operation completed normally, register 15 is set to 0. For some macroinstructions completing normally but with a special condition, register 0 is also set. If an error occurred and the LERAD or SYNAD exit routine has been invoked, registers 0 and 15 contain the values set in them by the exit routine. If an error occurred and no LERAD or SYNAD exit routine exists, VTAM sets register 15 to 4 and places a recovery action return code in register 0 (if the error is that the ACB is not open, register 15 is set to decimal 32 and the RPL request code is set in register 0).
3. When GENCB completes successfully (register 15 is set to 0), register 1 contains the address of the generated control blocks and register 0 contains the length of the control blocks, in bytes.
4. If GENCB, SHOWCB, MODCB, or TESTCB completes unsuccessfully (with register 15 not set to 0), register 1 is unpredictable and register 0 contains an error code (if register 15 is set to 4 or 12) or else is unpredictable.

© Copyright IBM Corp. 2000, 2013

Appendix I. Return codes for manipulative macroinstructions

When the application program receives control from any of the manipulative macroinstructions (GENCB, MODCB, TESTCB, or SHOWCB), register 15 is set to one of the decimal values shown here:

0 The macroinstruction was completed successfully.

If the macroinstruction is GENCB, register 1 contains the address of the control blocks and register 0 contains their total length (in bytes).

4 An error occurred. A return code is placed in register 0 indicating the cause of the error. (See Table 133.)

8 An error occurred. Specifically, an attempt has been made to use the execute form of the macroinstruction to enter a new item in the parameter list. (Only modifications to existing parameter lists are allowed, as explained in Appendix K, "Forms of the manipulative macroinstruction," on page 865.) Register 0 is not set.

When a return code of 4 is placed in register 15, an error return code is placed in register 0. Table 133 explains these error return codes and indicates the manipulative macroinstructions that can return each code. An X in the macroinstruction column means that the return code value applies to that macroinstruction.

Table 133. Manipulative macroinstruction register 0 return codes when register 15 is 4

Decimal value	GENCB	MODCB	SHOWCB	TESTCB	Explanation
1	X	X	X	X	Request type not valid. When the access method processed the execute form, it found that the part of the parameter list that indicates the type of request (GENCB, MODCB, SHOWCB, or TESTCB) had been destroyed.
2	X	X	X	X	Block type not valid. You modified the list form's parameter list. When the access method processed the execute form, it found that the part of the parameter list which indicates the type of control block (ACB, EXLST, RPL, or NIB) had been destroyed.
3	X	X	X	X	Keyword not valid. You modified the list form's parameter list. When the access method processed the execute form, it found that part of the parameter list representing keyword types (for example, FIELDS= and ERET=) had been destroyed.
4		X	X	X	Block not valid. The address specified with the ACB, EXLST, RPL, or NIB keyword did not indicate a valid ACB, EXLST, RPL, or NIB control block, respectively.
5			X	X	Reserved (VSAM only)
6			X	X	Reserved (VSAM only)

© Copyright IBM Corp. 2000, 2013

Table 133. Manipulative macroinstruction register 0 return codes when register 15 is 4 (continued)

Decimal value	GENCB	MODCB	SHOWCB	TESTCB	Explanation
7		X	X		Field nonexistent. You attempted to modify or extract a field from an exit list, but the specified field does not exist. For example, you might have specified MODCB EXLST=EXLST1, LERAD=LERADPGM in order to place a valid address (LERADPGM) in EXLST1's LERAD field. The receipt of this return code means that EXLST1 has no LERAD field; you never specified on an EXLST or GENCB macroinstruction.
8	X				Insufficient main storage. There is not enough main storage in which to build the control block or blocks.
9	X		X		Insufficient program storage. The work area length you indicated with the LENGTH operand was not large enough to build the control blocks (GENCB) or to hold the control block fields (SHOWCB).
10	X	X			No address supplied. You attempted to generate an EXLST entry without specifying an address. For example, coding TPEND= is not valid.
11		X			RPL active. You attempted to modify an RPL that was active (it must be inactive).
12		X			ACB open. You attempted to modify an ACB after it had been opened (the ACB must not be opened when you modify it).
13		X			Reserved (VSAM only)
14	X	X		X	Parameter list not valid. You modified the list form's parameter list. When the access method processed the execute form, it found that the parameter list now indicates mutually exclusive keywords (as though you had, for example, specified BLK=RPL,ECB=ECB1,EXIT=PGM on a GENCB macroinstruction).
15	X		X		Alignment not valid. The work area in your application program does not begin on a fullword boundary.
16	X	X	X	X	Control block not valid (access method not valid). You coded AM=VTAM on the macroinstruction and included one or more parameters valid only for VSAM.
17				X	No internal ECB. TESTCB (IO=COMPLETE) failed because there is no internal ECB in the RPL.
22			X		AM=VTAM was specified and the RPL field's parameter conflicted with the RPLNIB bit status. Either the RPLNIB field was specified and the RPLNIB bit was off, or the RPL ARG field was specified and the RPLNIB bit was on.

Appendix J. Summary of operand specifications

The first figure in this appendix (Table 134) describes all of the operands of the manipulative macroinstructions (GENCB, MODCB, SHOWCB, and TESTCB) that do not involve a particular control block field. The remaining figures deal exclusively with the operands you use to select the control block field or fields to be set, moved, or tested. These figures indicate which manipulative macroinstructions apply for each operand and the types of values that can be coded with each operand.

For example, suppose you are interested in examining an ACB's OFLAGS field. Refer to Table 135 and locate the OFLAGS entry. TESTCB (but not SHOWCB) can be used to examine this field. The OFLAGS operand is coded in a fixed form, in this case, OFLAGS=OPEN.

The "Notation Category" and "Example" columns in Table 135 through Table 138 on page 860 do not apply to the SHOWCB macroinstruction whose control block field name is always coded after the FIELDS keyword (for example, FIELDS=PASSWD).

In the following table an X in the macroinstruction's column means that the operand applies to that macroinstruction.

Table 134. Manipulative macroinstruction operands exclusive of control block operands

Operand keyword	GENCB	MODCB	SHOWCB	TESTCB	Notation category	Example
ACB		X	X	X	Address	ACB=ACB1
AM	X	X	X	X	Fixed value	AM=VTAM
AREA			X		Address	AREA=WORKAREA
BLK	X				Fixed value	BLK=RPL
COPIES	X				Quantity	COPIES=7
ERET				X	Address	ERET=ERRPGM
EXLST		X	X	X	Address	EXLST=EXLST1
FIELDS			X		Fixed value	FIELDS=(ARG, ECB)
LENGTH	X		X		Quantity	LENGTH=132
MF	X	X	X	X	See Appendix K, "Forms of the manipulative macroinstruction," on page 865	MF=(E,PARMLIST)
NIB		X	X	X	Address	NIB=NIB1
RPL		X	X	X	Address	RPL=RPL1
WAREA	X				Address	WAREA=WORKAREA

Table 135. Manipulative macroinstruction operands for ACB fields

Operand keyword	GENCB	MODCB	SHOWCB	TESTCB	Notation category	Example
ACBLEN			X	X	Quantity	ACBLEN=(7)

© Copyright IBM Corp. 2000, 2013

Table 135. Manipulative macroinstruction operands for ACB fields (continued)

Operand keyword	GENCB	MODCB	SHOWCB	TESTCB	Notation category	Example
AM	X	X	X	X	Fixed value	AM=VTAM
APPLID	X	X	X	X	Address	APPLID=AREA
ERROR			X	X	Quantity	ERROR=13
EXLST	X	X	X	X	Address	EXLST=EXLST2
MACRF	X	X			Fixed value	MACRF=EXLST2
OFLAGS				X	Fixed value	OFLAGS=OPEN
PASSWD	X	X	X	X	Address	PASSWD=PASSWD1

Note:

1. An X in the macroinstruction's column means that the operand applies to that macroinstruction.
2. The ACB PARMS keyword is not supported by the manipulative macroinstructions. However, a map can be generated using the DSECT-creating macroinstructions, as described in "Using DSECT-creating assembler instructions and macroinstructions" on page 272.

In the following table an X in the macroinstruction's column means that the operand applies to that macroinstruction.

Table 136. Manipulative macroinstruction operands for EXLST fields

Operand keyword	GENCB	MODCB	SHOWCB	TESTCB	Notation category	Example
DFASY	X	X	X	X	Address	DFASY=(3)
EXLLEN			X	X	Quantity	EXLLEN=(3)
LERAD	X	X	X	X	Address	LERAD=LERTN
LOGON	X	X	X	X	Address	LOGON=LGNRTN
LOSTERM	X	X	X	X	Address	LOSTERM=(6)
NSEXIT	X	X	X	X	Address	NSEXIT=ERRORTN
RELREQ	X	X	X	X	Address	RELREQ=(S,AREA1)
RESP	X	X	X	X	Address	RESP=RESPEXIT
SCIP	X	X	X	X	Address	SCIP=(*,SCIPADR)
SYNAD	X	X	X	X	Address	SYNAD=(*,AREA2)
TPEND	X	X	X	X	Address	TPEND=(S,4(7))

In the following table an X in the macroinstruction's column means that the operand applies to that macroinstruction.

Table 137. Manipulative macroinstruction operands for RPL fields

Operand keyword	GENCB	MODCB	SHOWCB	TESTCB	Notation category	Example
AAREA	X	X	X	X	Address	AAREA=INAREA
AAREALN	X	X	X	X	Quantity	AAREALN=100
ACB	X	X	X	X	Address	ACB=ACB1
AREA	X	X	X	X	Address	AREA=(*,FLWORD)
AREALEN	X	X	X	X	Quantity	AREALEN=132

Table 137. Manipulative macroinstruction operands for RPL fields (continued)

Operand keyword	GENCB	MODCB	SHOWCB	TESTCB	Notation category	Example
ARECLEN	X	X	X	X	Quantity	ARECLEN=(S,QUANT1)
ARG	X	X	X	X	Register- indirect value	ARG=(7)
BRACKET[1]	X	X		X	Fixed value	BRACKET=(BB,NEB)
BRANCH	X	X		X	Fixed value	BRANCH=YES
CHAIN	X	X		X	Fixed value	CHAIN=LAST
CHNGDIR	X	X		X	Fixed value	CHNGDIR=(CMD,NREQ)
CODESEL	X	X		X	Fixed value	CODESEL=STANDARD
CONTROL	X	X		X	Fixed value	CONTROL=QEC
CRYPT	X	X		X	Fixed value	CRYPT=YES
ECB	X	X	X	X	Address	ECB=FULLWORD
EXIT	X	X	X	X	Address	EXIT=EXITRTN
FDBK			X	X	Quantity	FDBK=(4)
FDBK2			X	X	Quantity	FDBK2=128
IBSQAC	X	X		X	Fixed value	IBSQAC=TESTPOS
IBSQVAL	X	X	X	X	Quantity	IBSQVAL=0
IO				X	Fixed value	IO=COMPLETE
NIB	X	X	X	X	Address	NIB=NIB6
OBSQAC	X	X		X	Fixed value	OBSQAC=INVALID
OBSQVAL	X	X	X	X	Quantity	OBSQVAL=(4)
OPTCD[2]	X	X		X	Fixed value	OPTCD=(SYN,SPEC)
POST	X	X		X	Fixed value	POST=SCHED
RECLEN	X	X	X	X	Quantity	RECLEN=32
RESPOND	X	X		X	Fixed value	RESPOND=(NEX,FME)
REQ			X	X	Quantity	REQ=VAL23
RPLLEN			X	X	Quantity	RPLLEN=(7)
RTNCD			X	X	Quantity	RTNCD=(*,X'00',X'03')
RTYPE	X	X		X	Fixed value	RTYPE=(NDFSYN, DFASY)
SEQNO	X	X	X	X	Quantity	SEQNO=(10)
SIGDATA	X	X	X	X	Quantity	SIGDATA=32767
SSENSEI				X	Fixed value	SSENSEI=CPM
SSENSEO	X	X		X	Fixed value	SSENSEO=STATE
SSENSMI			X	X	Quantity	SSENSMI=255
SSENSMO	X	X	X	X	Quantity	SSENSMO=(*,0(12))
STYPE	X	X			Fixed value	STYPE=REQ
USENSEI			X	X	Quantity	USENSEI=4095
USENSEO	X	X	X	X	Quantity	USENSEO=(4)
USER			X	X	Quantity	USER=1024

The RPL PARMS keyword is not supported by the manipulative macroinstructions. However, a map can be generated for all operands by using DSECT-creating macroinstructions, as described in "Using DSECT-creating assembler instructions and macroinstructions" on page 272.

Note:

1. CEB/NCEB are not valid BRACKET operands when using the manipulative macroinstructions.
2. The following OPTCD operands are not applicable when using manipulative macroinstructions:

   LMPEO or NLMPEO, SENSE or NSENSE, HOLD, BUFFLST or NBUFFLST, SONCODE or NSONCODE, USERRH or NUSERRH, RSPQUED or NRSPQUED, CONTCHN or NCONTCHN, UNBIND, QALL or QSESSLIM or QNOTENAB, RESTORE, PERSIST or NPERSIST, PERSESS, BACKUP or NBACKUP, NQN, GNAMEADD or GNAMEDEL, SESSNAME, ENDAFFIN, KEEPSRB, NKEEPSRB

These notes refer to the following table.

Notes:

1. PROC=NEGBIND|NNEGBIND is not valid when using the manipulative macroinstructions. Use the DSECT-creating macroinstructions as an alternative as described in "Using DSECT-creating assembler instructions and macroinstructions" on page 272.
2. An X in the macroinstruction's column means that the operand applies to that macroinstruction.

Table 138. Manipulative macroinstruction operands for NIB fields

Operand keyword	GENCB	MODCB	SHOWCB	TESTCB	Notation category	Example
BNDAREA	X	X	X	X	Address	BNDAREA=BNDADDR
CID			X	X	Register- indirect value	CID=(7)
CON				X	Fixed value	CON=YES
DEVCHAR			X	X	Indirect value	DEVCHAR=(*,0(5))
ENCR	X	X		X	Fixed value	ENCR=SEL
EXLST	X	X	X	X	Address	EXLST=(*,EXLSTADR)
LISTEND	X	X		X	Fixed value	LISTEND=NO
LOGMODE	X	X	X	X	Name	LOGMODE=S3270
MODE	X	X	X	X	Fixed value	MODE=RECORD
NAME	X	X	X	X	Name	NAME=NYCTERM
NIBLEN			X	X	Quantity	NIBLEN=(8)
PROC	X	X		X	Fixed value	PROC=(RESPX,CS)
RESPLIM	X	X	X	X	Quantity	RESPLIM=10
SDT	X	X		X	Fixed value	SDT=SYSTEM
USERFLD	X	X	X	X	Quantity	USERFLD=(9)

A given operand can be coded in different ways, but the number of valid combinations is small. The valid coding combinations for each operand have been grouped under the heading "Notation Category" in Table 134 on page 857 through Table 138 on page 860. The rest of this appendix contains coding instructions for these notation categories. Three of these categories, described in "Address," "Quantity" on page 862, and "Fixed value" on page 863 encompass almost all of the operands. A few operands fall into categories described in "Name" on page 863, "Register-indirect value" on page 863, and "Indirect value" on page 863.

Address

You can code any of the following expressions after the keyword and equal sign:

- Any expression that is valid for an A-type address constant. For example:

```
         MODCB ACB=ACB1,APPLID=NAME1,AM=VTAM

⋮
NAME1    DC     X'07'
         DC     CL7'INQUIRY'
```

- A register number or the label of an EQU instruction for the register, enclosed in parentheses. For example:

```
        L        3,ADRNAME
        MODCB    ACB=ACB1,APPLID=(3),AM=VTAM

⋮
ADRNAME DC       A(NAME1)
```

  Note: This form is prohibited if you are using the "simple" list form of the macroinstruction (MF=L). List forms are explained in Appendix K, "Forms of the manipulative macroinstruction," on page 865.

- An expression of the form (S,*expr*) where *expr* is any expression valid for an S-type address constant. This form of operand is especially useful for gaining access to a control block field with a DSECT. For example, the program has already used GENCB to build an ACB in dynamically allocated storage and has placed the address of the ACB in register 7. The DSECT ACBMAP is used to access the information in MYACB:

```
         LA    5,MYACB
         USING ACBMAP,5
         MODCB ACB=(7),APPLID=(S,APPL1),AM=VTAM

⋮
MYACB    DS    XL108
ACBMAP   DSECT
         DS    XL72
APPL1    DS    A
         DS    XL32
         END
```

  Note: This form is prohibited if you are using the "simple" list form of the macroinstruction (MF=L).

- An expression of the form (*,*expr*) where *expr* is any expression valid for an S-type address constant. The address specified by *expr* is indirect; that is, it is the address of a fullword that contains the operand. For example, the program determines which APPLID address is used, and primes register 5 with the appropriate displacement into APPLIST:

```
L     7,APPLIST(5)
MODCB ACB=ACB1,APPLID=(*,0(7)),AM=VTAM
⋮
APPLIST  EQU   *
         DC    A(APPL1)
         DC    A(APPL2)
```

Quantity

You can code any of the following expressions after the keyword and equal sign:

- A decimal number, or an expression that you have equated to a decimal number. For example:

  ```
  TESTCB ACB=ACB1,ERROR=13,AM=VTAM
  ```

- A register number, or the label of an EQU instruction for the register number, enclosed in parentheses. For example:

  ```
          L     5,TESTVAL
          TESTCB ACB=ACB1,ERROR=(5),AM=VTAM

  ⋮
  TESTVAL  DS    F      TESTVAL SET DURING PROGRAM
  *                     EXECUTION
  ```

  Note: This form is prohibited if you are using the "simple" list form of the macroinstruction (MF=L).

- An expression of the form (S,*expr*) where *expr* is any expression valid for an S-type address constant. This form is especially useful for gaining access to a control block field with a DSECT. For example, the program has already used GENCB to build an ACB in dynamically allocated storage, and has placed the address of the ACB in register 7. A temporary work area, MYACB, contains the information with which the contents of the ERROR field in the ACB pointed to by register 7 are compared. The DSECT ACBMAP is used to access the information in MYACB.

  ```
          LA    5,MYACB
          USING ACBMAP,5
          TESTCB ACB=(7),ERROR=(S,ERR1),AM=VTAM

  ⋮
  MYACB    DS    XL108
  ACBMAP   DSECT
           DS    XL49
  ERR1     DS    X
           DS    XL58
           END
  ```

  Note: This form is prohibited if you are using the "simple" list form of the macroinstruction (MF=L).

- An expression of the form (*,*expr*) where *expr* is any expression valid for an S-type address constant. The address specified by *expr* is indirect; that is, it is the address of a fullword that contains the quantity for the operand. For example, the program has determined which ERROR value is tested and has primed register 5 with the appropriate displacement into ERRORLST:

  ```
          TESTCB ACB=ACB1,ERROR=(*,ERLST(5)),AM=VTAM

  ⋮
  ERLST    EQU   *
  BADNAME  DC    F'90'
  BADPSWD  DC    F'152'
  ```

Fixed value

You can code only the expressions that are specified in the macroinstruction description. For example:

```
GENCB BLK=ACB,MACRF=NLOGON,AM=VTAM
```

Name

You can code any of the following expressions:

- 1–8 EBCDIC characters. For example:

  ```
  TESTCB NIB=NIB1,NAME=TERM0003,AM=VTAM
  ```

- An expression of the form (*,*expr*) as explained in the preceding section. The address specified by *expr* is indirect; that is, it is the address of a doubleword containing the name. The name must be left-aligned and padded to the right with blanks if it does not occupy the entire doubleword. For example:

  ```
          L     7,NAMEPOOL
          ST    7,NEWNAME+4
          MODCB NIB=NIB1,NAME=(*,NEWNAME),AM=VTAM

  ⋮
  NEWNAME  DC    CL8'TERM'
  NAMEPOOL DC    CL4'0001'
  ```

Register-indirect value

You can code any of the following expressions:

- A register number or label of an EQU instruction for the register number, enclosed in parentheses. For example:

  ```
          MODCB RPL=RPL1,ARG=(REG5),AM=VTAM

  ⋮
  REG5     EQU   5
  ```

  Note: This form is prohibited if you are using the "simple" list form of the macroinstruction (MF=L).

- An expression of the form (*,*expr*) as explained in the preceding section. The address specified by *expr* is indirect; that is, it is the address of a fullword that contains the value. For example:

  ```
          MODCB RPL=RPL1,ARG=(*,NEWCID),AM=VTAM

  ⋮
  NEWCID   DS    F       NEWCID SET DURING PROGRAM
  *                      EXECUTION
  ```

Indirect value

You can code an expression of the form (*,*expr*) only as explained in the preceding section. The address specified by *expr* is indirect; it is the address of a doubleword that contains the value. For example:

```
         TESTCB NIB=NIB1,DEVCHAR=(*,DEVMASK),AM=VTAM
:
DEVMASK  DS    D       DEVMASK SET DURING PROGRAM
*                      EXECUTION
```

Appendix K. Forms of the manipulative macroinstruction

The standard form of a manipulative macroinstruction expands at assembly time into (1) nonexecutable code that represents the parameters you specified on the macroinstruction and (2) executable code that causes the access method to be entered when the macroinstruction is executed. The nonexecutable code, called the parameter list, is assembled at the point in your application program where the macroinstruction appears.

Various alternative forms of the manipulative macroinstructions cause the assembler to:

- Build the parameter list where the macroinstruction appears in your source code, but assemble no executable code ("simple list" form)
- Assemble code that builds the parameter list at a location of your selection, but assemble no executable code that causes the access method to be entered ("remote list" form)
- Assemble code that builds the parameter list at a location of your selection and assemble the code that causes the access method to be entered (generate form)
- Assemble code that modifies a parameter list and cause the access method to be entered during program execution (execute form).

Table 139 summarizes the actions of these various forms. It also indicates the types of programs that would use each form and how the MF operand is used.

Table 139. Forms of manipulative macroinstructions

Form	During assembly	During execution	Useful for	Coded with
Standard	Parameter list built where macroinstruction appears in source code	Access method entered	Nonreentrant programs that are not sharing or modifying parameter lists	No MF operand
Simple List	Parameter list built where macroinstruction appears in source code	No executable code (execute form required)	Nonreentrant programs that are sharing or modifying parameter lists	MF=L
Remote List	Assemble code to build parameter list at a location you specify	Parameter list built, but access method not entered (execute form required)	Reentrant programs that are sharing or modifying parameter lists	MF=(L,*address*[,*label*])
Generate	Code assembled to: 1. Build parameter list at a location you specify 2. Enter the access method	Parameter list built and access method entered	Reentrant programs are not sharing or modifying parameter lists	MF=(G,*address*[,*label*])

© Copyright IBM Corp. 2000, 2013

Table 139. Forms of manipulative macroinstructions (continued)

Form	During assembly	During execution	Useful for	Coded with
Execute	Code assembled (where macroinstruction appears in the source code) to modify the parameter list whose address you supply	Parameter list modified and the access method entered	Programs using the list form	MF=(E,*address*)

As indicated in Table 139 on page 865, the various alternative forms of the manipulative macroinstructions are designated with the MF operand.

The MF operand for the list form of any manipulative macroinstruction is coded as follows:

MF={L or (L,address[,label])}

L Indicates that this is the list form of the macroinstruction. If you code just MF=L (simple list form), the parameter list is assembled in place. If you code MF=(L,*address*) (remote list form), the parameter list is built during program execution at the specified location.

address
Indicates the location where you want the parameter list to be built during program execution. This area must begin on a fullword boundary and, if your program is reentrant, must be in dynamically allocated storage. Because the assembler builds executable codes that in turn builds the parameter list, the macroinstruction must be in the executable portion of your program—that is, not treated as a program constant.

You can code this address in any of the forms of the "address" notation category (described in Appendix J, "Summary of operand specifications," on page 857).

label
This is a unique name that is used as a label for an assembled EQU instruction. During program assembly, the assembler equates this label to the length (in bytes) of the parameter list that is built during program execution. You can use this label to assure that you are obtaining enough dynamically allocated storage to hold the parameter list.

When coding *label*, follow the same rules that apply to any label for an assembler instruction.

List form example:

```
LA       10,PLISTLEN    OBTAIN LENGTH OF PARAMETER LIST
GETMAIN  R,LV=(10)      OBTAIN STORAGE FOR PARAMETER LIST
LR       5,1            SAVE STORAGE ADDRESS
TESTCB   RPL=RPL1,CONTROL=DATA,AM=VTAM,MF=(L,(5),PLISTLEN)
```

The MF operand for the "generate" form of any of the manipulative macroinstructions is coded as follows:

MF=(G,address[,label])

G Indicates that this is the generate form of the macroinstruction.

address
Indicates the location where you want the parameter list to be built during

program execution. Presumably, this is in dynamically allocated storage. In both manner of use and manner of coding, this address is identical to the address described in the preceding list form discussion.

label
Indicates the label to be used on an EQU instruction for the length of the parameter list. The function of the *label* operand and its rules for coding are identical to those described in the preceding list form discussion.

Generate form example:

```
LA       10,PLISTLEN     OBTAIN LENGTH OF PARAMETER LIST
GETMAIN  R,LV=(10)       OBTAIN STORAGE FOR PARAMETER LIST
LR       5,1             SAVE STORAGE ADDRESS
GENCB    BLK=RPL,AM=VTAM,MF=(G,(5),PLISTLEN)
```

The MF operand for the execute form of any of the manipulative macroinstructions is coded as follows:

MF=(E,address)

E Indicates that this is the execute form of the macroinstruction.

address
Indicates the location of parameter list to be used by the access method.

The execute form allows you to modify the parameter list between the generation of that parameter list and the invocation of the access method routines that use the parameter list. Only the execute form provides a means for you to modify the parameter list after it has been built.

The optional operands you specify on the execute form of a particular macroinstruction are converted by the assembler into code that modifies a parameter list during execution. This code can only modify—and not expand—the parameter list. If the parameter list is actually a list form (as is typically the case), never refer to a control block field in an execute form that you did not specify in the list form. If you fail to observe this rule, and thereby attempt to expand the parameter list, the execute form is not processed successfully, and a return code of 8 is posted in register 15.

Execute form example:

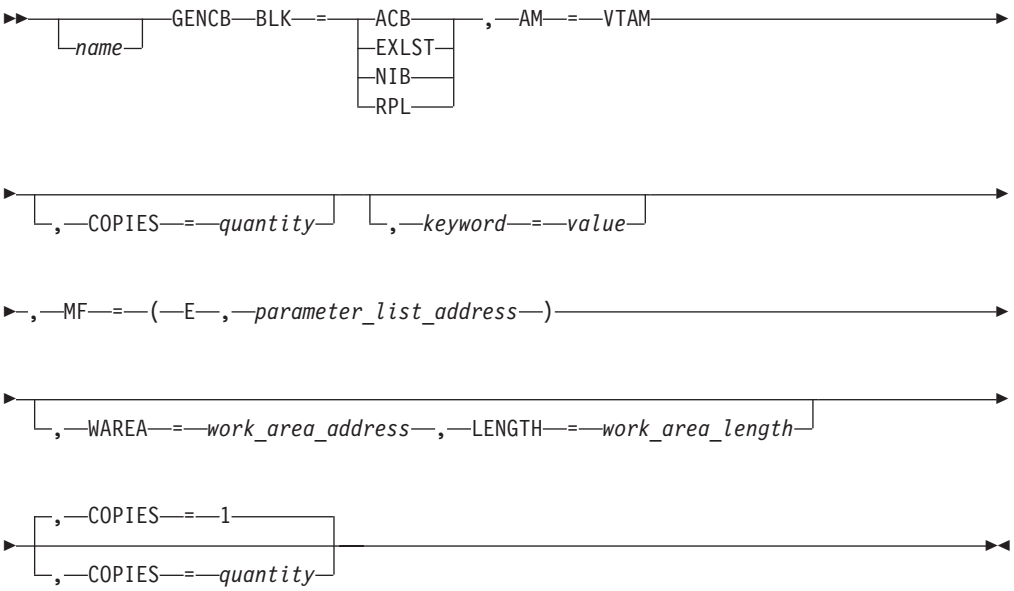
```
EFORM    MODCB EXLST=EXLST1,LERAD=(3),AM=VTAM,                        C
               MF=(E,LFORM)

⋮
LFORM    MODCB EXLST=0,LERAD=0,MF=L,AM=VTAM
```

Optional and required operands

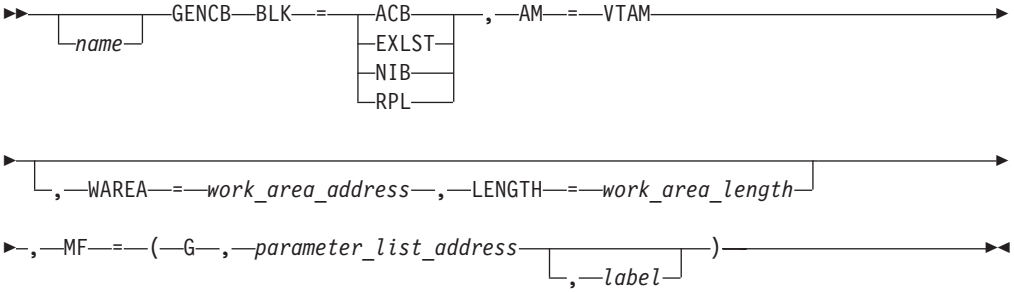
Operands that are required in the standard form of the manipulative macroinstructions can be optional in the list, generate, or execute forms, or prohibited in the execute form. The meanings of the operands, however, and the notation used to express them, are the same. The following syntax diagrams indicate which operands are required and which are optional for each form of each manipulative macroinstruction. Any operand that does not appear in a syntax diagram for a particular form is prohibited.

Optional and required operands for the alternative forms of GENCB

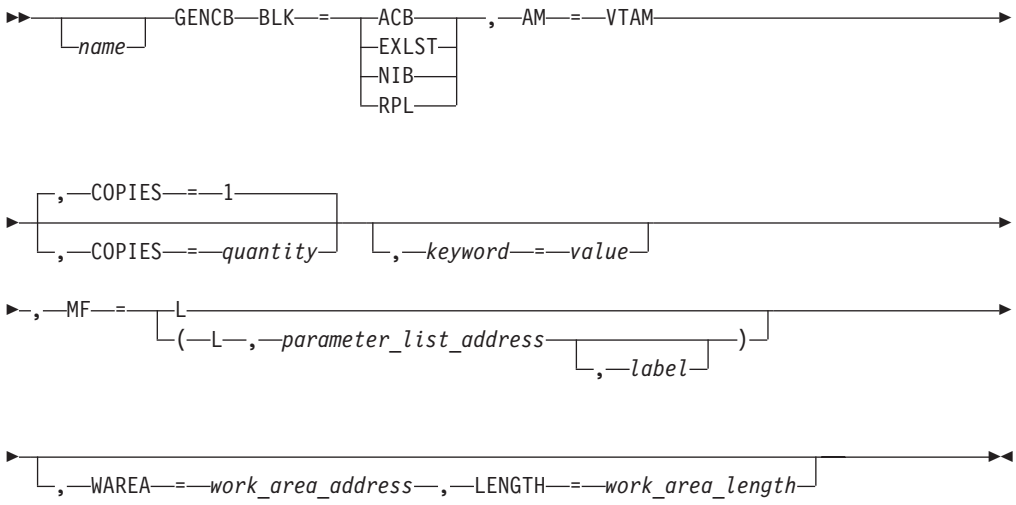
GENCB execute form



CB generate

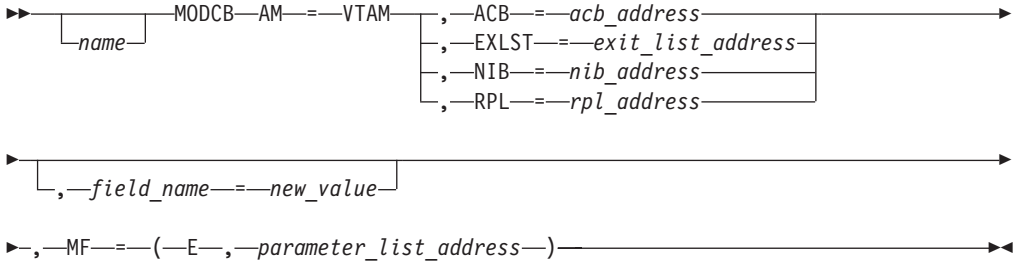


GENCB list form

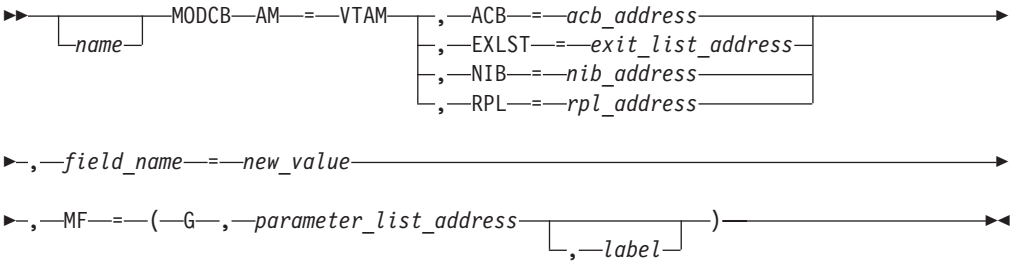


Optional and required operands for the alternative forms of MODCB

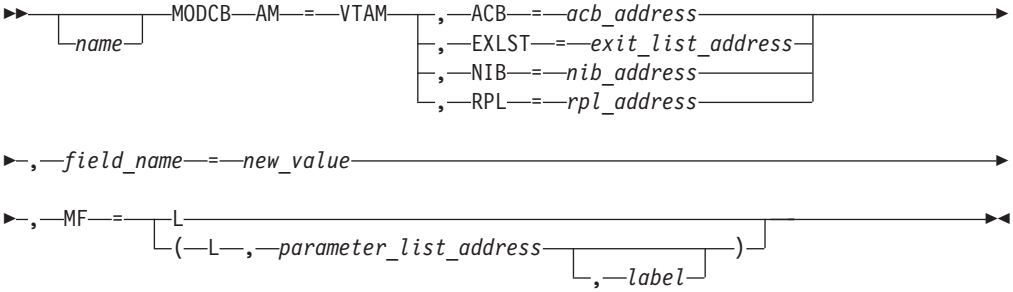
MODCB execute form



MODCB generate form

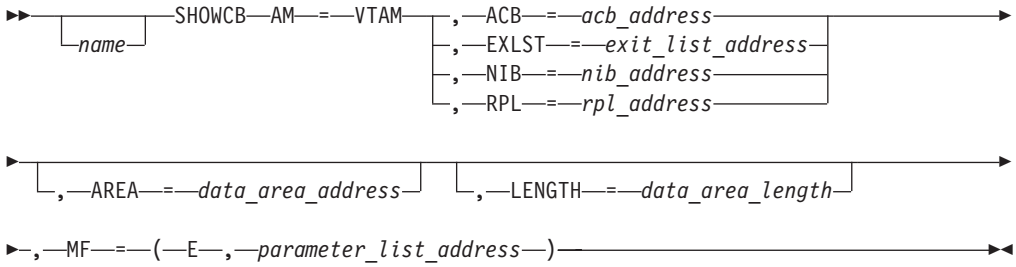


MODCB list form

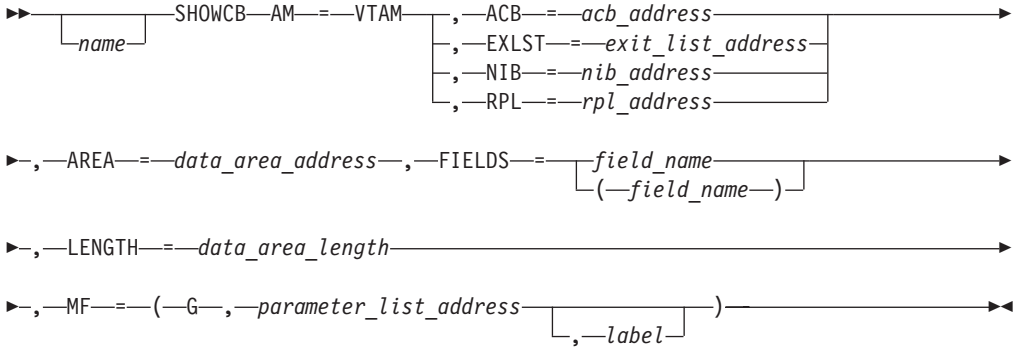


Optional and required operands for the alternative forms of SHOWCB

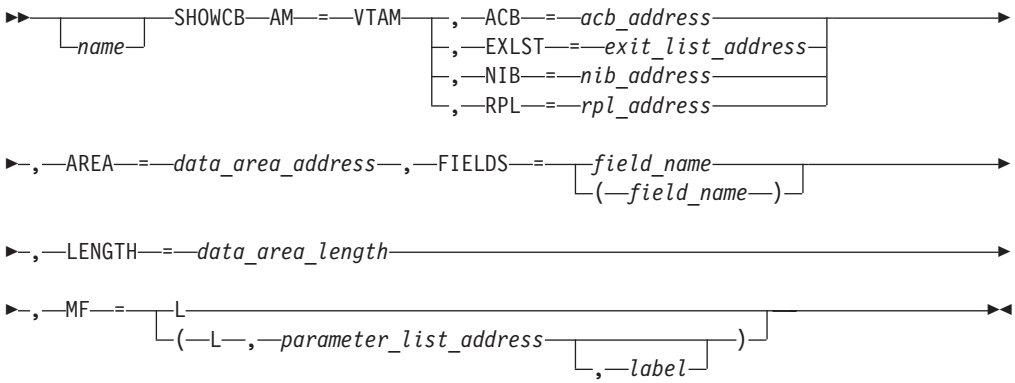
SHOWCB execute form



SHOWCB generate form

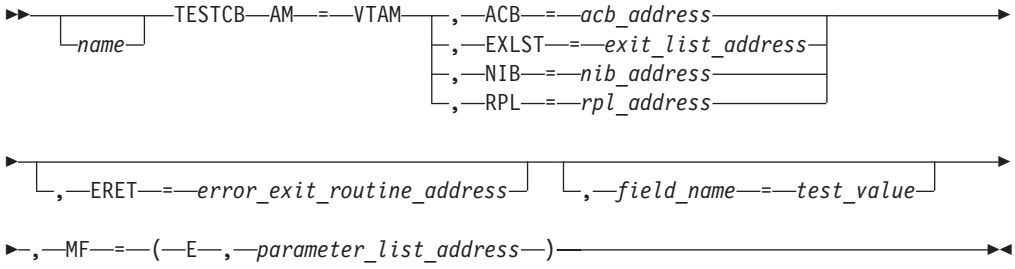


SHOWCB list form

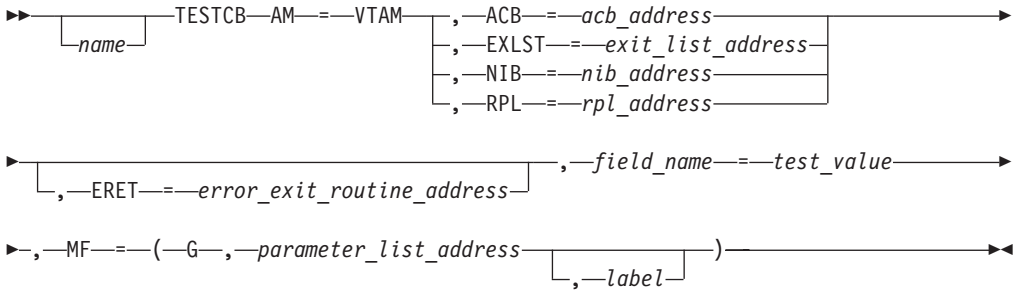


Optional and required operands for the alternative forms of TESTCB

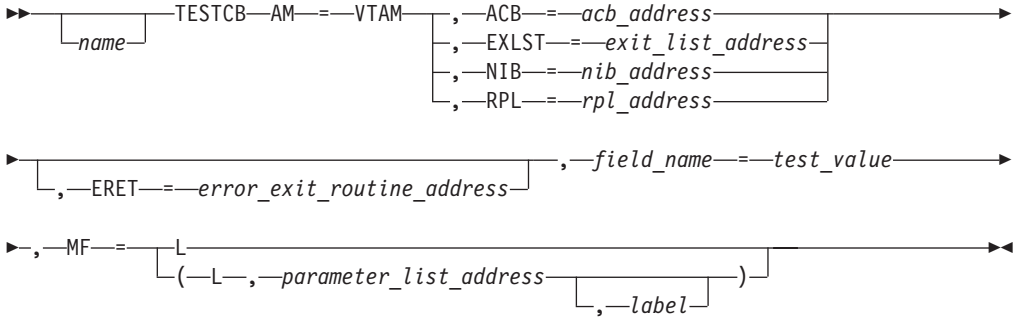
TESTCB execute form



TESTCB generate form



TESTCB list form



Appendix L. Program operator coding requirements

This appendix describes how to write the program operator portion of a VTAM application program using the SENDCMD and RCVCMD macroinstructions. See "RCVCMD—Receive a message from VTAM" on page 463 for more information on how to receive a message from VTAM. See "SENDCMD—Send a VTAM operator command to VTAM" on page 529 for more information on how to send a VTAM operator command to VTAM.

Defining a program operator

A VTAM application program can be authorized to issue VTAM operator commands to:

- Display the status of the network
- Control the status of the network
- Receive messages from VTAM
- Reply to VTAM messages.

Such an application program is called a ***program operator***, or a ***program operator application*** (POA), and permits a user to:

- Enter operator commands from any LU in the network (for example, from a terminal)
- Monitor and control elements in the network at program execution speed
- Specialize network control by dividing the network among several application programs
- Define specialized commands (for example, to display the status of the entire network with a single command)
- Reformat replies received for VTAM commands (for example, to reformat the status display of a part of the network to fit on a 3270 display screen)
- Coordinate control of different domains in a multiple-domain network.

A program operator can:

- Issue a DISPLAY, MODIFY, or VARY command by using a SENDCMD macroinstruction. The format of the command is the same as though it were issued from the system console.
- Receive messages from VTAM by using a RCVCMD macroinstruction.
- Reply to a VTAM message by using a SENDCMD macroinstruction to send a REPLY command.

Figure 183 on page 876 shows how the system console operator and a program operator send VTAM operator commands and receive VTAM operator messages to control the VTAM domain. OPNDST, SEND, RECEIVE, and CLSDST are examples of session-establishment and communication macroinstructions.

© Copyright IBM Corp. 2000, 2013

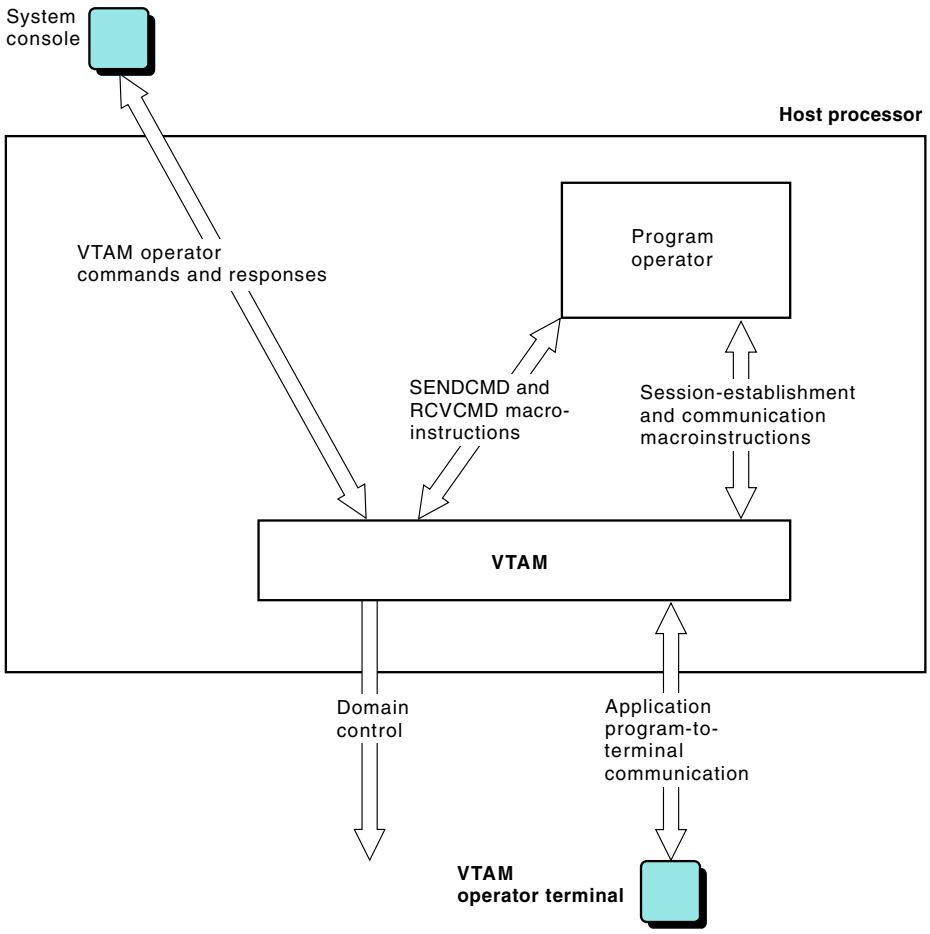


Figure 183. VTAM operator control of a VTAM domain

The NetView program provides many of the facilities listed in the preceding section while also providing a base for communication network management functions.

In a multiple-domain network, program operators in different domains can communicate by means of VTAM macroinstructions to allow a program operator or an operator at a terminal to monitor and control elements in other domains. Figure 184 on page 877 shows an example of how a multiple-domain network can be controlled using two program operators.

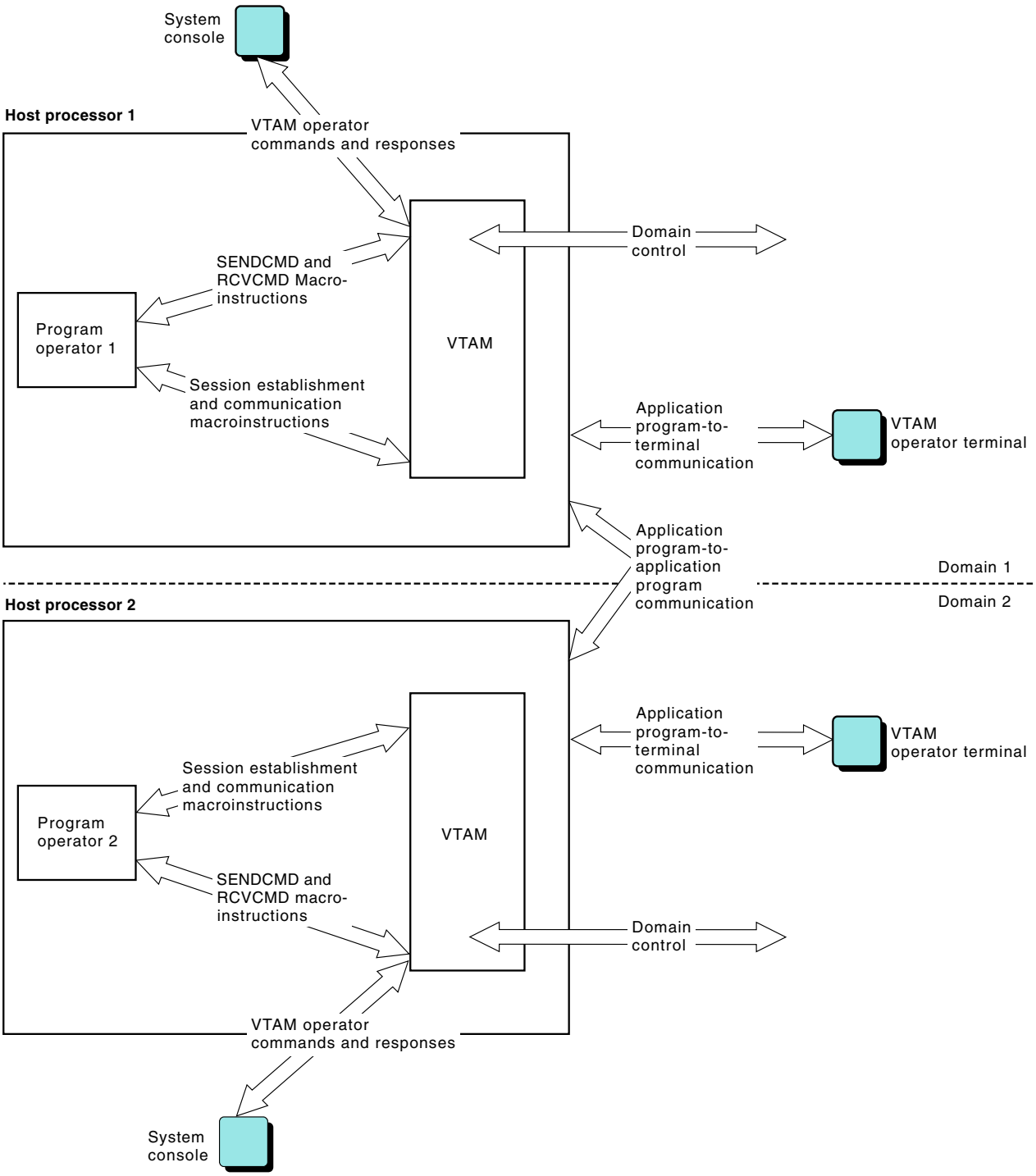


Figure 184. VTAM operator control of a multiple-domain VTAM network

To activate an element in domain 2:

1. An operator at a terminal in domain 1 sends a command to program operator 1.
2. Program operator 1 determines that the command is for domain 2 and sends it to program operator 2.

3. Program operator 2 sends the command to VTAM (using a SENDCMD macroinstruction), where the command is processed.
4. Program operator 2 sends any replies back to program operator 1, which, in turn, sends them to the terminal.

Another way to activate an element in domain 2 is to have a terminal establish a session directly with program operator 2.

Two levels of authorization for an application program determine the type of messages that the program can receive. A program can be authorized to:

- Receive only messages that are in reply to VTAM operator commands issued by a program operator (solicited messages). Such an application program is called a ***secondary program operator*** (SPO).
- Receive messages whether they are in reply to VTAM operator commands that the program operator issued (solicited messages) or are as a result of unexpected events in the network (unsolicited messages), for example, losing contact with a terminal. Such an application program is called a ***primary program operator*** (PPO).

Only one primary program operator can be active at a time; however, one or more secondary program operators can also be active.

Note: If a primary program operator is active, the systems' operations console does not receive unsolicited messages.

Authorizing a program operator

A program operator must be authorized to use the SENDCMD and RCVCMD macroinstructions during VTAM definition. The APPL definition statement for the application program specifies one of the following:

- AUTH=PPO: Authorizes the application program to send VTAM operator commands to VTAM and to receive both solicited and unsolicited VTAM messages.
- AUTH=SPO: Authorizes the application program to send VTAM operator commands to VTAM and to receive only those VTAM messages that are in reply to those commands.
- AUTH=NOPO: Prohibits the application program from issuing the SENDCMD and RCVCMD macroinstructions. NOPO is the default value for AUTH.

Method for writing a program operator

An application program can use the SENDCMD macroinstruction to issue VTAM operator commands to VTAM and the RCVCMD macroinstruction to receive messages from VTAM. For a description of the SENDCMD macroinstruction, see “SENDCMD—Send a VTAM operator command to VTAM” on page 529. For a description of the RCVCMD macroinstruction, see “RCVCMD—Receive a message from VTAM” on page 463. Terminals in the network can be in session with any program operator, permitting VTAM commands to be entered from any terminal in the network either in this domain or in another.

If a program operator fails, VTAM reroutes to the system console any commands still outstanding that require a reply and any unsolicited VTAM messages. All other messages are discarded.

The program operator interface allows flexibility in controlling and managing the network. Careful planning is necessary to decide how much or how little programmed control to use.

The following steps describe a suggested approach to coding and developing a program operator:

1. Write a switching program that performs a minimum amount of VTAM operator control to permit an operator at a terminal in the network to enter commands and receive VTAM messages. This program (1) relieves the system console operator of a part of the responsibility for monitoring and controlling the VTAM network and (2) provides a base for further analysis and extension of network control.
2. Analyze the daily command procedures used by the operator at the terminal. Some of these procedures can be written into the program operator. For example, the operator might be entering a series of repetitive, time-consuming commands that can be handled by the program operator, or the replies to a series of commands can be summarized or handled by the program operator.
3. Change and extend the program operator or add program operators. New functions can be needed to aid the terminal operators or to support changes to the network configuration. The program can be expanded gradually until the desired amount of automatic operator control is implemented.
4. For a multiple-domain network, add application program-to-application program communication capability (between VTAM domains) to allow operator commands and replies to be exchanged between domains.

VTAM operator commands

VTAM operator commands entered by a program operator have the same format and effect as commands entered at the system console. The VTAM operator commands supported are the VARY, DISPLAY, and MODIFY commands. The REPLY command is also supported to answer VTAM messages that require a reply.

A program operator can issue all commands except the VTAM START and HALT commands; they can be issued only by the system operator at a system console. The MODIFY QUERY command can be issued only from a program operator application. For a description of MODIFY QUERY, see "POA communication with tuning facility using the MODIFY QUERY COMMAND" on page 889.

For information about each VTAM operator command that can be issued by the system operator, refer to z/OS Communications Server: SNA Operation. For additional information about VTAM operator commands that use the unformatted system services (USS) facility, refer to the z/OS Communications Server: SNA Resource Definition Reference.

A program operator application (POA) can open its ACB indicating that can receive the command complete message (IST1746I). VTAM issues this message as the last message in response to a VTAM operator command. After receiving this message, the POA continues processing with the next command. A POA must set this capability in vector list ISTVACBV-ISTVAC81. The flag is VAC81EOM. For further information, see the "Application-ACB vector list (ISTVACBV)" on page 789.

The operating procedures supplied by the system programmer should not depend on internal VTAM execution sequences; that is, VTAM commands entered at nearly

the same time could be processed in parallel, and the order of processing of the commands should not be assumed to be the same as the order in which the commands are entered. In particular, if one command depends on the successful completion of another, the dependent command should not be entered until the successful completion of the first command is confirmed by a VTAM message.

Operational characteristics

A hard-copy log of the VTAM system console is not directly available to the program operator. A program operator, however, can:

- Request that a copy of each VTAM command entered at the system console and each VTAM message sent to the system console be sent to the primary program operator log. The operator can see them at the primary program console in the form of unsolicited messages if the primary program operator implements a log. See the description of the MODIFY PPOLOG command in z/OS Communications Server: SNA Operation or the PPOLOG option of the START command in the z/OS Communications Server: SNA Resource Definition Reference.
- Maintain a copy of each VTAM command issued by the program operator and each message received by the program operator by writing them to a printer or recording them on an auxiliary storage device.

An application program must control the display format of a terminal to be used as a VTAM operator terminal.

To have the same capability as a VTAM operator at a system console, a program operator must communicate with a system console operator to request certain operating system services (such as starting application programs). When there is not an active PPO, the messages or commands go to the system operator.

If a secondary program operator or a system console operator issues a MODIFY USERVAR command, the resulting solicited USERVAR messages are sent to both the originator of the command and the primary operator. The primary operator receives them in the form of unsolicited messages. If the command was issued by a secondary program operator and an active primary program operator is not available, the messages go to the system console as well as to the secondary program operator.

Note: The program operator can receive only VTAM messages. Operating system messages, even though related to VTAM, cannot be received by a program operator. This restriction does not apply to the system console operator.

In addition, it might be necessary to coordinate the operation of program operators and the system console operator.

Messages rerouted to a PPO

A message that is sent to PPOLOG or percolated to a PPO from the system console or from an SPO appears in the same format in which the message is defined in the appropriate USS table.

If the message is defined in a user-written table (specified on the USSTAB operand of the APPL definition statement or on the MODIFY TABLE command), VTAM presents the message to the PPO in the format in which the message is defined in the user-written table.

If the message is not defined in a user-written table and if SSCPFM=USSNOP is specified on the APPL definition statement that defines the PPO, VTAM presents the message to the PPO in the format in which the message is defined in the IBM-supplied USS table, ISTINCNO.

If the message is not defined in a user-written table and if SSCPFM=USSPOI is specified on the APPL definition statement that defines the PPO:

- VTAM presents the message to the PPO in the format in which the message is defined in the IBM-supplied USS table, ISTCFCMM.
- If the message is not defined in ISTCFCMM, VTAM presents the message to the PPO in the format in which the message is defined in the IBM-supplied USS table, ISTINCNO.

Programming requirements

Table 140 lists some of the considerations and questions that relate to writing a program operator. This list supplements the list of programming considerations found in Chapter 11, "Programming for the IBM 3270 Information Display System," on page 327.

Table 140. Some considerations that affect the coding of a program operator

Program function	Programming considerations
Handling VTAM operator commands and VTAM messages	• Can a timer routine perform predefined actions while the network is active? • How is the program operator going to examine and react to messages received from VTAM? • Are there situations that require human intervention? • Should displays be processed or reformatted prior to sending them to the terminal? • Should commands be processed before sending them to VTAM? • Are there other services that the program operator can provide? – Operator-to-operator communication – Hard-copy log (the hard-copy log available to the system console is not available to the program operator)
Opening the program operator	• How many ACBs are to be opened? • Are separate EXLSTs to be used if more that one ACB is used?
Establishing sessions to handle VTAM operator commands	• When a terminal logs on, how is authorization for the terminal and the operator to be determined? – Is the terminal or operator authorized to use the program operator? – What type of authorization is to be used (for example, a password as a part of the logon message)? • How does the program operator determine the terminal's authorization? – Can this terminal receive all VTAM messages? – Can this terminal enter only certain commands?

Table 140. Some considerations that affect the coding of a program operator (continued)

Program function	Programming considerations
Correlating messages and responses	• Is a table needed to associate messages and responses with a specific terminal? • What type of messages are received by the program operator (solicited or unsolicited)? **Note:** The message-suppression level specified by the system console operator or a program operator determines the levels of VTAM operator messages received by any VTAM operator. • How is the ID field of the header to be used? • How is the header set up to send the message to VTAM? • If a reply is requested, does the program operator – Generate the reply? – Pass the messages to a terminal? – Keep track of the message until a reply is sent? • If the terminal cannot receive the message, should it be sent to the system console, sent to another terminal, or discarded?
Outage of the session with the terminal operator	• What happens to messages that are intended for the terminal? – Should they be sent to the system console? – Should they be sent to another terminal? – Should they be discarded? • If all the terminals are lost, what should the program operator do? – Terminate? – Request operator intervention? – Wait for other terminals to log on?
Closing the program operator	• In what order are the ACBs in the application program to be closed? • Are outstanding messages to be sent to the system console? • What happens if the first CLOSE fails? • Should any additional messages be processed before issuing a second CLOSE?

Orderly closing of a program operator

When a program operator is about to be closed, unreceived VTAM messages can still be queued for it. Issuing the CLOSE macroinstruction results in a return code (76), which indicates there are messages waiting. At this point, before a second CLOSE macroinstruction is issued, the program operator can issue only RCVCMD macroinstructions or SENDCMD macroinstructions with a REPLY command. This permits the program operator to receive the messages that are still waiting and to answer any that require a reply.

When all the messages have been received, a return code for a RCVCMD macroinstruction indicates that there are no more messages waiting. A second CLOSE macroinstruction then completes normally. Because VTAM does not queue these RCVCMD macroinstructions, it is recommended that they be issued with NQ specified in the OPTCD operand. VTAM accepts the RCVCMD macroinstruction with the Q option; however, when there are no messages remaining on the queue, the RCVCMD macroinstruction completes with (RTNCD,FDB2)=(X'14',X'70'). If NQ is specified, the request completes with (RTNCD,FDB2)=(X'00',X'06') when there are no more messages queued. Alternatively, a program operator can issue a second CLOSE macroinstruction without receiving the queued messages. The second

CLOSE causes the outstanding messages that do not request a reply to be discarded. Messages that require a reply and have not been received or replied to are sent to the system console.

Limiting VTAM messages queued to a program operator

A program operator application (POA) has the responsibility of issuing enough RCVCMDs to handle the messages that it is expected to receive. In the case of an error in a POA, messages might arrive much faster than they are received. Consequently, a limit can be set on the number of messages queued to a POA. Should such an error occur, setting this limit allows VTAM to do the following:

- Reduce the possibility of VTAM abnormally terminating
- Allow information to be gathered to determine the source of the problem.

After the limit is reached, no further messages are queued and SENDCMDs from that POA are rejected until all messages on the queue have been removed. This limit can be specified with the POAQLIM parameter on the application (APPL) definition statement and only when AUTH=SPO or AUTH=PPO is specified. POAQLIM is not a required parameter, and any decimal value 1–2147483647 is accepted. Refer to the z/OS Communications Server: SNA Resource Definition Reference for the formula used to calculate POAQLIM. If a limit is not coded, there is no limit to the number of messages queued.

If a limit is coded, the POA recognizes a RPL return code of X'0C' and RPL feedback of X'0E' for a SENDCMD. When this value is returned, the POA issues enough RCVCMDs to clear the message queue before issuing any more SENDCMDs. If these RCVCMDs are issued with OPTCD=NQ (no queue), the POA knows it has issued enough when a RCVCMD is returned because no more messages exist to be received (RPL return code of X'00' and RPL feedback of X'06'). When the POA queue limit is reached, a “limit reached” message (IST983E) displays on the operator console. The message is highlighted and stays on the screen until cleared by the operator. It is suppressed so that only one occurrence per application displays within a 30-second period. Each occurrence is traced. When this message displays and the POA cannot handle the problem, you should take a dump and cancel the POA to clear the message queue. The type of messages queued to the POA should help determine the problem with the POA.

After the limit is reached, no further messages are queued until all messages on the queue have been received. Messages to be queued after the limit is reached are discarded except for unsolicited messages, which are sent to the operator console.

Partial groups of messages are not sent. If a solicited message causes the queue limit to be exceeded, the message or its entire group is discarded. If an unsolicited message causes the queue limit to be exceeded, the message or its entire group is rerouted to the system console. If a message requiring a reply causes the queue limit to be exceeded, the message is rerouted to the system console.

Data exchanged between a program operator and VTAM

The area pointed to by the AREA operand of the RPL associated with the program operator macroinstructions contains the data that is exchanged between the program operator and VTAM. The RPL associated with the SENDCMD macroinstruction points to the VTAM operator commands that are sent from the program operator to VTAM. The RPL associated with the RCVCMD

macroinstruction points to the message received by the program operator from VTAM. Every time data is exchanged in this way, the sender of the data appends a header to the beginning of the data. The header tells where the data is coming from, the status of the data (such as whether a reply is requested), and the identification number of the data.

Header

Every time VTAM sends a message to a program operator or every time a program operator sends a VTAM operator command to VTAM, a 4-byte header is appended to the data. The header has the following form:

X'00'	Status	ID number
1 byte	1 byte	2 bytes

X'00'
Is a required part of the header and is always present. Either it is provided by VTAM in data that the program operator receives, or it must be supplied by the program operator in data that is sent to VTAM.

Status
Indicates the sender of the message or command, whether a reply is required, and the type of message if it is part of a multi-line message.

When VTAM sends an operator command to the program operator, the program operator sets the following bits:

Status field bit setting 0123 4567	Meaning
....1	The command originates from a program operator.
....0	The command originates from VTAM
.... ..0.	VTAM is not to return a reply to the message issued as a result of a SENDCMD macroinstruction.
.... ..1.	VTAM returns an appropriate reply to the message issued as a result of a SENDCMD macroinstruction.
xxxx xx..	Reserved.

When the program operator receives a message from VTAM, VTAM sets the following bits:

Status field bit setting 0123 4567	Meaning
....0	This message is an unsolicited message. It originates from VTAM and is not in reply to a VTAM operator command. The ID in this header was generated by VTAM.
....1	This message is sent in reply to a VTAM operator command previously sent to VTAM by the program operator. The ID number in this header was generated by the program operator when the command was sent.
.... ..0.	A reply is not required.
.... ..1.	A reply is required. A reply ID is present in the message. Use the SENDCMD macroinstruction to issue a REPLY command to VTAM.
.... 00..	This message is not a copy of a VTAM operator command or a copy of a VTAM message sent to the system console.
.... 01..	This is an unsolicited message containing a copy of a VTAM operator command entered from the system console.

Status field bit setting 0123 4567	Meaning
.... 10..	This is an unsolicited message containing an unsuppressed message that had been sent to the system console.
.... 11..	This is an unsolicited message containing a copy of a VTAM-solicited, suppressed message that had been sent to the system console.
0000	This is not a multi-line message.
0001	This is the ***control line*** of a multi-line message. The control line is always the first line of a multi-line message and normally contains a message title. The control line should remain static during framing operations on a display console such as the 3270 (provided it is displayed in an out-of-line display area).
0010	This is a ***label line*** of a multi-line message. The label line might be the first line of a multi-line message if there is no control line. If there is a control line, the label line always follows it. There can be more than one label line, but they cannot be interspersed with other types of lines. Label lines usually contain message header information and remain static with the control line.
0100	This is a ***data line*** of a multi-line message. The data line always follows either a control line or a label line, if present. The data line contains the information intended for the VTAM operator and, unlike the control line and label line, is paged during framing operations.
1000	This is the ***end line*** of a multi-line message. The end line indicates that the previous data line is the last line of text that is to be passed to the VTAM program operator. If data is included in the end line, it is ignored.
1100	This is a combined data and end line of a multi-line message. It indicates that this is the last line of text that is passed to the program operator.

ID number
Any number 0–65535 (X'0000'–X'FFFF') can be specified by VTAM when it sends an unsolicited message to the program operator. The program operator, when sending a command or reply to VTAM, can select and interpret the identification number in any way it finds meaningful. When the program operator sends a command to VTAM, the program operator must set the identification number; if a reply is requested, VTAM returns the same number with the reply. If VTAM sends a message to the program operator and the message is not a reply to a command, VTAM sets the identification number. Using the identification number, the program operator can correlate solicited messages received from VTAM with the appropriate commands issued by the program operator.

Note: An IBM-supplied DSECT (ISTDPOHD) is available and can be used in creating or interpreting the header. Refer to "Format and DSECT of the message and command header" on page 888 for more information.

Data received from VTAM

A program operator can use the RCVCMD macroinstruction to receive messages from VTAM. The program operator receives the message data with a 4-byte header appended to it. The format of the message data is determined by the status byte of the header:

- If this is a solicited message that does not require a reply (bits 6–7) **or** if this is a copy of VTAM message that had been sent to system console (bit 4), the message is in the form:

Header	Message ID	Message text
4 bytes	7 bytes	(*n*-7) bytes

Message ID
Is IST*xxxx*. It represents the message ID of any VTAM message that can be sent to the VTAM operator. Refer to z/OS Communications Server: SNA Messages for the specific IDs and text of VTAM messages.

Message text
Is the text of the VTAM message that corresponds to the message ID.

n Represents the total length of the message without the header bytes.

The text and occurrence of each message are VTAM release-dependent. Therefore, a program operator application program might have to be changed in subsequent releases if the messages change.

You can define USS tables to allow the program operator to run independently of changes in message text. Refer to the z/OS Communications Server: SNA Resource Definition Reference for details about creating and modifying USS definition tables.

- If this is a message that requires a reply (bit 6), the message data is in the form:

Header	Reply ID	Message ID	Message text
4 bytes	4 bytes	7 bytes	(*n*-11) bytes

Reply ID
Is a reply identification number that must be returned to VTAM with the reply in the REPLY command. It is a decimal number 0–99. Leading zeros are included but do not have to be included in the reply. This identifier should not be confused with the identification number that is used in the header.

n Represents the total length of the message without the header bytes.

- If this is a message that contains a copy of a VTAM operator command entered from the system console (bit 5), the message data is in the form:

Header	Command text
4 bytes	*n* bytes

Command text
is the text of the command as entered from the system console. The command keyword (DISPLAY, VARY, or MODIFY) is always received spelled in full (that is, not abbreviated).

n Represents the total length of the command without the header bytes.

Only the primary program operator can receive a copy of commands entered from the system console or messages sent to the console, and only if PPOLOG=YES as a result of the MODIFY PPOLOG command or the PPOLOG start option. In order to receive the next message transmission from VTAM, the program operator must have an outstanding RCVCMD.

If the message containing the reply identifier is not replied to by the program operator and it is subsequently sent to the system console, the number is changed and does not correspond to the original number.

Data sent to VTAM

A program operator can use the SENDCMD macroinstruction to send VTAM operator commands and the REPLY command to VTAM. The VTAM operator commands that can be sent are the VARY, DISPLAY, and MODIFY commands. The data that the program operator sends to VTAM is in this form:

Header	Command sent to VTAM
4 bytes	*n* bytes

Each VTAM command issued by the program operator has the same format and meaning as when it is entered at the system console. The VARY, DISPLAY, and MODIFY commands are described in z/OS Communications Server: SNA Operation.

The REPLY command is used to answer VTAM messages that require a reply from the program operator. It has the following format:

{REPLY or R}	reply identifier,[']text[']

reply identifier
Is the reply identification number (*nn*) that was received.

text
Is the reply to the VTAM message. If the actual wording of the text is supplied in the original message, it should be copied exactly in the reply. The apostrophes are optional and included only if the reply is not to be translated into uppercase characters. A blank immediately following the reply identification number indicates a null reply.

The abbreviated format of the REPLY command supported by z/OS for the system console cannot be sent by a program operator.

Example:

```
  IST259I  INOP RECEIVED FOR LS3A4A1  CODE = 01
  IST619I  ID = LS3A4A1  FAILED - RECOVERY IN PROGRESS
  IST526I  ROUTE FAILED FROM  3 TO  4 - DSA   4 - NETID NETA
  IST526I  ROUTE FAILED FROM  3 TO  4 - DSA   4 - NETID NETA
  IST238I  AM GUNBIND REQ FOR ID = NCP4AA6  RCVD RECOVERY IN PROGRESS
           RU DATA-TYPE = X'03', CAUSE = X'07'

  IST521I  GBIND QUEUED FOR COS ISTVTCOS FROM  SSCP1A   TO NCP4AA6
  IST528I  VIRTUAL ROUTE NUMBER    0   1   2   3   4   5
  IST523I  REASON = NO ROUTES OPERATIVE
  *00 IST095A  OPTION TO DUMP NCP4AA6  AVAILABLE -
               REPLY 'YES' OR 'NO' OR 'YES,DUMPSTA=LINKSTANAME'
  *00 IST284A OPTION TO RELOAD NCP4AA6 AVAILABLE -
               REPLY 'YES' OR 'NO' OR 'YES,LOADSTA=LINKSTANAME'
r 00,yes
```

The informational message IST619I is followed by messages that require replies (IST095A and IST284A); VTAM assigns it a reply ID of 00 in both cases. In both cases, the reply to the message is “yes”.

Format and DSECT of the message and command header

Table 141 and Table 142 describe the format map and DSECT for the VTAM message and command header (ISTDPOHD). The header contains status information and an ID number for each message or command that is sent or received by the application program sending SENDCMD and RCVCMD macroinstructions.

The format map and DSECT description can help in examining or setting up a header. The IBM-supplied DSECT ISTDPOHD is provided as part of the system macroinstruction library on SYS1.MACLIB.

To avoid the risk of duplicating DSECT labels in a program, no label should begin with the characters POH. All relevant bits should be set.

Table 141. Format of the VTAM message and command header (ISTDPOHD)

Dec	Hex	Header code	Status	Message identifier
0	0	(POHRSVD)	(POHSTAT)	(POHID)

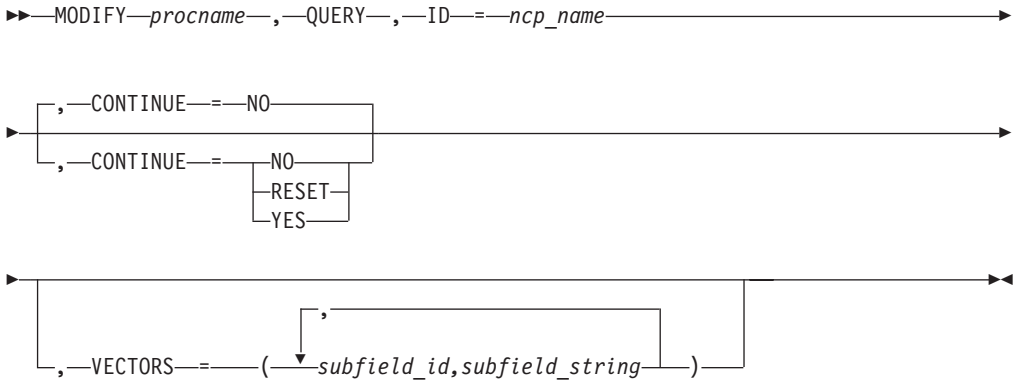
Table 142. VTAM message and command header DSECT (ISTDPOHD)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Displacement Dec	Displacement Hex	Length
Header Code	POHRSVD			Reserved.	0	0	1
Status	POHSTAT	POHEND	X'80'	Message is an end line.	1	1	1
		POHDATA	X'40'	Message is a data line.			
		POHLBL	X'20'	Message is a label line.			
		POHCNTRL	X'10'	Message is a control line.			
		POHPPCMD	X'04'	Message contains copy of VTAM operator command entered from the system console.			
		POHPPSOL	X'08'	Message contains copy of VTAM-solicited, unsuppressed message that had been sent to the system console.			
		POHPPSUP	X'0C'	Message contains copy of VTAM-solicited, suppressed message that had been sent to the system console.			
		POHRREQ	X'02'	A reply is requested.			
		POHGEN	X'01'	If the bit is off, the message was generated by VTAM. If the bit is on, the message was generated by the program operator.			

Table 142. VTAM message and command header DSECT (ISTDPOHD) (continued)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning	Displacement Dec	Displacement Hex	Length
Message Identifier	POHID			ID number.	2	2	2

POA communication with tuning facility using the MODIFY QUERY COMMAND



Abbreviations

Operand	Abbreviation
CONTINUE	CONT
CONTINUE=YES	YES
CONTINUE=NO	NO
CONTINUE=RESET	RESET
MODIFY	F
VECTORS	VECTOR or V

Purpose

The MODIFY QUERY command may be issued only from a program operator application.

The MODIFY QUERY command allows a program operator application to communicate with a tuning facility such as NTune

Note:

1. Any number of MODIFY QUERY commands may be issued, but the number and size of the subvectors are limited by the maximum number of bytes the NCP can receive in one PIU or PIU segment. The PIU size is determined by the value specified on the MAXDATA keyword of the PCCU definition statement.
2. Once a command group is started by a specific program operator application, commands are queried until a CONT=NO or RESET is received from the same program operator application. Multiple command groups may not be outstanding from the same program operator application.

3. The requestor must supply unique subvectors and track the responses to those subvectors.

Operands

procname
is the procedure name for the command.

If *procname* in the START command is specified as *startname.ident*, where *startname* is the VTAM start procedure and *ident* is the optional identifier, then either *startname.ident* or *ident* can be specified for *procname*.

If *procname* in the START command was *startname*, then *startname* must be specified for *procname*.

CONTINUE
specifies the command type.

CONTINUE=NO
specifies that this is a single command or the last command of a group. The VECTORS operand may be specified with CONTINUE=NO, but it is not required.

CONTINUE=RESET
specifies that all previous commands queued as part of a command group are to be purged and not sent to the NCP. The VECTORS operand is ignored if it is specified.

CONTINUE=YES
specifies that this command is part of a group of commands that make up a single request. The VECTORS operand is required when CONTINUE=YES.

ID=ncp_name
specifies the name of the NCP resource. The name cannot be a network-qualified name.

VECTORS=(subfield_id,subfield_string)
specifies the subvectors to be passed to the NCP. More than one subvector may be specified. A subvector consists of a subfield id paired with a subfield string.

For an example of the type of information used in the VECTORS operand, see *NTuneMON User's Guide*

subfield_id
specifies the subfield ID of the subvector passed to the NCP. This field must contain one or two hexadecimal characters. VTAM provides no other checks for validity.

subfield_string
specifies the user data portion of the subvector passed to the NCP. This string must be specified as an even number of characters in hexadecimal form.

Appendix M. List of macroinstructions

Attention: The macroinstructions identified in this appendix are provided as programming interfaces for customers by VTAM. Do not use as programming interfaces any VTAM macroinstructions other than those identified in this appendix.

The following macroinstructions are provided for general use:

ACB	ISTAPCVL	ISTVACBV
APPCCMD	ISTASDP	MODCB
CHANGE	ISTBLENT	NIB
CHECK	ISTDBIND	OPEN
CLOSE	ISTDBINH	OPNDST
CLSDST	ISTDNIB	OPNSEC
EXECRPL	ISTDPOHD	RCVCMD
EXLST	ISTDPROC	RECEIVE
GENCB	ISTDVCHR	REQSESS
GTDEVSIZ	ISTGAPPC	RESETSR
IFGACB	ISTGLBAL	RPL
IFGACBVS	ISTFM5	SEND
IFGACBVT	ISTMTS	SENDCMD
IFGEXLST	ISTRH	SESSIONC
IFGEXLVS	ISTRIVL	SETLOGON
IFGEXLVT	ISTRPL	SHOWCB
IFGRPL	ISTRPL6	SIMLOGON
IFGRPLVS	ISTRPL6X	STFSMODE
IFGRPLVT	ISTSLCNS	STLINENO
INQUIRE	ISTSLD	STTRAN
INTRPRET	ISTSREST	TERMSESS
ISTAMSVL	ISTUSFBC	TESTCB

The following macroinstructions are provided as product sensitive:

IKTIPARM
IKTMPL
IKTOPARM
IKTTCAST
IKTTSBX
IKTTVWA
IKTWESTD
IKTXSA

© Copyright IBM Corp. 2000, 2013

Appendix N. Application program migration

This appendix describes factors to consider in the following migration environments:

- When migrating application programs from a prior release of VTAM.
- When migrating from a single-domain environment to a multiple-domain environment
- When migrating from a multiple-domain environment to a multiple-network environment (SNA network interconnection)

"Simplifying migration and network upgrades" on page 35 describes how to avoid or minimize possible migration problems when coding new VTAM application programs.

Migrating from prior releases of VTAM

As each new release of VTAM was developed, new application program facilities were added, obsolete facilities were deleted, and errors were corrected. Significant internal changes were made to improve performance, reliability, availability, and serviceability. In this activity, the goal was to allow application programs that ran on a prior release of VTAM to run on the new release of VTAM without modification or reassembly of the application program. However, in certain cases, modification of an application program is required.

COS name and logon mode name

In current releases of VTAM, a class-of-service (COS) name is associated with each logon mode name. See "NIB LOGMODE and BNDAREA operands" on page 124. When a logon mode name is specified with an Initiate request, SIMLOGON, REQSESS, OPNDST OPTCD=ACQUIRE, or CLSDST OPTCD=PASS, the COS name associated with that logon mode name is used by VTAM to establish the session.

The COS name is not returned to the application program when INQUIRE OPTCD=SESSPARM is issued.

A program that issues INQUIRE OPTCD=SESSPARM with a logon mode name to obtain a session parameter, and then use OPNDST OPTCD=ACQUIRE with a BIND image (pointed to by the BNDAREA of the NIB) built from those a session parameter gets the class of service associated with the **default** logon mode name. That class of service might not be the one that is associated with the logon mode name used by the INQUIRE macroinstruction. If this is undesirable, use SIMLOGON with the logon mode name specified in the NIB LOGMODE operand. The session parameter and COS name are presented to the program in the CINIT RU. Current releases of VTAM use the associated class of service when an OPNDST OPTCD=ACCEPT is issued.

A program that issues CLSDST OPTCD=PASS must also specify the logon mode name in the NIB LOGMODE operand, to ensure that the subsequent OPNDST OPTCD=ACCEPT uses the proper class of service. If CLSDST OPTCD=PASS is issued with LOGMODE=0, the COS name of the default logon mode name is used.

© Copyright IBM Corp. 2000, 2013

Increase of ACB size

In current releases of VTAM, if the GENCB macroinstruction is used to build an ACB and a length is specified in the GENCB that is smaller than needed for the ACB, the GENCB fails with an error return code. Use the SHOWCB or TESTCB macroinstructions to get the correct ACB length at execution time before issuing GENCB. See the description of the GENCB macroinstruction in "GENCB—Generate a control block" on page 407, for information on how to use SHOWCB and TESTCB with GENCB.

If a program uses specific values to refer to an address beyond the ACB (for example, ORG MYACB+92), fields that are inside the ACB could be referenced (or overlaid) unintentionally. Use the IFGACB DSECT to get the ACB length at assembly time.

Application program minor node name in BIND

In current releases of VTAM, the network name of the primary logical unit (PLU) is used in the BIND request.

In installations where the ACB name and network name of PLUs are not the same, an SLU might receive a BIND request with a different PLU name than was expected. Some modifications might be required to take advantage of the network name. See "OPNSEC macroinstruction" on page 96 and "SESSIONC macroinstruction with CONTROL=BIND" on page 98 for more information.

Sequence number dependencies for LU type 0 3270 terminals

The sequence number associated with requests sent to or received from LU type 0 3270 terminals is handled differently, depending on how the 3270 is attached to the network. The sequence number wraps around to 0 after it reaches 255 for some 3270 terminals and after it reaches 65535 for others. Table 143 lists the wraparound point for the various types of attachment and products. Application programs sensitive to the sequence number wraparound point might require modification if the wraparound point changes. See the section "Summary of differences among LU type 0 3270 terminals" on page 335 for information on avoiding sequence number dependencies.

Table 143. Wraparound points for sequence numbers in sessions involving LU type 0 3270 terminals

Attachment	Product	Wrap point
BSC	NCP	255
SDLC	NCP	65535
Channel	VTAM	65535

Dynamic network access function

The application-supplied operands for dial connections function is supported in all current releases of VTAM.

Differences between BTAM and VTAM application programs

Table 144 shows the major similarities and differences between VTAM application programs and BTAM application programs. The chart provides assistance in evaluating the effort needed to revise the application program for communication with VTAM LUs.

BTAM can request sessions only with dedicated devices. Hence the next section about converting from a single-domain to a multiple-domain network offers hints that apply equally to BTAM application programs that are being converted to VTAM.

Table 144. Major similarities and differences between VTAM and BTAM application programs

Functions provided	VTAM application program	BTAM application program
Define line groups	Can be defined during VTAM definition or dynamically after VTAM is started	Use DCB or DTFBT macroinstruction to define line groups
Define terminals	Can be defined during VTAM definition or dynamically after VTAM is started	Use DFTRMLST macroinstruction
Initialize program	Use OPEN macroinstruction	Use OPEN macroinstruction
Sessions with terminals established dynamically	Use OPNDST macroinstruction	Not available, because terminals are statically connected when application program's job step is initiated
Release control of a terminal to another program that requests a session with it	When requested, use CLSDST with OPTCD=RELEASE specified	No comparable function
Receive input	Use RECEIVE macroinstruction	Use READ macroinstruction
Receive input from any terminal	Specify input can be from any terminal	Must poll each line separately
Receive input from a specific terminal	Specify input is to be received from a specific terminal	Specify terminal list entry
Send output	Use SEND macroinstruction	Use WRITE macroinstruction
Have data schedule for output from access method buffers (output buffering)	Specify that the data is scheduled for output	No comparable function
Test, display, or modify control block fields	Use manipulative macroinstructions: TESTCB, SHOWCB, MODCB, or use assembler language instructions	Use assembler language instructions
Record transmission errors	Performed by NCP and VTAM	Use error recording macroinstructions

Migrating from a single-domain to a multiple-domain environment

This section discusses application program considerations when converting from a single-domain network to a multiple-domain network. In designing your application programs, it is good practice to code them as though they always operate in a multiple-domain network. This reduces the need for future programming changes if you eventually migrate from a single-domain to a multiple-domain network. See also “Simplifying migration and network upgrades” on page 35 for additional information on network migration.

Use of INQUIRE for a cross-domain resource

For an LU in another domain, INQUIRE OPTCD=DEVCHAR or INQUIRE OPTCD=TERMS can be issued only in two situations:

1. When there is a queued CINIT for the LU
2. When the LU is in session with a PLU application program in the host processor in which the macroinstruction is processed.

INQUIRE OPTCD=SESSPARM with LOGMODE=0 specified in the NIB can be issued only when there is a queued CINIT for the resource. The INQUIRE macroinstruction with LOGMODE=C' ' or with LOGMODE=*logon mode name* in the NIB cannot be used for a cross-domain resource.

Specifying LOGMODE names with OPNDST for a cross-domain resource

When OPNDST OPTCD=ACCEPT is issued to accept a session with an LU in another domain, no logon mode name can be present in the LOGMODE field of the NIB (pointed to by the RPL specified in the request). When OPNDST OPTCD=ACCEPT is issued for a cross-domain resource, the LOGMODE field of the NIB can contain 0 (to specify that the BIND is to contain the session parameter that accompanied the CINIT), or the application program can put an address in the BNDAREA field of the NIB (to specify that the session parameter starting at that address are to be used in the BIND).

Use of INTRPRET for a cross-domain resource

In VTAM, if an interpret table is defined, it must be stored in the same domain as the device-type LUs related to the table. The interpret table is used only by the VTAM of that domain. A VTAM application program in one domain cannot issue an INTRPRET macroinstruction that calls for use of an interpret table in another domain.

Considerations for a multiple-network environment

This section describes application program considerations for SNA network interconnection.

Use of INQUIRE for a cross-network resource

For an LU in another network, the two valid option codes for the INQUIRE macroinstruction are OPTCD=STATUS and OPTCD=APPSTAT. These options provide information about the application program's ability to establish cross-network sessions. Other forms of the INQUIRE macroinstruction provide no information LUs in other networks.

Appendix O. Architectural specifications

This appendix lists documents that provide architectural specifications for the SNA Protocol.

The APPN Implementers' Workshop (AIW) architecture documentation includes the following architectural specifications for SNA APPN and HPR:

- APPN Architecture Reference (SG30-3422-04)
- APPN Branch Extender Architecture Reference Version 1.1
- APPN Dependent LU Requester Architecture Reference Version 1.5
- APPN Extended Border Node Architecture Reference Version 1.0
- APPN High Performance Routing Architecture Reference Version 4.0
- SNA Formats (GA27-3136-20)
- SNA Technical Overview (GC30-3073-04)

For more information, refer to the AIW documentation page at http://www.ibm.com/support/docview.wss?rs=852&uid=swg27017843.

The following RFC also contains SNA architectural specifications:

- RFC 2353 *APPN/HPR in IP Networks APPN Implementers' Workshop Closed Pages Document*

RFCs can be obtained from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Many RFCs are available online. Hardcopies of all RFCs are available from the NIC, either individually or by subscription. Online copies are available using FTP from the NIC at http://www.rfc-editor.org/rfc.html.

Use FTP to download the files, using the following format:

```
RFC:RFC-INDEX.TXT
RFC:RFCnnnn.TXT
RFC:RFCnnnn.PS
```

where:

- *nnnn* is the RFC number.
- TXT is the text format.
- PS is the postscript format.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to `service@nic.ddn.mil` with a subject line of `RFC nnnn` for text versions or a subject line of `RFC nnnn.PS` for PostScript versions. To request a copy of the RFC index, send a message with a subject line of `RFC INDEX`.

For more information, contact `nic@nic.ddn.mil`.

© Copyright IBM Corp. 2000, 2013

Appendix P. Accessibility

Publications for this product are offered in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when using PDF files, you can view the information through the z/OS Internet Library website or the z/OS Information Center. If you continue to experience problems, send an email to mhvrcfs@us.ibm.com or write to:

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. See z/OS TSO/E Primer, z/OS TSO/E User's Guide, and z/OS ISPF User's Guide Vol I for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer or Library Server versions of z/OS books in the Internet library at www.ibm.com/systems/z/os/zos/bkserv/.

One exception is command syntax that is published in railroad track format, which is accessible using screen readers with the Information Center, as described in "Dotted decimal syntax diagrams."

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always

© Copyright IBM Corp. 2000, 2013

present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should see separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- A question mark (?) means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- An exclamation mark (!) means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted

decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- An asterisk (*) means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

  Notes:

  1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
  2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
  3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the USA.

IBM may not offer all of the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

© Copyright IBM Corp. 2000, 2013

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want 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:

Site Counsel
IBM Corporation
P.O. Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations might not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing

application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

IBM is required to include the following statements in order to distribute portions of this document and the software described herein to which contributions have been made by The University of California. Portions herein © Copyright 1979, 1980, 1983, 1986, Regents of the University of California. Reproduced by permission. Portions herein were developed at the Electrical Engineering and Computer Sciences Department at the Berkeley campus of the University of California under the auspices of the Regents of the University of California.

Portions of this publication relating to RPC are Copyright © Sun Microsystems, Inc., 1988, 1989.

Some portions of this publication relating to X Window System** are Copyright © 1987, 1988 by Digital Equipment Corporation, Maynard, Massachusetts, and the Massachusetts Institute Of Technology, Cambridge, Massachusetts.

Some portions of this publication relating to X Window System are Copyright © 1986, 1987, 1988 by Hewlett-Packard Corporation.

Permission to use, copy, modify, and distribute the M.I.T., Digital Equipment Corporation, and Hewlett-Packard Corporation portions of this software and its documentation for any purpose without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the names of M.I.T., Digital, and Hewlett-Packard not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T., Digital, and Hewlett-Packard make no representation about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

Copyright © 1983, 1995-1997 Eric P. Allman

Copyright © 1988, 1993 The Regents of the University of California.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment:

```
This product includes software developed by the University of
California, Berkeley and its contributors.
```

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software program contains code, and/or derivatives or modifications of code originating from the software program "Popper." Popper is Copyright ©1989-1991 The Regents of the University of California. Popper was created by Austin Shelton, Information Systems and Technology, University of California, Berkeley.

Permission from the Regents of the University of California to use, copy, modify, and distribute the "Popper" software contained herein for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies. HOWEVER, ADDITIONAL PERMISSIONS MAY BE NECESSARY FROM OTHER PERSONS OR ENTITIES, TO USE DERIVATIVES OR MODIFICATIONS OF POPPER.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THE POPPER SOFTWARE, OR ITS DERIVATIVES OR MODIFICATIONS, AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE POPPER SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Copyright © 1983 The Regents of the University of California.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the University of California, Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior

written permission. THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Copyright © 1991, 1993 The Regents of the University of California.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment:

   `This product includes software developed by the University of California, Berkeley and its contributors.`
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 1990 by the Massachusetts Institute of Technology

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Furthermore if you modify this software you must label your software as modified software and not distribute it in such a fashion that it might be confused with the original M.I.T. software. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

Copyright © 1998 by the FundsXpress, INC.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of FundsXpress not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. FundsXpress makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Copyright © 1999, 2000 Internet Software Consortium.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND INTERNET SOFTWARE CONSORTIUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL INTERNET SOFTWARE CONSORTIUM BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright © 1995-1998 Eric Young (eay@cryptsoft.com)

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscape's SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)". The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related.
4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include acknowledgment:

   "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The license and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution license [including the GNU Public License.]

This product includes cryptographic software written by Eric Young.

Copyright © 1999, 2000 Internet Software Consortium.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND INTERNET SOFTWARE CONSORTIUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL INTERNET SOFTWARE CONSORTIUM BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright © 2004 IBM Corporation and its licensors, including Sendmail, Inc., and the Regents of the University of California.

Copyright © 1999,2000,2001 Compaq Computer Corporation

Copyright © 1999,2000,2001 Hewlett-Packard Company

Copyright © 1999,2000,2001 IBM Corporation

Copyright © 1999,2000,2001 Hummingbird Communications Ltd.

Copyright © 1999,2000,2001 Silicon Graphics, Inc.

Copyright © 1999,2000,2001 Sun Microsystems, Inc.

Copyright © 1999,2000,2001 The Open Group

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

X Window System is a trademark of The Open Group.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

You can obtain softcopy from the z/OS Collection (SK3T-4269), which contains BookManager and PDF formats.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: http://www-01.ibm.com/software/support/systemsz/lifecycle/
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS Communications Server.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ and all Java-based trademarks are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Adobe and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of IBM or other companies.

Bibliography

This bibliography contains descriptions of the documents in the z/OS Communications Server library.

z/OS Communications Server documentation is available in the following forms:

- Online at the z/OS Internet Library web page at www.ibm.com/systems/z/os/zos/bkserv/
- In softcopy on CD-ROM collections. See "Softcopy information" on page xxix.

z/OS Communications Server library updates

An index to z/OS Communications Server book updates is at http://www.ibm.com/support/docview.wss?uid=swg21178966. Updates to documents are also available on RETAIN® and in information APARs (info APARs). Go to http://www.ibm.com/software/network/commserver/zos/support to view information APARs. In addition, Info APARs for z/OS documents are in *z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation*, which can be found at http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/ZDOCAPAR.

z/OS Communications Server information

z/OS Communications Server product information is grouped by task in the following tables.

Planning

Title	Number	Description
z/OS Communications Server: New Function Summary	GC27-3664	This document is intended to help you plan for new IP or SNA function, whether you are migrating from a previous version or installing z/OS for the first time. It summarizes what is new in the release and identifies the suggested and required modifications needed to use the enhanced functions.
z/OS Communications Server: IPv6 Network and Application Design Guide	SC27-3663	This document is a high-level introduction to IPv6. It describes concepts of z/OS Communications Server's support of IPv6, coexistence with IPv4, and migration issues.

Resource definition, configuration, and tuning

Title	Number	Description
z/OS Communications Server: IP Configuration Guide	SC27-3650	This document describes the major concepts involved in understanding and configuring an IP network. Familiarity with the z/OS operating system, IP protocols, z/OS UNIX System Services, and IBM Time Sharing Option (TSO) is recommended. Use this document with the z/OS Communications Server: IP Configuration Reference.

© Copyright IBM Corp. 2000, 2013

Title	Number	Description
z/OS Communications Server: IP Configuration Reference	SC27-3651	This document presents information for people who want to administer and maintain IP. Use this document with the z/OS Communications Server: IP Configuration Guide. The information in this document includes: • TCP/IP configuration data sets • Configuration statements • Translation tables • Protocol number and port assignments
z/OS Communications Server: SNA Network Implementation Guide	SC27-3672	This document presents the major concepts involved in implementing an SNA network. Use this document with the z/OS Communications Server: SNA Resource Definition Reference.
z/OS Communications Server: SNA Resource Definition Reference	SC27-3675	This document describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA. Use this document with the z/OS Communications Server: SNA Network Implementation Guide.
z/OS Communications Server: SNA Resource Definition Samples	SC27-3676	This document contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.
z/OS Communications Server: IP Network Print Facility	SC27-3658	This document is for systems programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP Services.

Operation

Title	Number	Description
z/OS Communications Server: IP User's Guide and Commands	SC27-3662	This document describes how to use TCP/IP applications. It contains requests with which a user can log on to a remote host using Telnet, transfer data sets using FTP, send and receive electronic mail, print on remote printers, and authenticate network users.
z/OS Communications Server: IP System Administrator's Commands	SC27-3661	This document describes the functions and commands helpful in configuring or monitoring your system. It contains system administrator's commands, such as TSO NETSTAT, PING, TRACERTE and their UNIX counterparts. It also includes TSO and MVS commands commonly used during the IP configuration process.
z/OS Communications Server: SNA Operation	SC27-3673	This document serves as a reference for programmers and operators requiring detailed information about specific operator commands.
z/OS Communications Server: Quick Reference	SC27-3665	This document contains essential information about SNA and IP commands.

Customization

Title	Number	Description
z/OS Communications Server: SNA Customization	SC27-3666	This document enables you to customize SNA, and includes the following information: • Communication network management (CNM) routing table • Logon-interpret routine requirements • Logon manager installation-wide exit routine for the CLU search exit • TSO/SNA installation-wide exit routines • SNA installation-wide exit routines

Writing application programs

Title	Number	Description
z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference	SC27-3660	This document describes the syntax and semantics of program source code necessary to write your own application programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this document to adapt your existing applications to communicate with each other using sockets over TCP/IP.
z/OS Communications Server: IP CICS Sockets Guide	SC27-3649	This document is for programmers who want to set up, write application programs for, and diagnose problems with the socket interface for CICS using z/OS TCP/IP.
z/OS Communications Server: IP IMS Sockets Guide	SC27-3653	This document is for programmers who want application programs that use the IMS TCP/IP application development services provided by the TCP/IP Services of IBM.
z/OS Communications Server: IP Programmer's Guide and Reference	SC27-3659	This document describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the z/OS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.
z/OS Communications Server: SNA Programming	SC27-3674	This document describes how to use SNA macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain.
z/OS Communications Server: SNA Programmer's LU 6.2 Guide	SC27-3669	This document describes how to use the SNA LU 6.2 application programming interface for host application programs. This document applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this document.)
z/OS Communications Server: SNA Programmer's LU 6.2 Reference	SC27-3670	This document provides reference material for the SNA LU 6.2 programming interface for host application programs.
z/OS Communications Server: CSM Guide	SC27-3647	This document describes how applications use the communications storage manager.

Title	Number	Description
z/OS Communications Server: CMIP Services and Topology Agent Guide	SC27-3646	This document describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The document provides guide and reference information about CMIP services and the SNA topology agent.

Diagnosis

Title	Number	Description
z/OS Communications Server: IP Diagnosis Guide	GC27-3652	This document explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.
z/OS Communications Server: ACF/TAP Trace Analysis Handbook	GC27-3645	This document explains how to gather the trace data that is collected and stored in the host processor. It also explains how to use the Advanced Communications Function/Trace Analysis Program (ACF/TAP) service aid to produce reports for analyzing the trace data information.
z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures and z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT	GC27-3667 GC27-3668	These documents help you identify an SNA problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation.
z/OS Communications Server: SNA Data Areas Volume 1 and z/OS Communications Server: SNA Data Areas Volume 2	GC31-6852 GC31-6853	These documents describe SNA data areas and can be used to read an SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

Messages and codes

Title	Number	Description
z/OS Communications Server: SNA Messages	SC27-3671	This document describes the ELM, IKT, IST, IUT, IVT, and USS messages. Other information in this document includes: • Command and RU types in SNA messages • Node and ID types in SNA messages • Supplemental message-related information
z/OS Communications Server: IP Messages Volume 1 (EZA)	SC27-3654	This volume contains TCP/IP messages beginning with EZA.
z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)	SC27-3655	This volume contains TCP/IP messages beginning with EZB or EZD.
z/OS Communications Server: IP Messages Volume 3 (EZY)	SC27-3656	This volume contains TCP/IP messages beginning with EZY.
z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)	SC27-3657	This volume contains TCP/IP messages beginning with EZZ and SNM.
z/OS Communications Server: IP and SNA Codes	SC27-3648	This document describes codes and other information that appear in z/OS Communications Server messages.

Index

Special characters

&ISTGLRL global variable
  declared and set 274

Numerics

0 value on LOGMODE operand 436
31-bit addressing 51, 319
3270 display station
  characteristics of 328
  communicating with 327
  data flow control 329
  transmission control 331
3270 terminals, types of 327

A

AAREA operand 492
AAREALN operand 492
ABEND (abnormal end)
  error handling 322, 324
  of VTAM, causing entry to TPEND exit routine 263
  pattern of abnormal termination processing 278
abnormal end (ABEND)
  error handling 322, 324
  of VTAM, causing entry to TPEND exit routine 263
  pattern of abnormal termination processing 278
ACB (access method control block)
  address 447
  address operand
    of CLOSE 392
    of OPEN 446, 447
  address space 312
  allocation of storage 376
  contents 56
  control block 740
  data space 375
  ERROR field 277, 383
  error field settings
    CLOSE ACB 393
    OPEN ACB 449
  fields, set by application program
    APPLID 56, 377
    EXLST 378
    MACRF 378
    PARMS 380
    PASSWD 382
  fields, set by VTAM
    ACBAMSVL 383
    ACBPSINS 383
    ACBRIVL 383
    ERROR 383
    OFLAGS 383
  IFGACB DSECT for 272, 740
  level of error isolation 322
  macroinstruction
    CLOSE 22, 73
    definition of 21
    example 57
    OPEN 22, 55
ACB (access method control block) *(continued)*
  multiple 65
  operand
    of the MODCB 431
    of the RPL 487
    of the SHOWCB 551
    of the TESTCB 565
  storage key of 376
  testing OFLAGS field 277
  using 375
  using multiple ACBs within one task 299
ACB-based macroinstruction 320
ACB-oriented exit routines 406
ACBAMSVL (access-method-support vector list)
  address of 383
  format 62
ACBLEN operand value
  field name operand for TESTCB 567
  obtaining 410
ACBRIVL (resource-information vector list) 383
ACCEPT value on OPTCD operand 497
accepting a session with OPNDST macroinstruction 89
access method control block (ACB)
  address 447
  address operand
    of CLOSE 392
    of OPEN 446, 447
  address space 312
  allocation of storage 376
  contents 56
  control block 740
  data space 375
  ERROR field 277, 383
  error field settings
    CLOSE ACB 393
    OPEN ACB 449
  fields, set by application program
    APPLID 56, 377
    EXLST 378
    MACRF 378
    PARMS 380
    PASSWD 382
  fields, set by VTAM
    ACBAMSVL 383
    ACBPSINS 383
    ACBRIVL 383
    ERROR 383
    OFLAGS 383
  IFGACB DSECT for 272, 740
  level of error isolation 322
  macroinstruction
    CLOSE 22, 73
    definition of 21
    example 57
    OPEN 22, 55
  multiple 65
  operand
    of the MODCB 431
    of the RPL 487
    of the SHOWCB 551
    of the TESTCB 565

© Copyright IBM Corp. 2000, 2013

access method control block (ACB) *(continued)*
storage key of 376
testing OFLAGS field 277
using 375
using multiple ACBs within one task 299
access-method-support vector list (ACBAMSVL)
address of 383
format 62
accessibility 899
ACQUIRE parameter
explanation of 453
operand value 497
acquiring sessions with OPNDST macroinstruction 90
action code
for inbound sequence number 537
for outbound sequence number 537
active application program, testing for 417
active logical unit, definition of 81
address
31-bit 319
address space
associated 316
multiple 312
session 316
termination 323
types of 312
used for exit routine execution 316
addressability in exit routines 234
ALIAS application 350
ALT value on CODESEL operand 519
AM operand
of the ACB macroinstruction 377
of the EXLST macroinstruction 406
of the GENCB macroinstruction 409
of the MODCB macroinstruction 432
of the RPL macroinstruction 492
of the SHOWCB macroinstruction 552
of the TESTCB macroinstruction 567
AMODE specifications 319
ANY value on OPTCD operand 502
any-mode
used to handle an inquiry 175
API (application program interface)
general requirements 50
handling control blocks 51
special requirements 51
APPL statement, name of application program in 375
APPL value on SDT operand 441
application program
as a logical unit 3
authorization to OPEN 10
authorized path under MVS 300
availability of 417
closing
as a generic resource 78
description 55, 73
with CLOSE macroinstruction 13, 28
with standard HALT command 76
closing in MVS 321
coding guidelines 33, 336
communicating with logical units 151
communication part 10, 13
controlling 85
designated for CNM routing 340
easing migration and upgrades 35
identification 56, 78, 377
in relation to a terminal operator and devices 17
application program *(continued)*
in relation to logical units in a network 17
in relation to other application programs 17
in SNA network 14
interfacing with MVS and VTAM 36, 50
ISTPDCLU application program 355
ISTSWBFR (session awareness data buffer) 356
LU 6.2 components 17
LU Initiate and Terminate request
description 85, 86
TERMSESS restrictions 86
mainline part 31
major functions 18
major start functions 7
opening
description 55
relationship to ACB 27
through an ACB 10
opening in MVS 320
organizing 33, 45
processing part 16
required control blocks for 14
schematic picture of 8
serial execution 309
sharing resources among 15
terminating 390
types of instructions 15
using multiple ACB's in 65
using to manage the network 18
VTAM definition requirements 355
VTAM interfaces and interactions 355
application program interface (API)
general requirements 50
handling control blocks 51
special requirements 51
application program migration
BTAM programs, differences between BTAM and VSAM 895
from a single-domain to a multiple-domain network
INQUIRE, for a cross-domain resource 896
INTRPRET, for a cross-domain resource 896
LOGMODE names, specifying with OPNDST for a cross-domain resource 896
from prior releases of VTAM
ACB size, increase of 894
BIND, application program minor node name in 894
sequence number dependencies for LU type 0 3270 terminals 894
SNA network interconnection requirements
INQUIRE, for a cross-network resource 896
application-supplied dial parameters (ASDP)
control block
format 743
using 119
ISTASDP DSECT for 745
macroinstructions
NIB 122
OPNDST 89, 122
SIMLOGON 122
APPLID operand 56, 377
APPLID processing 377
APPSTAT value on OPTCD operand 417
AREA operand
in RPL macroinstruction 493
in SHOWCB macroinstruction 552
AREALEN operand 493
ARECLEN field in RPL 506

ARG field in RPL 506
ASDP (application-supplied dial parameters)
control block
format 743
using 119
ISTASDP DSECT for 745
macroinstructions
NIB 122
OPNDST 89, 122
SIMLOGON 122
associated address space 316
association, task 310
ASY (asynchronous handling) 502
ASY operand value 502, 521
asynchronous exit routine 31
asynchronous handling (ASY) 502
asynchronous operation
advantages of 45
characteristics of 41
errors for 279
versus synchronous 169
asynchronous request 170
ATTN operand
EXLST 406
AUTHEXIT=YES 307
authorization
of application programs 296
authorized exit routine
MVS 307
authorized path
BRANCH operand 494
coding requirements 301
definition of 300
description 302
examples 304
macroinstructions
MVS 301
versus categories of VTAM macroinstructions 302
with RPL exit routines 302
authorized path facility, coded example 627
available logical unit, definition of 82

B

batch function, communication with 17
BB (Begin Bracket) indicator
operand value
for RPL 493
for SEND 518
using 214
Begin Bracket (BB) indicator
operand value
for RPL 493
for SEND 518
using 214
BID data 138
BID request
operand value 526
sending 526
bidder, in bracket protocol 215
BIND area
BNDAREA operand 116, 124, 435
definition of 123
format and DSECT 816
BIND image
session parameter area format 793
BIND request
basic function of 7
establishing a cryptographic session 130, 131
establishing an LU-LU session 7
need for SCIP exit to process 256
negotiable 131
non-negotiable 91
OPNSEC PROC options 97
receiving 102
rejection of 528
response 97
sending 97
session parameters in 356
BIS (bracket initiation stopped) 526
BIS data 138
BIS value on CONTROL operand 526
BLK operand of GENCB macroinstruction 409
BNDAREA
for LU profiles 828, 840
ISTDBIND DSECT 821
BRACKET field
for RPL 493, 507
for SEND 518
bracket indicators 681
bracket initiation stopped (BIS) 526
brackets
bracket protocol 214
bracket state transitions at the 3270 SLU 331
protocols used in session with 3270 terminals 330
BRANCH operand 494
branching table
using TESTCB return codes 568
buffer group 194
buffer list
entry format 192
LMPEO state transitions 195
buffer-list entry (ISTBLENT)
format of 192
ISTBLENT DSECT for 746
buffer-list LMPEO states
accumulate state 193
reset state 193
split state 193
buffer-list option (BUFFLST)
buffer-list operation 192
description 191
example of 200
operating considerations 191
BUFFLST (buffer list option)
buffer-list operation 192
description 191
example of 200
operating considerations 191

C

C value on LOGMODE operand 436
CA (continue-any mode)
CA value
OPTCD operand 498
for a RECEIVE operation 173
operand value 438, 520
processing option 438
used to handle concurrent inquiries 177
versus continue-specific mode 176
cancel closedown 263

CANCEL field
for SEND 526
CANCEL request
discarding incomplete chain 202
receiving 514
cancelling RECEIVE requests 483
CEB (conditional end bracket)
in user RH (USERRH) option 197
LMPEO handling of 186
when turned on 214
CHAIN field
for RPL 494
for SEND 519
chain indicator
from initial RH chain indicators 186
chaining
using a 3270 terminal 329
chaining of data requests
bracket indicators 681
change-direction indicators 681
description 201
example of 202
chaining output routine
logic of the 3600 617
logic of the 3601 617
change direction command (CMD) indicator
using 212, 213
change-direction
indicators
sending 519
protocol
description 212, 213
CHASE operand value
for SEND 527
Chase request
ensuring all responses have been received 206
sending 527
using 216
CHECK
addressing mode 321
basic function of 25
in an RPL exit routine 219
using 387
CHNGDIR operand
RPL 494
SEND 519
CID (communication identifier)
communicating with logical units 168, 216
explanation of 115, 442
operand value 553
CIDXLATE operand value 419
CINIT (Control Initiate request)
using session parameters with 127
CINIT (Control Initiate Request)
and LOGON exit routine 6
basic function of 6
purpose 83
class of service 93, 126
CLEANUP request
as one of several session outage notification signals 110
definition of 104
examples of 252
format of 252
received by an application program 244
Clear request
need for SCIP exit routine to process 256
sending 532
Clear request *(continued)*
to stop flow of requests and responses 164, 167
CLEAR value on CONTROL operand 535
CLOSE
ACB storage allocation 390
basic function of 23
CLOSE ACB errors 393
errors and special conditions
organization of information 277
forms of
list and execute form 391
standard form 391
using 390
closedown of VTAM 407
closing a program
description 73
as a generic resource 78
in MVS 321
with CLOSE 13
closing an ACB 390
CLSDST
basic function of 23
OPTCD=PASS operand
determining session parameters for 125
using to initiate sessions 93
OPTCD=RELEASE operand 93
OPTCD=TERMQ operand 94
scope of 92
network-qualified names 93
SSENSEO 401
using 92, 394
network-qualified names with 399, 402
CMD (Change Direction Command) indicator
using 212, 213
CNM (communication network management)
ALIAS application 350
application program 337
description 337
interface
coding requirements 340
protocol and procedure 342
RU (request unit) format 342
standard headers 343
COBOL, in writing an application program 16
CODESEL operand
RPL 494, 507
SEND 519
coding
macroinstructions and exit routines 316
requirements for authorized path 301
coding guidelines
application programs 33, 336
program structure 33
coding requirements for communication network management interface 337
coding rules for multiple address space 312
communicating with logical units
introduction 151
requests and responses 151
using SNA protocols 201
using VTAM 168
communication activity
separating from other activity 297
communication identifier (CID)
communicating with logical units 168, 216
explanation of 115, 442
operand value 553

communication network management (CNM)
ALIAS application 350
application program 337
description 337
interface
coding requirements 340
protocol and procedure 342
RU (request unit) format 342
standard headers 343
communication part of an application program 16
Communications Server for z/OS, online information xxxi
COMPLETE value on I/O operand 568
completion conditions
asynchronous requests 286
component ID vector 62
CON field in NIB 443
CONALL value on OPTCD operand 498
CONANY
concepts of establishing and terminating sessions 81
value on OPTCD operand 498
condition code of TERMSESS 563
conditional connection request (Q-NQ) 501
conditional end bracket (CEB)
in user RH (USERRH) option 197
LMPEO handling of 186
when turned on 214
confidential data handling 439
contention 210
continue chain operand 189
continue-any mode (CA)
CA value
OPTCD operand 498
for a RECEIVE operation 173
operand value 438, 520
processing option 438
used to handle concurrent inquiries 177
versus continue-specific mode 176
continue-specific (CS) mode
CS value
OPTCD operand 498, 520
PROC operand 438
processing option 438
continue-specific mode
used to handle concurrent inquiries 177
versus continue-any mode 176
control block
field lengths 547
field testing 565
generating
during program execution 407
with ACB 21, 377
with EXLST 21, 378, 404
with GENCB 407
with NIB 21, 433
with RPL 21, 487
manipulating
with GENCB 22, 407
with MODCB 22, 431
with SHOWCB 22, 551
with TESTCB 22, 565
required for application program 14
setting values in 267
techniques for handling 51
using for session establishment and termination 114
control block field
length of 553
tested with SHOWCB 553
control block field *(continued)*
tested with TESTCB 568
usage, summary 633
control block fields set by VTAM 633
control block format
ACB
MVS 740
ASDP 743
BLENT 746, 761
BNDAREA (ISTDBIND) 817
EXLST 750
MTS 752
NIB 754
RH 763
RPL 766
control block formats and DSECTs 739, 789
CONTROL field
RPL 495, 507
SEND 526, 527
SESSIONC 535
Control Initiate request (CINIT)
and LOGON exit routine 6
basic function of 6
purpose 83
using session parameters with 127
control points 2
control requests and indicators, summary of 681
Control Terminate request (CTERM)
cleanup 84
forced 84
orderly 84
control vector hex 29 747
controlling flow of requests and responses 164, 205
conventions used to describe VTAM macroinstructions 371
converting a CID to a symbolic name 419
converting a symbolic name to a CID 423
COUNTS value of OPTCD operand
INQUIRE 419
CP-CP sessions 4
cross-memory API support
and CHECK 388
and CLOSE 390
and OPEN 445
function of 317
limitations for application programs 318
CRYPT operand
RPL 495
SEND 519
cryptographic session
control 815
cross-domain 133
determining level of 133
establishing 132
INQUIRE OPTCD=SESSKEY 422
session-cryptography key
cross-domain 133
single-domain 132
single-domain 132
cryptography
definition of 117
establishing requirements from the logon mode entry 135
level for OPNDST request 134
level for OPNSEC request 136
requirements 218
CS (continue-specific) mode
CS value
OPTCD operand 498, 520

CS (continue-specific) mode *(continued)*
CS value *(continued)*
PROC operand 438
processing option 438
CTERM (Control Terminate Request)
cleanup 84
forced 84
orderly 84

D

data buffer
session awareness 356
trace 357
data communication activity
dividing among several tasks 297
separating from other activity 297
data facility storage management system(DFSMS) 50
data in a message 152
data integrity damage
handling of 293
data stream
3270, LU type 0 328
DATA value on CONTROL operand
SEND 526
data-flow-control
3270, LU type 0 329
purpose 164
requests 329
declarative macroinstruction
building control blocks 320
description 21
DSECT-creating
designation of 21
default entry in the logon mode table 123
defining sets of session parameters 123
definite response
need for requesting, with SEND POST=RESP 171
definite response indication (types 1 and 2)
meaning of 156
receiving 157
requesting 157
sending 157
delayed request mode 205
delayed response mode 205
Deliver and Forward RU flow 339
Deliver request unit
flow 339
format 345, 348
interface
coding requirements 340
requests and responses 341
DEVCHAR field
in a NIB 443
value on OPTCD operand 413, 420
device characteristics field 334
device-type logical unit
Initiate and Terminate request 82, 86
DFASY exit routine
advantages and disadvantages 236
and the RPL user RH field 198
any-mode 174
executing
in SRB mode 307
in TCB mode 307
expedited requests and responses 161
how to use 236
DFASY exit routine *(continued)*
how VTAM handles DFASY input 180
list of expedited requests and responses 236
parameters passed to 180
registers upon entry 237
sample program 2 logic 624
scheduled when an expedited-flow request is received 168, 178
specific-mode 174
specifying in ACB or NIB 229
versus RECEIVE macroinstruction 158
DFASY operand
EXLST 406
RECEIVE 471
RESETSR 483
RPL 504
DFASY request and response units
definition of 160
DFASYX processing option 439
DFSMS (Data Facility Storage Management System) 50
DFSYN request and response units
definition of 160
how handled by VTAM 182
DFSYN value on RTYPE operand
RESETSR 483
RPL 504
dial usability enhancements
conditions for using 121
dial parameter list 120
format of connection subfield 745
format of CPNAME subfield 744
format of dial number subfield 743
format of direct call line name subfield 744
format of DLCADDR subfield 745
format of expanded dial information subfield 744
format of IDBLK/IDNUM subfield 744
format of ISTASDP 743
function of 119
disability 899
disabled logical unit, definition of 81
dispatching priorities 32
DISPLAY command 529
DNS, online information xxxii
Downstream Load Utility (DSLU) 338
DSECT-creating macroinstructions 21
DSECTs and control block formats 739, 789
DSLU (Downstream Load Utility) 338

E

EB (End Bracket) indicator
value on BRACKET operand
for RPL 493
for SEND 518
ECB (event control block)
field in RPL 495
posting 170
using 41
versus RPL exit routines 42, 170
enabled logical unit, definition of 81
enciphered data request
sending and receiving 218
ENCR operand on NIB macroinstruction 435
encryption facility 435
definition of 117
establishing requirements from the logon mode entry 135
level for OPNDST request 134

encryption facility *(continued)*
level for OPNSEC request 136
requirements 218
End Bracket (EB) indicator
value on BRACKET operand
for RPL 493
for SEND 518
environment errors
handling 293
ERET operand 567
ERP (error recovery procedure)
during session initiation 44
error
ACB (application program) isolation 323
request level isolation 322
session level isolation 322
task level isolation 323
ERROR field
using after CLOSE processing 393
using after OPEN processing 448
error recovery procedure (ERP)
during session initiation 44
errors and special conditions
3270, LU type 0 330, 332
analyzing
for error isolation 322
for manipulative macroinstructions 278
for OPEN and CLOSE 277
for RPL-based macroinstructions 278
asynchronous operations 283
handling of
data integrity damage 293
environment errors 293
exception requests 291
logic errors 294
negative responses 293
retriable completion 293
software errors 293
synchronous operations 282
using FDBK field 417
using LERAD and SYNAD exit routines for 290
establishing and terminating sessions
BIND and UNBIND 7
macroinstructions
CLSDST 92
OPNDST 89
OPNSEC 96
REQSESS 95
SESSIONC 98
SIMLOGON 87
TERMSESS 99
stages of 83, 85
with logical units 81
establishing cross-domain cryptographic session 133
establishing single-domain cryptographic session 132
ESTAE exit routine 324
event control block (ECB)
field in RPL 495
posting 170
using 41
versus RPL exit routines 42, 170
EX value on RESPOND operand
SEND 523
exception conditions
3270, LU type 0 332
and sense information 332
handling 291
exception requests
handling
by a PLU application 292
by an SLU application 292
excess data, saving 499
exchanging
requests 152, 157
responses 152, 157
EXECRPL
basic function of 25
using 403
EXIT operand
as internal ECB 507
instead of ECB operand with RPL exit routine 496
RPL exit routine address 488
exit routine
address space used for execution of 316
addressability and save area requirements 234
addressing mode 321
asynchronous 31
basic function of 19, 28
cautions, restrictions, and techniques for 235
creation 404
deciding how to use 228
entry procedures for 234
executing
in SRB mode 307, 309
in TCB mode 307, 309
execution of 321
exit procedures from 235
how to use 219
identified by ACB 228
identified by NIB 228
identified in RPL-based macroinstructions 219
inline 31
installation 219
parameters passed to 234
procedures for writing 231
requirements for reenterability 231, 234
RPL-specified 29
rules of coding 316
session establishment and termination 100
summary of 224
task association 310
types of
exit-list exit routines 28, 221, 224
RPL-specified exit routines 28, 219, 256
EXLLEN value on LENGTH operand 410
EXLST
basic function of 21
named in EXLST operand of ACB 222
named in EXLST operand of NIB 222
names of exit routines in 222
scheduling 178
using 404
EXLST (IFGEXLST) DSECT 751
EXLST control block 405, 750
EXLST exit routine
addressing mode 321
definition of 29, 219
executing
in SRB mode 307
in TCB mode 307
optional 228
registers upon entry 226
required 228
specification and function of 221, 224

EXLST exit routine *(continued)*
specified in ACB 222, 228
specified in NIB 222, 228
versus explicit RECEIVEs 178
EXLST operand
ACB 378
MODCB 432
NIB 435
SHOWCB 552
TESTCB 567
expedited-flow data-flow-control request
expedited-flow data-flow-control
summary of receiving 687
session-control
receiving, summary of 689
sending, summary of 687
summary of receiving 687
expedited-flow request
ability to send
during quiesced state 212
in change-direction protocol 212
and responses, table summary 161
controlling normal-flow responses 161
definition of 159
examples of
for synchronous operations 169
extracting control block fields 551
for a receive-any operation 174
for a receive-specific operation 174
sequence numbers in 163
versus normal-flow requests 160
ways of receiving
DFASY exit routine 174, 236
RECEIVE 157, 162
RECEIVE RTYPE=DFASY 178
RESETSR 482
extended reference facility (XRF)
and SESSIONC 532
programming 149
session requests 91
terminating sessions 74

F

FDB2 (reason code) 244
FDBK return code, for INQUIRE macroinstruction (OPTCD=APPSTAT) 417
FIELDS operand 552
FIRST operand
RPL 494
SEND 519
FM (function management) header
using 217
FMD (function management data)
header option 217, 499
sending of, by LMPEO 183, 500
FME operand value 503
FMH-5 138
Forward and Deliver RU flow 339
forward request unit flow 339
forward request unit, CNM interface 341
FRR (functional recovery routines) 324
function management (FM) header
using 217
function management data (FMD)
header option 217, 499
sending of, by LMPEO 183, 500
function management profile 795
function management usage field 796
function-list macroinstruction global variables 275
function-list vector 62
functional recovery routines (FRR) 324

G

gathering performance data 358
GENCB
advantage of 267
basic function of 22
errors and special conditions for 278
examples of 268
how to use 268, 407
generating control blocks
during program execution 407
generating NIBs 271
generic resource 78
determining network qualified name of real instance 414, 422
example use of SETLOGON 547
opening and closing an application program 78
specifying application name 436
terminating LU-to-application association 384, 545
GETMAIN facility 410
global values in control blocks
setting 267
testing 267
global variables
declared and set 274

H

half-duplex contention communication 211
half-duplex devices 209
half-duplex flip-flop communication 211
HALT command
action for HALT NET, CANCEL or abnormal termination 77
action for HALT NET, QUICK or VTAM-initiated HALT 76
action for standard HALT 76
for application program without TPEND exit 76
hardware monitor 338
header
for VTAM messages 464
for VTAM operator commands 529
function management 499
HOLD value on OPTCD operand
RPL 501
SETLOGON 548

I

I/O operations
cancelling 483
input 466
output 514
I/O routine
logic of the 3270 619
IBM Software Support Center, contacting xxiv
IBSQAC operand
designating type of STSN request 536
used by SESSIONC 496
when SESSIONC is completed 507

IBSQVAL operand
  assigned to inbound requests 537
  used by SESSIONC 496
  when SESSIONC is completed 507
IFGACB DSECT for ACB 272, 740
IFGEXLST DSECT for EXLST 272, 751
IFGRPL DSECT for RPL 272
immediate request mode 206
immediate response mode 206
inactive application program 417
inbound sequence number
  description 535, 542
inbound STSN indicators 536
indicators
  in requests and responses
    definition of 151
    in a request 151
Information APARs xxix
inhibited logical unit, definition of 82
initial RH, location of 185
Initiate Load Request RU format 350
Initiate request
  basic function of 6
  LOGON, character-coded 85
  purpose 83, 85
  sources 83, 85
initiating sessions
  initiate request 6
  macroinstructions
    SIMLOGON 87
  use of LOGON exit routine 6
  using generic resource name 78
inline exit routine 31, 224
input operations, receiving 466
input RU
  classified by VTAM 180
INQUIRE
  basic function of 25, 412
  determining session parameters for 124
  OPTCD=TERMS 271
  permissible option codes 413
  using 412
  using to get a logon message 575
Internet, finding z/OS information online xxxi
interpret table, definition of 426
interpreting an input sequence 426
INTRPRET
  basic function of 25, 426
  using 426
isolating errors
  application program 323
  request 322
  session 322
  task 323
ISTASDP DSECT 745
ISTBLENT (buffer list entry)
  format of 192
  ISTBLENT DSECT for 746
ISTBLENT DSECT 746
ISTDBIND DSECT
  for BNDAREA 821
  using to build or examine session parameters 272
ISTDNIB DSECT for NIB 272, 754
ISTDPOHD DSECT 888
ISTDPROC DSECT for NIB 761
ISTDPROC macroinstruction for processing options fields of the NIB 272
ISTDVCHR DSECT for NIB 756
ISTDVCHR macroinstruction for device characteristics field of the NIB 272
ISTGLBAL macroinstruction
  control block fields 639
  how to use 430
  macroinstruction global variables set by
    &ISTGLCI (component-ID) 274
    &ISTGLRL (release-level) 274
    &ISTGLxy (function-level) 274, 275
ISTMTS DSECT 753, 762
ISTPDCLU application program 355
ISTRH DSECT 272, 763
ISTUSFBC DSECT 272, 774

K

KEEP option for overlength input data
  in record-mode operations 182
  value on PROC operand
    NIB 472
    RPL 499
keyboard 899
keyword operand
  as part of the VTAM macroinstruction language 19
  of the GENCB macroinstruction 410

L

LANG 117
LANGTAB 117
language code values 117
large message performance enhancement outbound (LMPEO)
  Begin RU/End RU combinations 195
  buffer-list option and 184, 191
  chaining of data requests 201
  data stream considerations 189
  description 183
  encrypt/decrypt facility and 184
  example of using 200
  exception conditions 190
  handling negative response 190
  handling request headers 185
  handling selected RH indicators 186
  operating considerations 184
  operation on a message sent to an SNA LU 184
  performance conditions 191
  sending FM data 189
  sequence number handling 189
  state transitions 195
  status during buffer list processing 193
LAST value on CHAIN operand
  RPL 494
  SEND 519
LENGTH operand
  GENCB 410
  SHOWCB 552
LERAD exit routine
  addressing mode 321
  advantages of 237
  basic function of 13, 406
  coding 290
  coding, special requirements 230
  executing
    in SRB mode 308
    in TCB mode 308

LERAD exit routine *(continued)*
how to use 237
linkages, conventions for 234, 237
not reentrant 234
operand 406
parameters passed to 237
purpose of 237
reentrant 234
register usage 291
registers upon entry 238
license, patent, and copyright information 903
list of NIBs
creating 433, 436
explanation of 433
LISTEND operand on NIB macroinstruction 436
LMPEO (large message performance enhancement outbound)
Begin RU/End RU combinations 195
buffer-list option and 184, 191
chaining of data requests 201
data stream considerations 189
description 183
encrypt/decrypt facility and 184
example of using 200
exception conditions 190
handling negative response 190
handling request headers 185
handling selected RH indicators 186
operating considerations 184
operation on a message sent to an SNA LU 184
performance conditions 191
sending FM data 189
sequence number handling 189
state transitions 195
status during buffer list processing 193
load operation 338
load request 338
Load Status (RU) format 350
logic errors
handling 294
logical unit (LU)
active 81, 84
available 81, 82
communicating with
application programs 151
description 28
VTAM 151, 168
communication protocol 209
connected 81
definition of 2
determining status 423
device-type 14
disabled 81
enabled 81
establishing sessions with 11, 27
examples of 14
identifying 168
in SNA network 2
primary session 4
quiescing an application program 207
receiving requests from 154
secondary session 4
SSCP-LU session 6
symbolic name 115
terminating sessions with 13, 28
logical unit presentation services
profile 799
profile 0 usage field 800
logical unit presentation services *(continued)*
profile 1 usage field 801
profile 2 and 3 usage fields 806
profile 4 usage field 808, 812
usage fields 800
Logical Unit Status (LUSTAT) request
sending 514, 526
LOGMODE operand values 126
LOGMODE operand, to identify a logon mode 436
logoff
using the 3270 terminal 335
logon
initiating 543
terminating 543
LOGON exit routine
accepting sessions in 238
advantages of 238
basic function of 6
examples of
in logic of sample program 1 575
executing
in SRB mode 307
in TCB mode 307
how to use 101
parameters passed to 238
registers upon entry 240
using INQUIRE macroinstruction in 238
versus RPL exit routine 238
with OPNDST OPTCD=ACCEPT 238
logon message
receiving 413
using the 3270 terminal 335
logon mode name
and session parameters 123
definition of 116
locating in the CINIT RU 128
using 123
logon mode, used by
CLSDST 92, 436
INQUIRE 436
OPNDST 89, 436
REQSESS 95, 436
SIMLOGON 87, 436
LOGON operand
EXLST 406
LOGON value of MACRF operand
ACB 378
LOGONMSG value of OPTCD operand
INQUIRE 413, 420
LOSTERM exit routine
entry codes for 242
executing
in SRB mode 307
in TCB mode 307
how to use 106, 241
operand 406
parameters passed to 241
reasons for entry to 241
registers upon entry 242
LU (logical unit)
active 81, 84
available 81, 82
communicating with
application programs 151
description 28
VTAM 151, 168
communication protocol 209

LU (logical unit) *(continued)*
connected 81
definition of 2
determining status 423
device-type 14
disabled 81
enabled 81
establishing sessions with 11, 27
examples of 14
identifying 168
in SNA network 2
primary session 4
quiescing an application program 207
receiving requests from 154
secondary session 4
SSCP-LU session 6
symbolic name 115
terminating sessions with 13, 28
LU-LU session protocols 342
LUSTAT (Logical Unit Status) request
sending 514, 526

M

MACRF operand
ACB 378
with SETLOGON 82
macroinstruction
ACB 375
ACB-based 22, 320
authorized path
MVS 301
categories of 20
CLOSE 387, 391
CLSDST 92, 394
declarative
ACB 21, 320
EXLST 21, 320
NIB 21, 320
RPL 21, 320
differences across operating systems 295
DSECT-creating
how to use 272, 273
list of 272
rules for coding 273, 316
establishing and terminating sessions 87
EXECRPL 403
EXLST 404
GENCB 407
global values in 274
how described 371
how to use 274
INQUIRE
permissible option codes 413
INTRPRET 426
ISTGLBAL 430
list of general-use 891
list of product-sensitive 891
MODCB 431
NIB 433
OPEN
forms of 446
OPNDST 89, 453
OPNSEC 96, 460
RCVCMD 463
RECEIVE
major options 467
macroinstruction *(continued)*
REQSESS 95
RESETSR
major options 482
RPL 487
RPL-based 320
SEND
major options 514
SENDCMD 529
SESSIONC
major options 541
SETLOGON 543
SHOWCB 551
SIMLOGON 555
specified in MVS 320
summary description 20, 375
task association 311
TERMSESS 99, 560
TESTCB 565
versus the authorized path function 302
macroinstruction global variables
declaring and setting 274
types of
component-ID 274
function-list 274
release-level 274
main storage, facility for obtaining 408
mainframe
education xxix
mainline program 31
maintenance-related information 18
management services 337
manipulating control blocks
GENCB 407
MODCB 431
SHOWCB 551
TESTCB 565
manipulative control block
description 267
manipulative macroinstructions
defined 22
description 19
errors and special conditions 278
forms of 22, 865
function of 14
GENCB
basic function of 22
how to use 268
list of 267
MODCB
basic function of 22
how to use 269
operands used with 857, 867
return codes 278, 855
SHOWCB
basic function of 22
FDBK2 field 651
how to use 270
TESTCB
basic function of 22
FDBK2 field 651
how to use 271
maximum RU size 183, 200
message and command header
DSECT example 888
format and DSECT 888

MF operand
CLOSE 392
GENCB 410
MODCB 432
SHOWCB 553
TESTCB 568
MIDDLE value on CHAIN operand
RPL 494
SEND 519
migration considerations
coding guidelines 35
MODCB
advantage of 267
basic function of 22, 431
errors and special conditions for 278
how to use 269, 431
MODE operand 437
MODEENT macroinstruction
LANG operand 241
MODENAME 139
MODIFY QUERY command from a POA 889
multiple address space 312
multiple control block generation 409
multiple monitor environment 363
multiple tasks
each with its own ACB 298
multitasking a program 297
use of multitasking 296
using the same ACB 298
multitasking 296
multithread application program
characteristics of 36
definition of 36
multithread operation
definition 36
VTAM facilities for 37
MVS operating system
31-bit addressing 319
asynchronous exit routines 307
authorization criteria 296
authorized path 300
authorized path macroinstructions 301
closing the application program 321
data facility storage management system(DFSMS) 50
interfacing with an application program 50
multiple addresses 312
opening the application program 320
special exits 307
specifying macroinstructions 320

N

NAME field in NIB 443
NAME operand 437
national language support (NLS)
language code settings 117
NAU (network addressable unit)
definition of 1
NBB value on BRACKET operand
RPL 493
SEND 518
NCP (Network Control Program)
basic function of 16
negative response
receiving
examples of in RECEIVE macroinstruction 474
exception response requested 154
negative response *(continued)*
requesting 154
sending 154, 394
transferring sense fields before sending 292
with RECEIVE 473
with RPL 503
with SEND 525
network addressable unit (NAU)
definition of 1
Network Control Program (NCP)
basic function of 16
network operator macroinstruction
DISPLAY 529
MODIFY 529
RCVCMD 463
REPLY 529
SENDCMD 529
VARY 529
network services request unit
deliver request unit 345
embedded 348
forward request unit 344
network upgrades
coding guidelines 35
network-qualified names support 65
NIB (node initialization block)
contents 115
control block 754
fields set by VTAM
CID 442
CON 443
DEVCHAR 443
NAME 443
NETID 443
NIBNACLQ 443
NIBPSDFA 444
NIBPSDFS 443
NIBPSPLU 443
NIBPSRSP 443
NIBRPARM 444
generating with INQUIRE 271
ISTDNIB DSECT for 272, 754
ISTDPROC DSECT for 761
ISTDPROC macroinstruction for processing options in 273
ISTDVCHR DSECT for 756
ISTDVCHR macroinstruction for device characteristics field in 273
PROC options 161
USERFLD field of 37
using 115, 116
NIB field versus ARG field 497
NIB generation for logical unit groups 417
NIB list
creating 433
defining 119
explanation of 433
NIB macroinstruction
basic function of 21
BNDAREA 124
information specified in 433
LOGMODE 124
specifying affinity ownership condition 436
specifying generic name of application 436
using 433
NIB operand
MODCB 432
RPL 497

NIB operand *(continued)*
SHOWCB 552
TESTCB 567
NIB-oriented exit routines 433
NIBLEN value of LENGTH operand 410
NIBNACLQ field in NIB 443
NIBPSDFA field in NIB 444
NIBPSDFS field in NIB 443
NIBPSPLU field in NIB 443
NIBPSRSP field in NIB 443
NIBRPARM field in NIB 444
NIBTK option code 499
NLS (national language support)
language code settings 117
NO value
BRANCH operand 494, 518, 535
LISTEND operand 436
no response 154
node initialization block (NIB)
contents 115
control block 754
fields set by VTAM
CID 442
CON 443
DEVCHAR 443
NAME 443
NETID 443
NIBNACLQ 443
NIBPSDFA 444
NIBPSDFS 443
NIBPSPLU 443
NIBPSRSP 443
NIBRPARM 444
generating with INQUIRE 271
ISTDNIB DSECT for 272, 754
ISTDPROC DSECT for 761
ISTDPROC macroinstruction for processing options in 273
ISTDVCHR DSECT for 756
ISTDVCHR macroinstruction for device characteristics field in 273
PROC options 161
USERFLD field of 37
using 115, 116
non-negotiable BIND 91
normal environment for VTAM application programs 30
normal operating system environment
description 30
dispatching priorities 32
normal-flow request
and responses, summary table 161
definition of 158
expedited-flow 160
quiescing the sending of 206
sent sequentially 158
sequence numbers in 163
normal-flow requests and responses (DFSYN)
in RECEIVE 472
in RPL 504
normal-flow send/receive mode 329
Notify request
definition of 106
examples of 245
format of 247
received by an application program 244
notifying a session partner of a request for a session 89
NQN support 65
NSEXIT exit routine
address 406
executing
in SRB mode 307
in TCB mode 307
formats of RUs received by 244
registers upon entry 253
using 104, 244
NSPE request
definition of 105
examples of 245
format of 246
received by an application program 244

O

OBSQAC operand 507, 536
OBSQVAL operand 496, 537
OFLAGS field
contents 383
testing 393
OFLAGS operand 568
ONLY value on CHAIN operand
RPL 494
SEND 519
open destination 453
OPEN macroinstruction
ACB data space 375
ACB storage allocation 445
basic function of 10, 23
errors and special conditions
organization of information 277
example 57, 65
forms of 446
OPEN ACB errors 449
using 444
vector lists 58
where to issue 66
OPEN operand of TESTCB 568
opening a logon queue 543
opening a program
description 55
in MVS 320
opening an ACB 444
operand specification summary 857
operating system considerations
authorization 296
introduction 295
operating system differences 295
operating system environment 30
OPNDST
accepting a session 89
network-qualified names 90
acquiring a session 90
basic function of 23
BIND request 91
coding information for 453
completion information for 459
description 453
establishing an LU-LU session 7
examples of 575
general relationship to RPL and NIB 575
OPTCD=ACCEPT
determining session parameters for 125
OPTCD=ACQUIRE
determining session parameters for 125
requirements 453

OPNDST *(continued)*
to acquire logical unit characteristics 575
use
in accepting pending-active sessions 453
in establishing sessions 89
in supplying dial parameters 122
OPNDST request
level of cryptography 134
OPNSEC
basic function of 23, 96, 460
network-qualified names and 96
PROC options 97
requirements 460
using 460
OPNSEC request
level of cryptography 136
OPTCD operand 498
option codes 498
options, processing 438
ORDRESP value on PROC operand
as used with LMPEO 199
NIB 440
outage notification 110
outbound sequence number
action code 537
description 537
outbound STSN indicators 537
output
responded 173
scheduled 172
scheduling 38
overlength data
handling 182

P

pacing count 798
parameter lists for exit routines 234
parameters, session 793
PARMS field 380
PARMS operand
ACB 380, 381
CLSDST
communicating with application program 94
passing a logical unit to an application program 245
setting system and user sense with existing fields 400
RPL 400, 503
TERMSESS 564
PASS value on OPTCD operand
CLSDST 399
PASSWD operand
ACB 382
password protection 382
path, authorized
MVS 300
pending active session, definition of 83
performance monitor interface 358
data collection dynamics 362
data collection mechanism 358
definition requirements for initialization 358
implications of multiple monitor environment 363
performance data types 363
request unit formats 364
sense codes 368
termination 363
PERSESS value of OPTCD operand
INQUIRE 421
persistent LU-LU session support
and TPEND exit 69
and VARY command 68, 70
application states of 67
CLOSE ACB flow 67
OPEN ACB flow 67
persistence capable, definition of 67, 381
persistence enabled, definition of 67, 548
restoring sessions pending recovery
control vector hex 29 138, 747
data tracking 137
PSTIMER 68, 548
reissuing the OPEN ACB 137
taking over a failed application 69, 137
using INQUIRE 139
using OPNDST 139
physical unit (PU)
definition of 1
in SNA network 2
SSCP-PU session 5
PLU (primary logical unit)
definition of 4
name 816
name length 816
POA (program operator application)
authorization 878
closing 882
data exchanged with VTAM 883
introduction 875
limiting queued messages 883
macroinstructions
RCVCMD 26, 882
SENDCMD 26, 882
message and command header, format and DSECT of 888
message header 884
message ID with VTAM
receiving data 885
sending data 887
method for writing 878
operational characteristics 880
POAQLIM 883
programming requirements 881
VTAM operator commands 879
MODIFY QUERY 889
positive response
meaning of 154
requesting and receiving 154
sending 524
type 1 and 2
with RECEIVE 466
with SEND 524
POST operand
RPL 503
SEND 12, 521
posting of return codes 651
prerequisite information xxix
presentation services 799
preventing logon request queuing
after OPEN processing 543
during OPEN processing 378
PRID (procedure-related identifier)
CNM application program 342
primary logical unit (PLU)
definition of 4
name 816
name length 816
PROC operand 438

procedure-related identifier (PRID)
  CNM application program 342
processing options
  of a session 116
  specification 438
processing part of an application program 16
program operator application (POA)
  authorization 878
  closing 882
  data exchanged with VTAM 883
  introduction 875
  limiting queued messages 883
  macroinstructions
    RCVCMD 26, 882
    SENDCMD 26, 882
  message and command header, format and DSECT of 888
  message header 884
  message ID with VTAM
    receiving data 885
    sending data 887
  method for writing 878
  operational characteristics 880
  POAQLIM 883
  programming requirements 881
  VTAM operator commands 879
    MODIFY QUERY 889
program structure
  coding guidelines 33
programming considerations
  general 33
protocol
  bracket 213, 216
  change-direction 212
  CNM (communication network management) 342
  half-duplex 211
  LU-LU session 342
  predefined sets of 795
  quiesce 210
  Systems Network Architecture (SNA) 201
PSERVIC operand 799
PU (physical unit)
  definition of 1
  in SNA network 2
  SSCP-PU session 5

Q

Q value on OPTCD operand 501
QC (Quiesce Complete) request
  example 207
  SEND 514, 526
QEC (Quiesce at End-of-Chain) request
  example 207
  SEND 514, 526
queued response notification 182
queued session, definition of 83
queuing a request
  for a session with an SLU 110
queuing of session-establishment request 83
quick closedown 264
Quiesce at End-of-Chain (QEC) request
  example 207
  SEND 514, 526
Quiesce Complete (QC) request
  example 207
  SEND 514, 526
quiesce protocol
  description 206, 207
QUIESCE value on OPTCD operand
  RPL 501
  SETLOGON 544, 548
quiescing
  of an application program by an LU 207
  protocol 210
  using 206

R

RCVCMD
  basic function of 26
  using 463
Ready to Receive (RTR) request
  using 514, 526
reason code
  OPTCD operand of the RPL macroinstruction 500
  PARMS operand of TERMSESS macroinstruction 564
  PARMS operand of the RPL macroinstruction 503
reason code (FDB2) 244
RECEIVE
  basic function of 11, 24
  continue-any mode for 176
  continue-specific mode for 176
  handling overlength data in 182
  keeping or truncating overlength data for 182
  major options 467
  processing 375
  receive-any operation 39
  requirements 466
  to receive a response
    RTYPE=DFASY 178
    RTYPE=RESP 154, 178
  using 157, 466
  versus DFASY or RESP exit routine 178
  versus EXLST exit routines 178
receive-any operation
  versus receive-specific 174
receiving a BIND request
  SCIP exit routine 102
receiving an UNBIND request, SCIP exit routine 103
receiving requests and responses 466
RECLEN field in an RPL 183
RECLEN field or operand 503
record application program interface
  general description 14
RECORD operand value 437
recovery action return codes
  general meanings 280
recovery routines 324
register contents
  control block address 853
  exit routine address 853
  return codes 853
  RPL address 853
register usage
  LERAD exit routine 291
  summary 853
  SYNAD exit routine 291
registers set by VTAM 633
Release Quiesce (RELQ) request
  sending 514
  using 207
RELEASE value on OPTCD operand
  CLSDST 399

release-level macroinstruction global variables 274
release-level vector 62
releasing logical units, method of 394
RELQ (Release Quiesce) request
  sending 514
  using 207
RELREQ exit routine
  entry to 254, 406
  executing
    in SRB mode 307
    in TCB mode 307
  for notifying a program of release request 253
  parameters passed to 254
  possible actions in 253
  registers upon entry 254
RELREQ operand
  EXLST 406
RELRQ operand
  RPL 501
REPLY command 529
replying to VTAM messages 529
REQ operand value
  following RECEIVE
    in request to be sent 519
    to indicate a RECEIVE 472
    to show indicator status 507
REQSESS
  basic function of 23
  determining session parameters for 125
  using 95
request
  code 508
  modes 495
  normal-flow 495
request and response exchanges 693
request header (RH)
  chain indicators 186
  generated by LMPEO 185
  in SNA 151
  indicators handled by LMPEO 186
  location of the initial RH 185
request level error isolation 322
request parameter list (RPL)
  AREA field in 257
  basic function of 21
  control block 487, 766
  description 114
  error and special condition information in 279
  FDB2 field in 279
  fields set by VTAM 505
  fields, applicability of (per macroinstruction) 510
  IFGRPL DSECT for 272
  ISTUSFBC DSECT for 774
  macroinstruction 487
  operand
    MODCB macroinstruction 431, 432
    SHOWCB macroinstruction 551
    TESTCB macroinstruction 565
  RESPOND field in 576
  RTNCD field in 280
  sense fields in 280
Request Recovery (RQR) request
  need for SCIP exit routine to process 256
  summary of 532
  using 167
request unit names, CNM interface 348
request unit size
  description 797
  maximum size 797
request/response unit (RU)
  classified by VTAM 179
  communication network services, format 351, 352
  definition 151
  format
    deliver 348
    forward (MVS) 344
  initiate load request format 350
  load status request format 350
  network services
    embedded 350
    not embedded 351
  translate-inquiry request (TR-INQ), format 352
requests
  and responses 151
  asynchronous 170
  chaining 201
  chaining (example) 202
  Chase request 216
  contents 11, 152
  example of sending and receiving 153
  exchanging requests and responses 152, 157
  overlength data in 182
  quiescing the sending of 206
  received from a logical unit 154
  receiving from LUs 11
  request and response modes 205
  responses to 153
  sending 12
  sequence number in 163
  sequence relationship between normal-flow and expedited flow 158
  starting and stopping the flow of 165
  synchronous 169
  transmitted on expedited flow 161
  transmitted on normal-flow 161
requests and responses, CNM interface 341
RESETSR
  basic function of 24
  major options 482
  using 481
resetting
  a session's CA-CS mode 482
  RECEIVE 483
resource ID vector 63
resource-information 383
resource-information vector list (ACBRIVL) 383
RESP exit routine
  advantages and disadvantages of 254
  examples of
    in logic of sample program 1 573, 577
  executing
    in SRB mode 307
    in TCB mode 307
  how to use 254
  how VTAM handles RESP input 181
  parameters passed to 255
  read-only RPL provided for 255
  registers upon entry 255
  request and response units 160
  sample program 2 logic 621
  scheduled when an expedited-flow request is received 178
  scheduling of, after receiving a response 155, 178
  specifying in an ACB or NIB 229, 255

RESP operand
EXLST 407
RESETSR 486
RPL 503
RESP request and response units
definition of 160
RESP value on POST operand
SEND 522
RESP value on RTYPE operand
RECEIVE 472, 473
SEND 520
RESPLIM operand 441
RESPOND field
RPL 503, 509
SEND 521, 522
responded output
using 173
with SEND 522
response
contents 153
exchanging 152
limit 441
receiving 13, 153
request and response modes 205
requesting 153
sending 12
sequence number in 163
starting and stopping the flow of 165
to a normal-flow request 158
to an expedited-flow request 158
types of 153
using the 3270 terminal 330
ways of receiving
RECEIVE RTYPE=DFSYN 161, 178
RECEIVE RTYPE=RESP 161, 178
RESP exit routine 161, 178
without RECEIVE RTYPE=RESPONSE 225
response header indicators in SNA 151
response modes 205
RESPX processing option 440
resumption of LOGON exit routine scheduling 543
retriable completion
handling of 293
retrying RPL-based requests 403
return codes
combinations 655
for CLOSE 277, 393
for manipulative macroinstructions 278
errors and special conditions 278
registers 0, 15 855
for OPEN 277, 448
for RPL-based macroinstructions
FDB2 field 279
registers 0, 15 279
reuse of RPLs 502
RTNCD field 279
posting 651
recovery action 280
sense fields for RPL-based macroinstructions 651
RFC (request for comments)
accessing online xxxi
RH (request header)
chain indicators 186
generated by LMPEO 185
in SNA 151
indicators handled by LMPEO 186
location of the initial RH 185
RMODE specifications 319
RPL
basic function of 487
using 487
RPL (request parameter list)
AREA field in 257
basic function of 21
control block 487, 766
description 114
error and special condition information in 279
FDB2 field in 279
fields set by VTAM 505
fields, applicability of (per macroinstruction) 510
IFGRPL DSECT for 272
ISTUSFBC DSECT for 774
macroinstruction 487
operand
MODCB macroinstruction 431, 432
SHOWCB macroinstruction 551
TESTCB macroinstruction 565
RESPOND field in 576
RTNCD field in 280
sense fields in 280
RPL exit routine
addressing mode 321
compared to ECB-posting 219
definition of 29, 219
example
of using 219
of VTAM scheduling 172
executing
in SRB mode 309
in TCB mode 309
how it works 219
how to use 256
list of special purpose routines 29
parameters passed to 256
registers upon entry 256
specifications and functions of 219
using 42
using instead of ECB-posting
examples 170
illustration of 42
problems avoided by 231
using with ECBs 220
versus ECB posting 42
with asynchronous operations 170
RPL operand
CHECK 389
CLSDST
AAREA 396
ACB 397
AREA 397
ARG 397
BRANCH 397, 416, 428
ECB 397, 416, 428
NIB 398
OPTCD 398, 399, 400
PARMS 400
RECLEN 400
RPL-based macroinstruction
errors and special conditions 279, 280
OPNDST 23
OPNSEC 23
return codes for 279
using 23
RPLC value of PROC operand 438

RPLLEN value of LENGTH operand 410
RQR (Request Recovery) request
need for SCIP exit routine to process 256
summary of 532
using 167
RQR value of CONTROL operand 535
RRN value of RESPOND operand 523
RSHUTD request
sending 514
RTNCD field 279, 509
RTR (Ready to Receive) request
using 514, 526
RTYPE operand
RECEIVE
example 472
explicit 178
indicating type of response received or expected in return 473
table 467
RESETSR 481
RPL 504, 507
RU (request/response unit)
classified by VTAM 179
communication network services, format 351, 352
definition 151
format
deliver 348
forward (MVS) 344
initiate load request format 350
load status request format 350
network services
embedded 350
not embedded 351
translate-inquiry request (TR-INQ), format 352

S

sample programs
sample program 1
data interface with logical units 580
how SAMP1 code relates to sample program 1 579
logic of 573
notes on LERAD and SYNAD exit routines 584
notes on LOGON exit routine 583
notes on LOSTERM exit routine 585
notes on mainline program 582
notes on RESP exit routine 584
what it does 579
sample program 2
logic of 605
sample program 3
assembler language code 628
logic of 627
using authorized path 627
save area, requirement for 234
SBI (stop bracket initiative) 526
SBI value on CONTROL operand 527
SCHED value on POST operand
RPL 503
SEND 521
scheduled output 172, 521
scheduling priority of I/O requests 503
SCIP exit routine
address 407
basic function of 256
entered as a result of
BIND request 258
SCIP exit routine *(continued)*
entered as a result of *(continued)*
Clear request 257
RQR request 257
SDT request 257
STSN request 258
UNBIND request 260
executing
in SRB mode 307
in TCB mode 307
read-only RPL provided to 256
registers upon entry 262
resynchronization of sequence numbers in 258
specifying in an ACB or NIB 229
using 102, 256
SDT (Start Data Traffic) request
basic function of 7
in request flow 532
indication 117
need for SCIP exit routine to process 256
receiving 407
sending 532, 535
using 164
secondary logical unit (SLU)
definition of 4
SEND
basic function of 24
examples of
for asynchronous operations 169
for synchronous operations 169, 576
major options 514
OPTCD=LMPEO
handling of negative response 190
POST operand 12, 522
POST=RESP 170
POST=SCHED 169, 170
RESPOND operand 156, 216
scheduling 169
specific mode for 174
specifying ECB posting in 170
specifying execution of RPL exit routine in 170
STYPE=REQ 169
STYPE=RESP 12, 154
using 157, 514
SEND operation (example) 200
SEND options 514
SENDCMD
basic function of 26, 529
using 529
sending network operator commands 529
sending requests and responses 514
sense code 500
sense fields and return codes for RPL-based macroinstructions 651
sense information
for a 3270 device 332
received at the application program 332
SEQNO field
for RECEIVE 473
for RPL 504, 509
for SEND 164, 525
how used with LMPEO 189, 509
sequence numbers
for RECEIVE 473
for RPL 504
for SEND 525
for STSN commands 537

sequence numbers *(continued)*
handling during LMPEO operation 189
in requests and responses 163
of normal-flow RUs 331
resetting to zero with Clear request 164
resynchronization of
general description 163
serialization of execution 309
session
accepting 89
acquiring 90
active 81, 84
address space 316
available 82
communication
macroinstructions 24
connected 81
control commands 532
cryptographic 131
cryptographic control 815
determining parameters for
INQUIRE 124
OPNDST OPTCD=ACCEPT 125
OPNDST OPTCD=ACQUIRE 125
REQSESS 125
SIMLOGON or CLSDST OPTCD=PASS 125
disabled 81
enabled 81
establishment
control block 115
macroinstructions 23
role of exit routines 100
stages of 5, 7, 83
with logical units 11, 27, 81
identifying 168
initiation
error recovery procedure 44
exit routines involved in 107
initiate request 6
initiate request types 85
use of SIMLOGON 87
using generic resource name 78
with logon information 6
limit 82
LU-LU 4
LU-LU session protocols 342
major communication alternatives 169
outage
codes 111
exit routines involved in 108
summary 105
parallel 4
parameters associated with CINIT 127
SSCP-LU 4
SSCP-LU session 6
SSCP-PU 4
SSCP-PU session 5
SSCP-SSCP 4
termination
by one of the session participants 109
control block 115
macroinstructions 23
stages of 84
terminate request types 86
with logical units 13, 28
types 3
session awareness data buffer 356
session control
transmission services profile 796
session establishment macroinstruction
CLSDST 23
OPNDST 23
OPNSEC 23
REQSESS 23
SESSIONC 24
SIMLOGON 23
TERMSESS 24
session instance identifier 139
session level error isolation 322
session monitor 338
session outage notification (SON) 105, 110
session parameter
3270, LU type 0 333
agreement 122
building and using in a BIND area 130
defining and naming (logon mode) 123
defining sets 123
example of
associated with a CINIT 127
in a BIND area 130
processing of by an application program 127, 128
specifying 34, 793
using 123
session parameter fields
format (BIND image) 793
function management 795
profile 795, 819
session qualifier pair 139
session state control vector 747
session termination
and Chase request 217
by a secondary application program 560
by one of the session participants 109
stages of 84
terminate request types 86
SESSIONC
and XRF 532
for communication 24
for session establishment 24
in sending SDT requests 157
options 541
using
general description 532
to reject a BIND request 98
with CONTROL=BIND 98
network-qualified names 98
SESSIONC command 532
SESSKEY value on OPTCD operand
RPL 498
SESSPARM value on OPTCD operand
INQUIRE
description 423
possible restrictions 412
source of session parameters 414
RPL 498
Set and Test Sequence Numbers request (STSN)
need for SCIP exit routine to process 258
possible responses to 532
receiving 407
sending 542
using 167, 532
SETLOGON
ACB MACRF operand, interaction with 83
basic function of 25, 543

SETLOGON *(continued)*
examples of use 575
HOLD 84
LOGON exit routine scheduling 543
START 84
using 543
shortcut keys 899
SHOWCB
advantage of 267
basic function of 22, 551
errors and special conditions for 278
use and examples of 270
using 551
SHUTD (Shutdown Complete request)
in data-flow-control request 518
on CONTROL operand of SEND macroinstruction 527
on STYPE operand of SEND macroinstruction 527
Shutdown Complete request (SHUTD)
in data-flow-control request 518
on CONTROL operand of SEND macroinstruction 527
on STYPE operand of SEND macroinstruction 527
SIGDATA operand
RPL 504, 509
SEND 525
signal request 213
SIGNAL value on STYPE operand 526
SIMLOGON
basic function of 23, 555
defined 87
OPTCD=CONALL 90
OPTCD=CONANY 91
OPTCD=PASS
determining session parameters for 125
using 87, 555
simulated logon requests 555
single task with multiple ACBs 299
single-thread application program
characteristics of 36
definition of 36
example of
sample program 1 573
single-thread operation 36
SLU (secondary logical unit)
definition of 4
SNA (Systems Network Architecture)
key concepts for VTAM 1
LU (logical unit) 2
NAU (network addressable unit) 1
protocols
for ensuring orderly communication 209
specifying 217
using 201
PU (physical unit) 1
sense fields 679
SSCP (system services control point) 1
task association
exit routine 310
macroinstruction 311
SNA network interconnect vectors
host-subarea-network-name vector 63
host-subarea-PU-network-address vector 63
host-subarea-PU-network-name vector 63
maximum-subarea vector 64
network-name vector 63
SSCP-name vector 63
SNA protocol specifications 897
softcopy information xxix
SON (session outage notification) 105, 110
SON type codes 260
SONCODE 564
sources of SNA Initiate and Terminate requests 85
SPEC value on OPTCD operand 502
specific-mode
in a SEND or RECEIVE operation 174, 176
used to handle an inquiry 175
SRBEXIT operand 307
of ACB 381
SSCP (system services control point)
in SNA network 2
LU-LU session 4
role of, in VTAM 1
SSCP-LU session 4, 6
SSCP-PU session 4, 5
SSCP-SSCP session 4
SSENSEI field 510
SSENSEO field
for CLSDST request 401
for Logical Unit Status (LUSTAT) request 504
to represent a major class of error 525, 538
with requests and negative responses 510
SSENSMI field 510
SSENSMO field
for CLSDST request 401
for Logical Unit Status (LUSTAT) request 504
with requests and negative responses 510
with SNA-defined errors, how coded 525, 538, 564
stages of session establishment 83
stages of session termination
definition of 84
STANDARD value on CODESEL operand 519
Start Data Traffic request (SDT)
basic function of 7
in request flow 532
indication 117
need for SCIP exit routine to process 256
receiving 407
sending 532, 535
using 164
START value on OPTCD operand
RPL 501
SETLOGON 548
stop bracket initiative (SBI) 526
STOP value on OPTCD operand
RPL 501
SETLOGON 548
stopping logon request queuing 543, 550
storage key
ACB storage allocation 376
storage limitation
ACB data space 375
STSN (Set and Test Sequence Numbers request)
need for SCIP exit routine to process 258
possible responses to 532
receiving 407
sending 542
using 167, 532
STSN operand value 532
STYPE operand
RPL 505, 525
subtasks
using separate ACBs 298
using the same ACB 298
supervisor state, for use of authorized path 302

symbolic name
of a logical unit 115, 437
of an application program 375
SYN (synchronous handling) 502
SYN operand value 502
SYNAD exit routine
addressing mode 238, 263
advantage of 263
basic function of 13
coding 263, 290
coding, special requirements 230
executing
in SRB mode 308
in TCB mode 308
given control 321, 407
how to use 263
linkage conventions for 234, 263
not reentrant 234
parameters passed to 263
purpose of 263
reentrant 234
register usage 291
registers upon entry 263
synchronous handling (SYN) 502
synchronous operation
advantages of 45
characteristics of 40
errors for 282
returning to application under same SRB 499
versus asynchronous 169
synchronous request 169
syntax diagram, how to read xxv
SYSTEM operand value 441
system services control point (SSCP)
in SNA network 2
LU-LU session 4
role of, in VTAM 1
SSCP-LU session 4, 6
SSCP-PU session 4, 5
SSCP-SSCP session 4
system-sense information
sending 525, 538, 564
system-sense modifier information
sending 525, 538, 564
Systems Network Architecture (SNA)
key concepts for VTAM 1
LU (logical unit) 2
NAU (network addressable unit) 1
protocols
for ensuring orderly communication 209
specifying 217
using 201
PU (physical unit) 1
sense fields 679
SSCP (system services control point) 1
task association
exit routine 310
macroinstruction 311

T

target resource name 344
task association
description 310
of exit routines 310
of macroinstructions 311
task level error isolation 323
task termination 323
TCBEXIT operand 307
TCP/IP
online information xxxi
Technotes xxix
terminals
characteristics of LU type 0 3270 328
differences among LU type 0 3270 335
flow
deliver 339
forward 339
Terminate Cleanup request 84
Terminate Forced request 84
Terminate Orderly request 84
terminating affinities 79
terminating sessions with logical units 390
generic resources 79, 543
termination
address space 323
task 323
TERMSESS
basic function of 24
using 99, 560
network-qualified names 99
test request RUs, 3270 Information Display System
actions taken by the network 335
TESTCB
advantage of 267
basic function of 22
errors and special conditions for 278
testing OFLAGS field 277
use and examples of 271
using 565
testing
control block fields 565
multiple field values 565
processing options or option codes 565
third party Initiate and Terminate requests 85, 86
THRDPTY operand 400, 503
timeout CNM request unit format 352
TOPLOGON operand 414, 426
TPEND exit routine
closedown of VTAM 407
closing an application program 74
entry to, after HALT commands 263
executing
in SRB mode 307
in TCB mode 307
parameters available on entry to 265
reason codes 264
registers upon entry 265
user exit queues 309
with reason code 8 32
TPEND operand 407
trademark information 911
transmission control 331
transmission services
profile 795
usage field 797
type code
in PARMS operand of TERMSESS macroinstruction 564
UNBIND in RPL 500
UNBIND used on UNBIND RU 503

U

UNBIND request
  basic function of 7
  need for SCIP exit routine to process 256
  receiving 103
  SON codes 93, 260
  TERMSESS restrictions 100
USENSEI field 510
USENSEO field
  errors indicated by 527, 564
  for CLSSDT request 401
  for Logical Unit Status (LUSTAT) request 505
  how coded 527, 564
  when RPL-based macroinstruction is completed 510
user data 816
user data length 816
user exit queues 309
user RH option (USERRH)
  description 196
  example of using 200
  handling the Sense Data Included (SDI) indicator 199
  operating considerations 196
  operation for inbound RUs 198
  operation for outbound RUs 197
  relationship to NIB 199
user sense information
  receiving 402
  sending 527, 564
USERFLD field of the NIB 37
USERFLD operand
  of ACB 381
  of NIB 441
USERRH (user RH option)
  description 196
  example of using 200
  handling the Sense Data Included (SDI) indicator 199
  operating considerations 196
  operation for inbound RUs 198
  operation for outbound RUs 197
  relationship to NIB 199
USERRH field in the RPL
  relationship to the request/response header 198
using logon mode names and session parameters 123
using network-qualified names support 65
USS Messages
  national language code values 117

V

VARY command 529
vector list
  access method support vector 62
  component-ID vector 62
  function-list vector 62
  release-level vector 62
  resource-ID vector 63
VTAM
  domain 16
  exit routines 19
  FRR (functional recovery routine) 324
  general programming considerations 33
  interfacing with an application program 50
  keyword operands 19
  language 19
  macroinstruction differences
    across operating systems 295
VTAM *(continued)*
  macroinstructions
    conventions and descriptions 371
    summary of 20
  manipulative macroinstructions 19
  scheduling output 38
  SNA concepts 1
  special programming considerations 295
  VTAM-initiated HALT 76
VTAM, online information xxxi

W

WAREA operand 411

X

XRF (extended reference facility)
  and SESSIONC 532
  programming 149
  session requests 91
  terminating sessions 74
XRF session activation control vector (MVS only) 845

Y

YES value
  BRANCH operand 494, 518, 535
  LISTEND operand 436

Z

z/OS Basic Skills Information Center xxix
z/OS, documentation library listing 913

Index for Communications Server: SNA Programming

Special characters

&ISTGLRL global variable
  declared and set 274

Numerics

0 value on LOGMODE operand 436
31-bit addressing 51, 319
3270 display station
  characteristics of 328
  communicating with 327
  data flow control 329
  transmission control 331
3270 terminals, types of 327

A

AAREA operand 492
AAREALN operand 492
ABEND (abnormal end)
  error handling 322, 324
  of VTAM, causing entry to TPEND exit routine 263
  pattern of abnormal termination processing 278
abnormal end (ABEND)
  error handling 322, 324
  of VTAM, causing entry to TPEND exit routine 263
  pattern of abnormal termination processing 278
ACB (access method control block)
  address 447
  address operand
    of CLOSE 392
    of OPEN 446, 447
  address space 312
  allocation of storage 376
  contents 56
  control block 740
  data space 375
  ERROR field 277, 383
  error field settings
    CLOSE ACB 393
    OPEN ACB 449
  fields, set by application program
    APPLID 56, 377
    EXLST 378
    MACRF 378
    PARMS 380
    PASSWD 382
  fields, set by VTAM
    ACBAMSVL 383
    ACBPSINS 383
    ACBRIVL 383
    ERROR 383
    OFLAGS 383
  IFGACB DSECT for 272, 740
  level of error isolation 322
  macroinstruction
    CLOSE 22, 73
    definition of 21
    example 57
    OPEN 22, 55
ACB (access method control block) (continued)
  multiple 65
  operand
    of the MODCB 431
    of the RPL 487
    of the SHOWCB 551
    of the TESTCB 565
  storage key of 376
  testing OFLAGS field 277
  using 375
  using multiple ACBs within one task 299
ACB-based macroinstruction 320
ACB-oriented exit routines 406
ACBAMSVL (access-method-support vector list)
  address of 383
  format 62
ACBLEN operand value
  field name operand for TESTCB 567
  obtaining 410
ACBRIVL (resource-information vector list) 383
ACCEPT value on OPTCD operand 497
accepting a session with OPNDST macroinstruction 89
access method control block (ACB)
  address 447
  address operand
    of CLOSE 392
    of OPEN 446, 447
  address space 312
  allocation of storage 376
  contents 56
  control block 740
  data space 375
  ERROR field 277, 383
  error field settings
    CLOSE ACB 393
    OPEN ACB 449
  fields, set by application program
    APPLID 56, 377
    EXLST 378
    MACRF 378
    PARMS 380
    PASSWD 382
  fields, set by VTAM
    ACBAMSVL 383
    ACBPSINS 383
    ACBRIVL 383
    ERROR 383
    OFLAGS 383
  IFGACB DSECT for 272, 740
  level of error isolation 322
  macroinstruction
    CLOSE 22, 73
    definition of 21
    example 57
    OPEN 22, 55
  multiple 65
  operand
    of the MODCB 431
    of the RPL 487
    of the SHOWCB 551
    of the TESTCB 565

© Copyright IBM Corp. 2000, 2013

access method control block (ACB) *(continued)*
storage key of 376
testing OFLAGS field 277
using 375
using multiple ACBs within one task 299
access-method-support vector list (ACBAMSVL)
address of 383
format 62
accessibility 899
ACQUIRE parameter
explanation of 453
operand value 497
acquiring sessions with OPNDST macroinstruction 90
action code
for inbound sequence number 537
for outbound sequence number 537
active application program, testing for 417
active logical unit, definition of 81
address
31-bit 319
address space
associated 316
multiple 312
session 316
termination 323
types of 312
used for exit routine execution 316
addressability in exit routines 234
ALIAS application 350
ALT value on CODESEL operand 519
AM operand
of the ACB macroinstruction 377
of the EXLST macroinstruction 406
of the GENCB macroinstruction 409
of the MODCB macroinstruction 432
of the RPL macroinstruction 492
of the SHOWCB macroinstruction 552
of the TESTCB macroinstruction 567
AMODE specifications 319
ANY value on OPTCD operand 502
any-mode
used to handle an inquiry 175
API (application program interface)
general requirements 50
handling control blocks 51
special requirements 51
APPL statement, name of application program in 375
APPL value on SDT operand 441
application program
as a logical unit 3
authorization to OPEN 10
authorized path under MVS 300
availability of 417
closing
as a generic resource 78
description 55, 73
with CLOSE macroinstruction 13, 28
with standard HALT command 76
closing in MVS 321
coding guidelines 33, 336
communicating with logical units 151
communication part 10, 13
controlling 85
designated for CNM routing 340
easing migration and upgrades 35
identification 56, 78, 377
in relation to a terminal operator and devices 17
application program *(continued)*
in relation to logical units in a network 17
in relation to other application programs 17
in SNA network 14
interfacing with MVS and VTAM 36, 50
ISTPDCLU application program 355
ISTSWBFR (session awareness data buffer) 356
LU 6.2 components 17
LU Initiate and Terminate request
description 85, 86
TERMSESS restrictions 86
mainline part 31
major functions 18
major start functions 7
opening
description 55
relationship to ACB 27
through an ACB 10
opening in MVS 320
organizing 33, 45
processing part 16
required control blocks for 14
schematic picture of 8
serial execution 309
sharing resources among 15
terminating 390
types of instructions 15
using multiple ACB's in 65
using to manage the network 18
VTAM definition requirements 355
VTAM interfaces and interactions 355
application program interface (API)
general requirements 50
handling control blocks 51
special requirements 51
application program migration
BTAM programs, differences between BTAM and VSAM 895
from a single-domain to a multiple-domain network
INQUIRE, for a cross-domain resource 896
INTRPRET, for a cross-domain resource 896
LOGMODE names, specifying with OPNDST for a cross-domain resource 896
from prior releases of VTAM
ACB size, increase of 894
BIND, application program minor node name in 894
sequence number dependencies for LU type 0 3270 terminals 894
SNA network interconnection requirements
INQUIRE, for a cross-network resource 896
application-supplied dial parameters (ASDP)
control block
format 743
using 119
ISTASDP DSECT for 745
macroinstructions
NIB 122
OPNDST 89, 122
SIMLOGON 122
APPLID operand 56, 377
APPLID processing 377
APPSTAT value on OPTCD operand 417
AREA operand
in RPL macroinstruction 493
in SHOWCB macroinstruction 552
AREALEN operand 493
ARECLEN field in RPL 506

ARG field in RPL 506
ASDP (application-supplied dial parameters)
control block
format 743
using 119
ISTASDP DSECT for 745
macroinstructions
NIB 122
OPNDST 89, 122
SIMLOGON 122
associated address space 316
association, task 310
ASY (asynchronous handling) 502
ASY operand value 502, 521
asynchronous exit routine 31
asynchronous handling (ASY) 502
asynchronous operation
advantages of 45
characteristics of 41
errors for 279
versus synchronous 169
asynchronous request 170
ATTN operand
EXLST 406
AUTHEXIT=YES 307
authorization
of application programs 296
authorized exit routine
MVS 307
authorized path
BRANCH operand 494
coding requirements 301
definition of 300
description 302
examples 304
macroinstructions
MVS 301
versus categories of VTAM macroinstructions 302
with RPL exit routines 302
authorized path facility, coded example 627
available logical unit, definition of 82

B

batch function, communication with 17
BB (Begin Bracket) indicator
operand value
for RPL 493
for SEND 518
using 214
Begin Bracket (BB) indicator
operand value
for RPL 493
for SEND 518
using 214
BID data 138
BID request
operand value 526
sending 526
bidder, in bracket protocol 215
BIND area
BNDAREA operand 116, 124, 435
definition of 123
format and DSECT 816
BIND image
session parameter area format 793
BIND request
basic function of 7
establishing a cryptographic session 130, 131
establishing an LU-LU session 7
need for SCIP exit to process 256
negotiable 131
non-negotiable 91
OPNSEC PROC options 97
receiving 102
rejection of 528
response 97
sending 97
session parameters in 356
BIS (bracket initiation stopped) 526
BIS data 138
BIS value on CONTROL operand 526
BLK operand of GENCB macroinstruction 409
BNDAREA
for LU profiles 828, 840
ISTDBIND DSECT 821
BRACKET field
for RPL 493, 507
for SEND 518
bracket indicators 681
bracket initiation stopped (BIS) 526
brackets
bracket protocol 214
bracket state transitions at the 3270 SLU 331
protocols used in session with 3270 terminals 330
BRANCH operand 494
branching table
using TESTCB return codes 568
buffer group 194
buffer list
entry format 192
LMPEO state transitions 195
buffer-list entry (ISTBLENT)
format of 192
ISTBLENT DSECT for 746
buffer-list LMPEO states
accumulate state 193
reset state 193
split state 193
buffer-list option (BUFFLST)
buffer-list operation 192
description 191
example of 200
operating considerations 191
BUFFLST (buffer list option)
buffer-list operation 192
description 191
example of 200
operating considerations 191

C

C value on LOGMODE operand 436
CA (continue-any mode)
CA value
OPTCD operand 498
for a RECEIVE operation 173
operand value 438, 520
processing option 438
used to handle concurrent inquiries 177
versus continue-specific mode 176
cancel closedown 263

CANCEL field
for SEND 526
CANCEL request
discarding incomplete chain 202
receiving 514
cancelling RECEIVE requests 483
CEB (conditional end bracket)
in user RH (USERRH) option 197
LMPEO handling of 186
when turned on 214
CHAIN field
for RPL 494
for SEND 519
chain indicator
from initial RH chain indicators 186
chaining
using a 3270 terminal 329
chaining of data requests
bracket indicators 681
change-direction indicators 681
description 201
example of 202
chaining output routine
logic of the 3600 617
logic of the 3601 617
change direction command (CMD) indicator
using 212, 213
change-direction
indicators
sending 519
protocol
description 212, 213
CHASE operand value
for SEND 527
Chase request
ensuring all responses have been received 206
sending 527
using 216
CHECK
addressing mode 321
basic function of 25
in an RPL exit routine 219
using 387
CHNGDIR operand
RPL 494
SEND 519
CID (communication identifier)
communicating with logical units 168, 216
explanation of 115, 442
operand value 553
CIDXLATE operand value 419
CINIT (Control Initiate request)
using session parameters with 127
CINIT (Control Initiate Request)
and LOGON exit routine 6
basic function of 6
purpose 83
class of service 93, 126
CLEANUP request
as one of several session outage notification signals 110
definition of 104
examples of 252
format of 252
received by an application program 244
Clear request
need for SCIP exit routine to process 256
sending 532
Clear request *(continued)*
to stop flow of requests and responses 164, 167
CLEAR value on CONTROL operand 535
CLOSE
ACB storage allocation 390
basic function of 23
CLOSE ACB errors 393
errors and special conditions
organization of information 277
forms of
list and execute form 391
standard form 391
using 390
closedown of VTAM 407
closing a program
description 73
as a generic resource 78
in MVS 321
with CLOSE 13
closing an ACB 390
CLSDST
basic function of 23
OPTCD=PASS operand
determining session parameters for 125
using to initiate sessions 93
OPTCD=RELEASE operand 93
OPTCD=TERMQ operand 94
scope of 92
network-qualified names 93
SSENSEO 401
using 92, 394
network-qualified names with 399, 402
CMD (Change Direction Command) indicator
using 212, 213
CNM (communication network management)
ALIAS application 350
application program 337
description 337
interface
coding requirements 340
protocol and procedure 342
RU (request unit) format 342
standard headers 343
COBOL, in writing an application program 16
CODESEL operand
RPL 494, 507
SEND 519
coding
macroinstructions and exit routines 316
requirements for authorized path 301
coding guidelines
application programs 33, 336
program structure 33
coding requirements for communication network management interface 337
coding rules for multiple address space 312
communicating with logical units
introduction 151
requests and responses 151
using SNA protocols 201
using VTAM 168
communication activity
separating from other activity 297
communication identifier (CID)
communicating with logical units 168, 216
explanation of 115, 442
operand value 553

communication network management (CNM)
ALIAS application 350
application program 337
description 337
interface
coding requirements 340
protocol and procedure 342
RU (request unit) format 342
standard headers 343
communication part of an application program 16
Communications Server for z/OS, online information xxxi
COMPLETE value on I/O operand 568
completion conditions
asynchronous requests 286
component ID vector 62
CON field in NIB 443
CONALL value on OPTCD operand 498
CONANY
concepts of establishing and terminating sessions 81
value on OPTCD operand 498
condition code of TERMSESS 563
conditional connection request (Q-NQ) 501
conditional end bracket (CEB)
in user RH (USERRH) option 197
LMPEO handling of 186
when turned on 214
confidential data handling 439
contention 210
continue chain operand 189
continue-any mode (CA)
CA value
OPTCD operand 498
for a RECEIVE operation 173
operand value 438, 520
processing option 438
used to handle concurrent inquiries 177
versus continue-specific mode 176
continue-specific (CS) mode
CS value
OPTCD operand 498, 520
PROC operand 438
processing option 438
continue-specific mode
used to handle concurrent inquiries 177
versus continue-any mode 176
control block
field lengths 547
field testing 565
generating
during program execution 407
with ACB 21, 377
with EXLST 21, 378, 404
with GENCB 407
with NIB 21, 433
with RPL 21, 487
manipulating
with GENCB 22, 407
with MODCB 22, 431
with SHOWCB 22, 551
with TESTCB 22, 565
required for application program 14
setting values in 267
techniques for handling 51
using for session establishment and termination 114
control block field
length of 553
tested with SHOWCB 553
control block field *(continued)*
tested with TESTCB 568
usage, summary 633
control block fields set by VTAM 633
control block format
ACB
MVS 740
ASDP 743
BLENT 746, 761
BNDAREA (ISTDBIND) 817
EXLST 750
MTS 752
NIB 754
RH 763
RPL 766
control block formats and DSECTs 739, 789
CONTROL field
RPL 495, 507
SEND 526, 527
SESSIONC 535
Control Initiate request (CINIT)
and LOGON exit routine 6
basic function of 6
purpose 83
using session parameters with 127
control points 2
control requests and indicators, summary of 681
Control Terminate request (CTERM)
cleanup 84
forced 84
orderly 84
control vector hex 29 747
controlling flow of requests and responses 164, 205
conventions used to describe VTAM macroinstructions 371
converting a CID to a symbolic name 419
converting a symbolic name to a CID 423
COUNTS value of OPTCD operand
INQUIRE 419
CP-CP sessions 4
cross-memory API support
and CHECK 388
and CLOSE 390
and OPEN 445
function of 317
limitations for application programs 318
CRYPT operand
RPL 495
SEND 519
cryptographic session
control 815
cross-domain 133
determining level of 133
establishing 132
INQUIRE OPTCD=SESSKEY 422
session-cryptography key
cross-domain 133
single-domain 132
single-domain 132
cryptography
definition of 117
establishing requirements from the logon mode entry 135
level for OPNDST request 134
level for OPNSEC request 136
requirements 218
CS (continue-specific) mode
CS value
OPTCD operand 498, 520

CS (continue-specific) mode *(continued)*
CS value *(continued)*
PROC operand 438
processing option 438
CTERM (Control Terminate Request)
cleanup 84
forced 84
orderly 84

D

data buffer
session awareness 356
trace 357
data communication activity
dividing among several tasks 297
separating from other activity 297
data facility storage management system(DFSMS) 50
data in a message 152
data integrity damage
handling of 293
data stream
3270, LU type 0 328
DATA value on CONTROL operand
SEND 526
data-flow-control
3270, LU type 0 329
purpose 164
requests 329
declarative macroinstruction
building control blocks 320
description 21
DSECT-creating
designation of 21
default entry in the logon mode table 123
defining sets of session parameters 123
definite response
need for requesting, with SEND POST=RESP 171
definite response indication (types 1 and 2)
meaning of 156
receiving 157
requesting 157
sending 157
delayed request mode 205
delayed response mode 205
Deliver and Forward RU flow 339
Deliver request unit
flow 339
format 345, 348
interface
coding requirements 340
requests and responses 341
DEVCHAR field
in a NIB 443
value on OPTCD operand 413, 420
device characteristics field 334
device-type logical unit
Initiate and Terminate request 82, 86
DFASY exit routine
advantages and disadvantages 236
and the RPL user RH field 198
any-mode 174
executing
in SRB mode 307
in TCB mode 307
expedited requests and responses 161
how to use 236
DFASY exit routine *(continued)*
how VTAM handles DFASY input 180
list of expedited requests and responses 236
parameters passed to 180
registers upon entry 237
sample program 2 logic 624
scheduled when an expedited-flow request is received 168, 178
specific-mode 174
specifying in ACB or NIB 229
versus RECEIVE macroinstruction 158
DFASY operand
EXLST 406
RECEIVE 471
RESETSR 483
RPL 504
DFASY request and response units
definition of 160
DFASYX processing option 439
DFSMS (Data Facility Storage Management System) 50
DFSYN request and response units
definition of 160
how handled by VTAM 182
DFSYN value on RTYPE operand
RESETSR 483
RPL 504
dial usability enhancements
conditions for using 121
dial parameter list 120
format of connection subfield 745
format of CPNAME subfield 744
format of dial number subfield 743
format of direct call line name subfield 744
format of DLCADDR subfield 745
format of expanded dial information subfield 744
format of IDBLK/IDNUM subfield 744
format of ISTASDP 743
function of 119
disability 899
disabled logical unit, definition of 81
dispatching priorities 32
DISPLAY command 529
DNS, online information xxxii
Downstream Load Utility (DSLU) 338
DSECT-creating macroinstructions 21
DSECTs and control block formats 739, 789
DSLU (Downstream Load Utility) 338

E

EB (End Bracket) indicator
value on BRACKET operand
for RPL 493
for SEND 518
ECB (event control block)
field in RPL 495
posting 170
using 41
versus RPL exit routines 42, 170
enabled logical unit, definition of 81
enciphered data request
sending and receiving 218
ENCR operand on NIB macroinstruction 435
encryption facility 435
definition of 117
establishing requirements from the logon mode entry 135
level for OPNDST request 134

encryption facility *(continued)*
level for OPNSEC request 136
requirements 218
End Bracket (EB) indicator
value on BRACKET operand
for RPL 493
for SEND 518
environment errors
handling 293
ERET operand 567
ERP (error recovery procedure)
during session initiation 44
error
ACB (application program) isolation 323
request level isolation 322
session level isolation 322
task level isolation 323
ERROR field
using after CLOSE processing 393
using after OPEN processing 448
error recovery procedure (ERP)
during session initiation 44
errors and special conditions
3270, LU type 0 330, 332
analyzing
for error isolation 322
for manipulative macroinstructions 278
for OPEN and CLOSE 277
for RPL-based macroinstructions 278
asynchronous operations 283
handling of
data integrity damage 293
environment errors 293
exception requests 291
logic errors 294
negative responses 293
retriable completion 293
software errors 293
synchronous operations 282
using FDBK field 417
using LERAD and SYNAD exit routines for 290
establishing and terminating sessions
BIND and UNBIND 7
macroinstructions
CLSDST 92
OPNDST 89
OPNSEC 96
REQSESS 95
SESSIONC 98
SIMLOGON 87
TERMSESS 99
stages of 83, 85
with logical units 81
establishing cross-domain cryptographic session 133
establishing single-domain cryptographic session 132
ESTAE exit routine 324
event control block (ECB)
field in RPL 495
posting 170
using 41
versus RPL exit routines 42, 170
EX value on RESPOND operand
SEND 523
exception conditions
3270, LU type 0 332
and sense information 332
handling 291
exception requests
handling
by a PLU application 292
by an SLU application 292
excess data, saving 499
exchanging
requests 152, 157
responses 152, 157
EXECRPL
basic function of 25
using 403
EXIT operand
as internal ECB 507
instead of ECB operand with RPL exit routine 496
RPL exit routine address 488
exit routine
address space used for execution of 316
addressability and save area requirements 234
addressing mode 321
asynchronous 31
basic function of 19, 28
cautions, restrictions, and techniques for 235
creation 404
deciding how to use 228
entry procedures for 234
executing
in SRB mode 307, 309
in TCB mode 307, 309
execution of 321
exit procedures from 235
how to use 219
identified by ACB 228
identified by NIB 228
identified in RPL-based macroinstructions 219
inline 31
installation 219
parameters passed to 234
procedures for writing 231
requirements for reenterability 231, 234
RPL-specified 29
rules of coding 316
session establishment and termination 100
summary of 224
task association 310
types of
exit-list exit routines 28, 221, 224
RPL-specified exit routines 28, 219, 256
EXLLEN value on LENGTH operand 410
EXLST
basic function of 21
named in EXLST operand of ACB 222
named in EXLST operand of NIB 222
names of exit routines in 222
scheduling 178
using 404
EXLST (IFGEXLST) DSECT 751
EXLST control block 405, 750
EXLST exit routine
addressing mode 321
definition of 29, 219
executing
in SRB mode 307
in TCB mode 307
optional 228
registers upon entry 226
required 228
specification and function of 221, 224

EXLST exit routine *(continued)*
specified in ACB 222, 228
specified in NIB 222, 228
versus explicit RECEIVEs 178
EXLST operand
ACB 378
MODCB 432
NIB 435
SHOWCB 552
TESTCB 567
expedited-flow data-flow-control request
expedited-flow data-flow-control
summary of receiving 687
session-control
receiving, summary of 689
sending, summary of 687
summary of receiving 687
expedited-flow request
ability to send
during quiesced state 212
in change-direction protocol 212
and responses, table summary 161
controlling normal-flow responses 161
definition of 159
examples of
for synchronous operations 169
extracting control block fields 551
for a receive-any operation 174
for a receive-specific operation 174
sequence numbers in 163
versus normal-flow requests 160
ways of receiving
DFASY exit routine 174, 236
RECEIVE 157, 162
RECEIVE RTYPE=DFASY 178
RESETSR 482
extended reference facility (XRF)
and SESSIONC 532
programming 149
session requests 91
terminating sessions 74

F

FDB2 (reason code) 244
FDBK return code, for INQUIRE macroinstruction (OPTCD=APPSTAT) 417
FIELDS operand 552
FIRST operand
RPL 494
SEND 519
FM (function management) header
using 217
FMD (function management data)
header option 217, 499
sending of, by LMPEO 183, 500
FME operand value 503
FMH-5 138
Forward and Deliver RU flow 339
forward request unit flow 339
forward request unit, CNM interface 341
FRR (functional recovery routines) 324
function management (FM) header
using 217
function management data (FMD)
header option 217, 499
sending of, by LMPEO 183, 500
function management profile 795
function management usage field 796
function-list macroinstruction global variables 275
function-list vector 62
functional recovery routines (FRR) 324

G

gathering performance data 358
GENCB
advantage of 267
basic function of 22
errors and special conditions for 278
examples of 268
how to use 268, 407
generating control blocks
during program execution 407
generating NIBs 271
generic resource 78
determining network qualified name of real instance 414, 422
example use of SETLOGON 547
opening and closing an application program 78
specifying application name 436
terminating LU-to-application association 384, 545
GETMAIN facility 410
global values in control blocks
setting 267
testing 267
global variables
declared and set 274

H

half-duplex contention communication 211
half-duplex devices 209
half-duplex flip-flop communication 211
HALT command
action for HALT NET, CANCEL or abnormal termination 77
action for HALT NET, QUICK or VTAM-initiated HALT 76
action for standard HALT 76
for application program without TPEND exit 76
hardware monitor 338
header
for VTAM messages 464
for VTAM operator commands 529
function management 499
HOLD value on OPTCD operand
RPL 501
SETLOGON 548

I

I/O operations
cancelling 483
input 466
output 514
I/O routine
logic of the 3270 619
IBM Software Support Center, contacting xxiv
IBSQAC operand
designating type of STSN request 536
used by SESSIONC 496
when SESSIONC is completed 507

IBSQVAL operand
assigned to inbound requests 537
used by SESSIONC 496
when SESSIONC is completed 507
IFGACB DSECT for ACB 272, 740
IFGEXLST DSECT for EXLST 272, 751
IFGRPL DSECT for RPL 272
immediate request mode 206
immediate response mode 206
inactive application program 417
inbound sequence number
description 535, 542
inbound STSN indicators 536
indicators
in requests and responses
definition of 151
in a request 151
Information APARs xxix
inhibited logical unit, definition of 82
initial RH, location of 185
Initiate Load Request RU format 350
Initiate request
basic function of 6
LOGON, character-coded 85
purpose 83, 85
sources 83, 85
initiating sessions
initiate request 6
macroinstructions
SIMLOGON 87
use of LOGON exit routine 6
using generic resource name 78
inline exit routine 31, 224
input operations, receiving 466
input RU
classified by VTAM 180
INQUIRE
basic function of 25, 412
determining session parameters for 124
OPTCD=TERMS 271
permissible option codes 413
using 412
using to get a logon message 575
Internet, finding z/OS information online xxxi
interpret table, definition of 426
interpreting an input sequence 426
INTRPRET
basic function of 25, 426
using 426
isolating errors
application program 323
request 322
session 322
task 323
ISTASDP DSECT 745
ISTBLENT (buffer list entry)
format of 192
ISTBLENT DSECT for 746
ISTBLENT DSECT 746
ISTDBIND DSECT
for BNDAREA 821
using to build or examine session parameters 272
ISTDNIB DSECT for NIB 272, 754
ISTDPOHD DSECT 888
ISTDPROC DSECT for NIB 761
ISTDPROC macroinstruction for processing options fields of the NIB 272
ISTDVCHR DSECT for NIB 756
ISTDVCHR macroinstruction for device characteristics field of the NIB 272
ISTGLBAL macroinstruction
control block fields 639
how to use 430
macroinstruction global variables set by
&ISTGLCI (component-ID) 274
&ISTGLRL (release-level) 274
&ISTGLxy (function-level) 274, 275
ISTMTS DSECT 753, 762
ISTPDCLU application program 355
ISTRH DSECT 272, 763
ISTUSFBC DSECT 272, 774

K

KEEP option for overlength input data
in record-mode operations 182
value on PROC operand
NIB 472
RPL 499
keyboard 899
keyword operand
as part of the VTAM macroinstruction language 19
of the GENCB macroinstruction 410

L

LANG 117
LANGTAB 117
language code values 117
large message performance enhancement outbound (LMPEO)
Begin RU/End RU combinations 195
buffer-list option and 184, 191
chaining of data requests 201
data stream considerations 189
description 183
encrypt/decrypt facility and 184
example of using 200
exception conditions 190
handling negative response 190
handling request headers 185
handling selected RH indicators 186
operating considerations 184
operation on a message sent to an SNA LU 184
performance conditions 191
sending FM data 189
sequence number handling 189
state transitions 195
status during buffer list processing 193
LAST value on CHAIN operand
RPL 494
SEND 519
LENGTH operand
GENCB 410
SHOWCB 552
LERAD exit routine
addressing mode 321
advantages of 237
basic function of 13, 406
coding 290
coding, special requirements 230
executing
in SRB mode 308
in TCB mode 308

LERAD exit routine *(continued)*
how to use 237
linkages, conventions for 234, 237
not reentrant 234
operand 406
parameters passed to 237
purpose of 237
reentrant 234
register usage 291
registers upon entry 238
license, patent, and copyright information 903
list of NIBs
creating 433, 436
explanation of 433
LISTEND operand on NIB macroinstruction 436
LMPEO (large message performance enhancement outbound)
Begin RU/End RU combinations 195
buffer-list option and 184, 191
chaining of data requests 201
data stream considerations 189
description 183
encrypt/decrypt facility and 184
example of using 200
exception conditions 190
handling negative response 190
handling request headers 185
handling selected RH indicators 186
operating considerations 184
operation on a message sent to an SNA LU 184
performance conditions 191
sending FM data 189
sequence number handling 189
state transitions 195
status during buffer list processing 193
load operation 338
load request 338
Load Status (RU) format 350
logic errors
handling 294
logical unit (LU)
active 81, 84
available 81, 82
communicating with
application programs 151
description 28
VTAM 151, 168
communication protocol 209
connected 81
definition of 2
determining status 423
device-type 14
disabled 81
enabled 81
establishing sessions with 11, 27
examples of 14
identifying 168
in SNA network 2
primary session 4
quiescing an application program 207
receiving requests from 154
secondary session 4
SSCP-LU session 6
symbolic name 115
terminating sessions with 13, 28
logical unit presentation services
profile 799
profile 0 usage field 800
logical unit presentation services *(continued)*
profile 1 usage field 801
profile 2 and 3 usage fields 806
profile 4 usage field 808, 812
usage fields 800
Logical Unit Status (LUSTAT) request
sending 514, 526
LOGMODE operand values 126
LOGMODE operand, to identify a logon mode 436
logoff
using the 3270 terminal 335
logon
initiating 543
terminating 543
LOGON exit routine
accepting sessions in 238
advantages of 238
basic function of 6
examples of
in logic of sample program 1 575
executing
in SRB mode 307
in TCB mode 307
how to use 101
parameters passed to 238
registers upon entry 240
using INQUIRE macroinstruction in 238
versus RPL exit routine 238
with OPNDST OPTCD=ACCEPT 238
logon message
receiving 413
using the 3270 terminal 335
logon mode name
and session parameters 123
definition of 116
locating in the CINIT RU 128
using 123
logon mode, used by
CLSDST 92, 436
INQUIRE 436
OPNDST 89, 436
REQSESS 95, 436
SIMLOGON 87, 436
LOGON operand
EXLST 406
LOGON value of MACRF operand
ACB 378
LOGONMSG value of OPTCD operand
INQUIRE 413, 420
LOSTERM exit routine
entry codes for 242
executing
in SRB mode 307
in TCB mode 307
how to use 106, 241
operand 406
parameters passed to 241
reasons for entry to 241
registers upon entry 242
LU (logical unit)
active 81, 84
available 81, 82
communicating with
application programs 151
description 28
VTAM 151, 168
communication protocol 209

LU (logical unit) *(continued)*
connected 81
definition of 2
determining status 423
device-type 14
disabled 81
enabled 81
establishing sessions with 11, 27
examples of 14
identifying 168
in SNA network 2
primary session 4
quiescing an application program 207
receiving requests from 154
secondary session 4
SSCP-LU session 6
symbolic name 115
terminating sessions with 13, 28
LU-LU session protocols 342
LUSTAT (Logical Unit Status) request
sending 514, 526

M

MACRF operand
ACB 378
with SETLOGON 82
macroinstruction
ACB 375
ACB-based 22, 320
authorized path
MVS 301
categories of 20
CLOSE 387, 391
CLSDST 92, 394
declarative
ACB 21, 320
EXLST 21, 320
NIB 21, 320
RPL 21, 320
differences across operating systems 295
DSECT-creating
how to use 272, 273
list of 272
rules for coding 273, 316
establishing and terminating sessions 87
EXECRPL 403
EXLST 404
GENCB 407
global values in 274
how described 371
how to use 274
INQUIRE
permissible option codes 413
INTRPRET 426
ISTGLBAL 430
list of general-use 891
list of product-sensitive 891
MODCB 431
NIB 433
OPEN
forms of 446
OPNDST 89, 453
OPNSEC 96, 460
RCVCMD 463
RECEIVE
major options 467
macroinstruction *(continued)*
REQSESS 95
RESETSR
major options 482
RPL 487
RPL-based 320
SEND
major options 514
SENDCMD 529
SESSIONC
major options 541
SETLOGON 543
SHOWCB 551
SIMLOGON 555
specified in MVS 320
summary description 20, 375
task association 311
TERMSESS 99, 560
TESTCB 565
versus the authorized path function 302
macroinstruction global variables
declaring and setting 274
types of
component-ID 274
function-list 274
release-level 274
main storage, facility for obtaining 408
mainframe
education xxix
mainline program 31
maintenance-related information 18
management services 337
manipulating control blocks
GENCB 407
MODCB 431
SHOWCB 551
TESTCB 565
manipulative control block
description 267
manipulative macroinstructions
defined 22
description 19
errors and special conditions 278
forms of 22, 865
function of 14
GENCB
basic function of 22
how to use 268
list of 267
MODCB
basic function of 22
how to use 269
operands used with 857, 867
return codes 278, 855
SHOWCB
basic function of 22
FDBK2 field 651
how to use 270
TESTCB
basic function of 22
FDBK2 field 651
how to use 271
maximum RU size 183, 200
message and command header
DSECT example 888
format and DSECT 888

MF operand
CLOSE 392
GENCB 410
MODCB 432
SHOWCB 553
TESTCB 568
MIDDLE value on CHAIN operand
RPL 494
SEND 519
migration considerations
coding guidelines 35
MODCB
advantage of 267
basic function of 22, 431
errors and special conditions for 278
how to use 269, 431
MODE operand 437
MODEENT macroinstruction
LANG operand 241
MODENAME 139
MODIFY QUERY command from a POA 889
multiple address space 312
multiple control block generation 409
multiple monitor environment 363
multiple tasks
each with its own ACB 298
multitasking a program 297
use of multitasking 296
using the same ACB 298
multitasking 296
multithread application program
characteristics of 36
definition of 36
multithread operation
definition 36
VTAM facilities for 37
MVS operating system
31-bit addressing 319
asynchronous exit routines 307
authorization criteria 296
authorized path 300
authorized path macroinstructions 301
closing the application program 321
data facility storage management system(DFSMS) 50
interfacing with an application program 50
multiple addresses 312
opening the application program 320
special exits 307
specifying macroinstructions 320

N

NAME field in NIB 443
NAME operand 437
national language support (NLS)
language code settings 117
NAU (network addressable unit)
definition of 1
NBB value on BRACKET operand
RPL 493
SEND 518
NCP (Network Control Program)
basic function of 16
negative response
receiving
examples of in RECEIVE macroinstruction 474
exception response requested 154
negative response *(continued)*
requesting 154
sending 154, 394
transferring sense fields before sending 292
with RECEIVE 473
with RPL 503
with SEND 525
network addressable unit (NAU)
definition of 1
Network Control Program (NCP)
basic function of 16
network operator macroinstruction
DISPLAY 529
MODIFY 529
RCVCMD 463
REPLY 529
SENDCMD 529
VARY 529
network services request unit
deliver request unit 345
embedded 348
forward request unit 344
network upgrades
coding guidelines 35
network-qualified names support 65
NIB (node initialization block)
contents 115
control block 754
fields set by VTAM
CID 442
CON 443
DEVCHAR 443
NAME 443
NETID 443
NIBNACLQ 443
NIBPSDFA 444
NIBPSDFS 443
NIBPSPLU 443
NIBPSRSP 443
NIBRPARM 444
generating with INQUIRE 271
ISTDNIB DSECT for 272, 754
ISTDPROC DSECT for 761
ISTDPROC macroinstruction for processing options in 273
ISTDVCHR DSECT for 756
ISTDVCHR macroinstruction for device characteristics field in 273
PROC options 161
USERFLD field of 37
using 115, 116
NIB field versus ARG field 497
NIB generation for logical unit groups 417
NIB list
creating 433
defining 119
explanation of 433
NIB macroinstruction
basic function of 21
BNDAREA 124
information specified in 433
LOGMODE 124
specifying affinity ownership condition 436
specifying generic name of application 436
using 433
NIB operand
MODCB 432
RPL 497

NIB operand *(continued)*
SHOWCB 552
TESTCB 567
NIB-oriented exit routines 433
NIBLEN value of LENGTH operand 410
NIBNACLQ field in NIB 443
NIBPSDFA field in NIB 444
NIBPSDFS field in NIB 443
NIBPSPLU field in NIB 443
NIBPSRSP field in NIB 443
NIBRPARM field in NIB 444
NIBTK option code 499
NLS (national language support)
language code settings 117
NO value
BRANCH operand 494, 518, 535
LISTEND operand 436
no response 154
node initialization block (NIB)
contents 115
control block 754
fields set by VTAM
CID 442
CON 443
DEVCHAR 443
NAME 443
NETID 443
NIBNACLQ 443
NIBPSDFA 444
NIBPSDFS 443
NIBPSPLU 443
NIBPSRSP 443
NIBRPARM 444
generating with INQUIRE 271
ISTDNIB DSECT for 272, 754
ISTDPROC DSECT for 761
ISTDPROC macroinstruction for processing options in 273
ISTDVCHR DSECT for 756
ISTDVCHR macroinstruction for device characteristics field in 273
PROC options 161
USERFLD field of 37
using 115, 116
non-negotiable BIND 91
normal environment for VTAM application programs 30
normal operating system environment
description 30
dispatching priorities 32
normal-flow request
and responses, summary table 161
definition of 158
expedited-flow 160
quiescing the sending of 206
sent sequentially 158
sequence numbers in 163
normal-flow requests and responses (DFSYN)
in RECEIVE 472
in RPL 504
normal-flow send/receive mode 329
Notify request
definition of 106
examples of 245
format of 247
received by an application program 244
notifying a session partner of a request for a session 89
NQN support 65
NSEXIT exit routine
address 406
executing
in SRB mode 307
in TCB mode 307
formats of RUs received by 244
registers upon entry 253
using 104, 244
NSPE request
definition of 105
examples of 245
format of 246
received by an application program 244

O

OBSQAC operand 507, 536
OBSQVAL operand 496, 537
OFLAGS field
contents 383
testing 393
OFLAGS operand 568
ONLY value on CHAIN operand
RPL 494
SEND 519
open destination 453
OPEN macroinstruction
ACB data space 375
ACB storage allocation 445
basic function of 10, 23
errors and special conditions
organization of information 277
example 57, 65
forms of 446
OPEN ACB errors 449
using 444
vector lists 58
where to issue 66
OPEN operand of TESTCB 568
opening a logon queue 543
opening a program
description 55
in MVS 320
opening an ACB 444
operand specification summary 857
operating system considerations
authorization 296
introduction 295
operating system differences 295
operating system environment 30
OPNDST
accepting a session 89
network-qualified names 90
acquiring a session 90
basic function of 23
BIND request 91
coding information for 453
completion information for 459
description 453
establishing an LU-LU session 7
examples of 575
general relationship to RPL and NIB 575
OPTCD=ACCEPT
determining session parameters for 125
OPTCD=ACQUIRE
determining session parameters for 125
requirements 453

OPNDST *(continued)*
to acquire logical unit characteristics 575
use
in accepting pending-active sessions 453
in establishing sessions 89
in supplying dial parameters 122
OPNDST request
level of cryptography 134
OPNSEC
basic function of 23, 96, 460
network-qualified names and 96
PROC options 97
requirements 460
using 460
OPNSEC request
level of cryptography 136
OPTCD operand 498
option codes 498
options, processing 438
ORDRESP value on PROC operand
as used with LMPEO 199
NIB 440
outage notification 110
outbound sequence number
action code 537
description 537
outbound STSN indicators 537
output
responded 173
scheduled 172
scheduling 38
overlength data
handling 182

P

pacing count 798
parameter lists for exit routines 234
parameters, session 793
PARMS field 380
PARMS operand
ACB 380, 381
CLSDST
communicating with application program 94
passing a logical unit to an application program 245
setting system and user sense with existing fields 400
RPL 400, 503
TERMSESS 564
PASS value on OPTCD operand
CLSDST 399
PASSWD operand
ACB 382
password protection 382
path, authorized
MVS 300
pending active session, definition of 83
performance monitor interface 358
data collection dynamics 362
data collection mechanism 358
definition requirements for initialization 358
implications of multiple monitor environment 363
performance data types 363
request unit formats 364
sense codes 368
termination 363
PERSESS value of OPTCD operand
INQUIRE 421
persistent LU-LU session support
and TPEND exit 69
and VARY command 68, 70
application states of 67
CLOSE ACB flow 67
OPEN ACB flow 67
persistence capable, definition of 67, 381
persistence enabled, definition of 67, 548
restoring sessions pending recovery
control vector hex 29 138, 747
data tracking 137
PSTIMER 68, 548
reissuing the OPEN ACB 137
taking over a failed application 69, 137
using INQUIRE 139
using OPNDST 139
physical unit (PU)
definition of 1
in SNA network 2
SSCP-PU session 5
PLU (primary logical unit)
definition of 4
name 816
name length 816
POA (program operator application)
authorization 878
closing 882
data exchanged with VTAM 883
introduction 875
limiting queued messages 883
macroinstructions
RCVCMD 26, 882
SENDCMD 26, 882
message and command header, format and DSECT of 888
message header 884
message ID with VTAM
receiving data 885
sending data 887
method for writing 878
operational characteristics 880
POAQLIM 883
programming requirements 881
VTAM operator commands 879
MODIFY QUERY 889
positive response
meaning of 154
requesting and receiving 154
sending 524
type 1 and 2
with RECEIVE 466
with SEND 524
POST operand
RPL 503
SEND 12, 521
posting of return codes 651
prerequisite information xxix
presentation services 799
preventing logon request queuing
after OPEN processing 543
during OPEN processing 378
PRID (procedure-related identifier)
CNM application program 342
primary logical unit (PLU)
definition of 4
name 816
name length 816
PROC operand 438

procedure-related identifier (PRID)
  CNM application program 342
processing options
  of a session 116
  specification 438
processing part of an application program 16
program operator application (POA)
  authorization 878
  closing 882
  data exchanged with VTAM 883
  introduction 875
  limiting queued messages 883
  macroinstructions
    RCVCMD 26, 882
    SENDCMD 26, 882
  message and command header, format and DSECT of 888
  message header 884
  message ID with VTAM
    receiving data 885
    sending data 887
  method for writing 878
  operational characteristics 880
  POAQLIM 883
  programming requirements 881
  VTAM operator commands 879
    MODIFY QUERY 889
program structure
  coding guidelines 33
programming considerations
  general 33
protocol
  bracket 213, 216
  change-direction 212
  CNM (communication network management) 342
  half-duplex 211
  LU-LU session 342
  predefined sets of 795
  quiesce 210
  Systems Network Architecture (SNA) 201
PSERVIC operand 799
PU (physical unit)
  definition of 1
  in SNA network 2
  SSCP-PU session 5

Q

Q value on OPTCD operand 501
QC (Quiesce Complete) request
  example 207
  SEND 514, 526
QEC (Quiesce at End-of-Chain) request
  example 207
  SEND 514, 526
queued response notification 182
queued session, definition of 83
queuing a request
  for a session with an SLU 110
queuing of session-establishment request 83
quick closedown 264
Quiesce at End-of-Chain (QEC) request
  example 207
  SEND 514, 526
Quiesce Complete (QC) request
  example 207
  SEND 514, 526
quiesce protocol
  description 206, 207
QUIESCE value on OPTCD operand
  RPL 501
  SETLOGON 544, 548
quiescing
  of an application program by an LU 207
  protocol 210
  using 206

R

RCVCMD
  basic function of 26
  using 463
Ready to Receive (RTR) request
  using 514, 526
reason code
  OPTCD operand of the RPL macroinstruction 500
  PARMS operand of TERMSESS macroinstruction 564
  PARMS operand of the RPL macroinstruction 503
reason code (FDB2) 244
RECEIVE
  basic function of 11, 24
  continue-any mode for 176
  continue-specific mode for 176
  handling overlength data in 182
  keeping or truncating overlength data for 182
  major options 467
  processing 375
  receive-any operation 39
  requirements 466
  to receive a response
    RTYPE=DFASY 178
    RTYPE=RESP 154, 178
  using 157, 466
  versus DFASY or RESP exit routine 178
  versus EXLST exit routines 178
receive-any operation
  versus receive-specific 174
receiving a BIND request
  SCIP exit routine 102
receiving an UNBIND request, SCIP exit routine 103
receiving requests and responses 466
RECLEN field in an RPL 183
RECLEN field or operand 503
record application program interface
  general description 14
RECORD operand value 437
recovery action return codes
  general meanings 280
recovery routines 324
register contents
  control block address 853
  exit routine address 853
  return codes 853
  RPL address 853
register usage
  LERAD exit routine 291
  summary 853
  SYNAD exit routine 291
registers set by VTAM 633
Release Quiesce (RELQ) request
  sending 514
  using 207
RELEASE value on OPTCD operand
  CLSDST 399

release-level macroinstruction global variables 274
release-level vector 62
releasing logical units, method of 394
RELQ (Release Quiesce) request
  sending 514
  using 207
RELREQ exit routine
  entry to 254, 406
  executing
    in SRB mode 307
    in TCB mode 307
  for notifying a program of release request 253
  parameters passed to 254
  possible actions in 253
  registers upon entry 254
RELREQ operand
  EXLST 406
RELRQ operand
  RPL 501
REPLY command 529
replying to VTAM messages 529
REQ operand value
  following RECEIVE
    in request to be sent 519
    to indicate a RECEIVE 472
    to show indicator status 507
REQSESS
  basic function of 23
  determining session parameters for 125
  using 95
request
  code 508
  modes 495
  normal-flow 495
request and response exchanges 693
request header (RH)
  chain indicators 186
  generated by LMPEO 185
  in SNA 151
  indicators handled by LMPEO 186
  location of the initial RH 185
request level error isolation 322
request parameter list (RPL)
  AREA field in 257
  basic function of 21
  control block 487, 766
  description 114
  error and special condition information in 279
  FDB2 field in 279
  fields set by VTAM 505
  fields, applicability of (per macroinstruction) 510
  IFGRPL DSECT for 272
  ISTUSFBC DSECT for 774
  macroinstruction 487
  operand
    MODCB macroinstruction 431, 432
    SHOWCB macroinstruction 551
    TESTCB macroinstruction 565
  RESPOND field in 576
  RTNCD field in 280
  sense fields in 280
Request Recovery (RQR) request
  need for SCIP exit routine to process 256
  summary of 532
  using 167
request unit names, CNM interface 348
request unit size
  description 797
  maximum size 797
request/response unit (RU)
  classified by VTAM 179
  communication network services, format 351, 352
  definition 151
  format
    deliver 348
    forward (MVS) 344
  initiate load request format 350
  load status request format 350
  network services
    embedded 350
    not embedded 351
  translate-inquiry request (TR-INQ), format 352
requests
  and responses 151
  asynchronous 170
  chaining 201
  chaining (example) 202
  Chase request 216
  contents 11, 152
  example of sending and receiving 153
  exchanging requests and responses 152, 157
  overlength data in 182
  quiescing the sending of 206
  received from a logical unit 154
  receiving from LUs 11
  request and response modes 205
  responses to 153
  sending 12
  sequence number in 163
  sequence relationship between normal-flow and expedited flow 158
  starting and stopping the flow of 165
  synchronous 169
  transmitted on expedited flow 161
  transmitted on normal-flow 161
requests and responses, CNM interface 341
RESETSR
  basic function of 24
  major options 482
  using 481
resetting
  a session's CA-CS mode 482
  RECEIVE 483
resource ID vector 63
resource-information 383
resource-information vector list (ACBRIVL) 383
RESP exit routine
  advantages and disadvantages of 254
  examples of
    in logic of sample program 1 573, 577
  executing
    in SRB mode 307
    in TCB mode 307
  how to use 254
  how VTAM handles RESP input 181
  parameters passed to 255
  read-only RPL provided for 255
  registers upon entry 255
  request and response units 160
  sample program 2 logic 621
  scheduled when an expedited-flow request is received 178
  scheduling of, after receiving a response 155, 178
  specifying in an ACB or NIB 229, 255

RESP operand
EXLST 407
RESETSR 486
RPL 503
RESP request and response units
definition of 160
RESP value on POST operand
SEND 522
RESP value on RTYPE operand
RECEIVE 472, 473
SEND 520
RESPLIM operand 441
RESPOND field
RPL 503, 509
SEND 521, 522
responded output
using 173
with SEND 522
response
contents 153
exchanging 152
limit 441
receiving 13, 153
request and response modes 205
requesting 153
sending 12
sequence number in 163
starting and stopping the flow of 165
to a normal-flow request 158
to an expedited-flow request 158
types of 153
using the 3270 terminal 330
ways of receiving
RECEIVE RTYPE=DFSYN 161, 178
RECEIVE RTYPE=RESP 161, 178
RESP exit routine 161, 178
without RECEIVE RTYPE=RESPONSE 225
response header indicators in SNA 151
response modes 205
RESPX processing option 440
resumption of LOGON exit routine scheduling 543
retriable completion
handling of 293
retrying RPL-based requests 403
return codes
combinations 655
for CLOSE 277, 393
for manipulative macroinstructions 278
errors and special conditions 278
registers 0, 15 855
for OPEN 277, 448
for RPL-based macroinstructions
FDB2 field 279
registers 0, 15 279
reuse of RPLs 502
RTNCD field 279
posting 651
recovery action 280
sense fields for RPL-based macroinstructions 651
RFC (request for comments)
accessing online xxxi
RH (request header)
chain indicators 186
generated by LMPEO 185
in SNA 151
indicators handled by LMPEO 186
location of the initial RH 185
RMODE specifications 319
RPL
basic function of 487
using 487
RPL (request parameter list)
AREA field in 257
basic function of 21
control block 487, 766
description 114
error and special condition information in 279
FDB2 field in 279
fields set by VTAM 505
fields, applicability of (per macroinstruction) 510
IFGRPL DSECT for 272
ISTUSFBC DSECT for 774
macroinstruction 487
operand
MODCB macroinstruction 431, 432
SHOWCB macroinstruction 551
TESTCB macroinstruction 565
RESPOND field in 576
RTNCD field in 280
sense fields in 280
RPL exit routine
addressing mode 321
compared to ECB-posting 219
definition of 29, 219
example
of using 219
of VTAM scheduling 172
executing
in SRB mode 309
in TCB mode 309
how it works 219
how to use 256
list of special purpose routines 29
parameters passed to 256
registers upon entry 256
specifications and functions of 219
using 42
using instead of ECB-posting
examples 170
illustration of 42
problems avoided by 231
using with ECBs 220
versus ECB posting 42
with asynchronous operations 170
RPL operand
CHECK 389
CLSDST
AAREA 396
ACB 397
AREA 397
ARG 397
BRANCH 397, 416, 428
ECB 397, 416, 428
NIB 398
OPTCD 398, 399, 400
PARMS 400
RECLEN 400
RPL-based macroinstruction
errors and special conditions 279, 280
OPNDST 23
OPNSEC 23
return codes for 279
using 23
RPLC value of PROC operand 438

RPLLEN value of LENGTH operand 410
RQR (Request Recovery) request
need for SCIP exit routine to process 256
summary of 532
using 167
RQR value of CONTROL operand 535
RRN value of RESPOND operand 523
RSHUTD request
sending 514
RTNCD field 279, 509
RTR (Ready to Receive) request
using 514, 526
RTYPE operand
RECEIVE
example 472
explicit 178
indicating type of response received or expected in return 473
table 467
RESETSR 481
RPL 504, 507
RU (request/response unit)
classified by VTAM 179
communication network services, format 351, 352
definition 151
format
deliver 348
forward (MVS) 344
initiate load request format 350
load status request format 350
network services
embedded 350
not embedded 351
translate-inquiry request (TR-INQ), format 352

S

sample programs
sample program 1
data interface with logical units 580
how SAMP1 code relates to sample program 1 579
logic of 573
notes on LERAD and SYNAD exit routines 584
notes on LOGON exit routine 583
notes on LOSTERM exit routine 585
notes on mainline program 582
notes on RESP exit routine 584
what it does 579
sample program 2
logic of 605
sample program 3
assembler language code 628
logic of 627
using authorized path 627
save area, requirement for 234
SBI (stop bracket initiative) 526
SBI value on CONTROL operand 527
SCHED value on POST operand
RPL 503
SEND 521
scheduled output 172, 521
scheduling priority of I/O requests 503
SCIP exit routine
address 407
basic function of 256
entered as a result of
BIND request 258
SCIP exit routine *(continued)*
entered as a result of *(continued)*
Clear request 257
RQR request 257
SDT request 257
STSN request 258
UNBIND request 260
executing
in SRB mode 307
in TCB mode 307
read-only RPL provided to 256
registers upon entry 262
resynchronization of sequence numbers in 258
specifying in an ACB or NIB 229
using 102, 256
SDT (Start Data Traffic) request
basic function of 7
in request flow 532
indication 117
need for SCIP exit routine to process 256
receiving 407
sending 532, 535
using 164
secondary logical unit (SLU)
definition of 4
SEND
basic function of 24
examples of
for asynchronous operations 169
for synchronous operations 169, 576
major options 514
OPTCD=LMPEO
handling of negative response 190
POST operand 12, 522
POST=RESP 170
POST=SCHED 169, 170
RESPOND operand 156, 216
scheduling 169
specific mode for 174
specifying ECB posting in 170
specifying execution of RPL exit routine in 170
STYPE=REQ 169
STYPE=RESP 12, 154
using 157, 514
SEND operation (example) 200
SEND options 514
SENDCMD
basic function of 26, 529
using 529
sending network operator commands 529
sending requests and responses 514
sense code 500
sense fields and return codes for RPL-based macroinstructions 651
sense information
for a 3270 device 332
received at the application program 332
SEQNO field
for RECEIVE 473
for RPL 504, 509
for SEND 164, 525
how used with LMPEO 189, 509
sequence numbers
for RECEIVE 473
for RPL 504
for SEND 525
for STSN commands 537

sequence numbers *(continued)*
handling during LMPEO operation 189
in requests and responses 163
of normal-flow RUs 331
resetting to zero with Clear request 164
resynchronization of
general description 163
serialization of execution 309
session
accepting 89
acquiring 90
active 81, 84
address space 316
available 82
communication
macroinstructions 24
connected 81
control commands 532
cryptographic 131
cryptographic control 815
determining parameters for
INQUIRE 124
OPNDST OPTCD=ACCEPT 125
OPNDST OPTCD=ACQUIRE 125
REQSESS 125
SIMLOGON or CLSDST OPTCD=PASS 125
disabled 81
enabled 81
establishment
control block 115
macroinstructions 23
role of exit routines 100
stages of 5, 7, 83
with logical units 11, 27, 81
identifying 168
initiation
error recovery procedure 44
exit routines involved in 107
initiate request 6
initiate request types 85
use of SIMLOGON 87
using generic resource name 78
with logon information 6
limit 82
LU-LU 4
LU-LU session protocols 342
major communication alternatives 169
outage
codes 111
exit routines involved in 108
summary 105
parallel 4
parameters associated with CINIT 127
SSCP-LU 4
SSCP-LU session 6
SSCP-PU 4
SSCP-PU session 5
SSCP-SSCP 4
termination
by one of the session participants 109
control block 115
macroinstructions 23
stages of 84
terminate request types 86
with logical units 13, 28
types 3
session awareness data buffer 356
session control
transmission services profile 796
session establishment macroinstruction
CLSDST 23
OPNDST 23
OPNSEC 23
REQSESS 23
SESSIONC 24
SIMLOGON 23
TERMSESS 24
session instance identifier 139
session level error isolation 322
session monitor 338
session outage notification (SON) 105, 110
session parameter
3270, LU type 0 333
agreement 122
building and using in a BIND area 130
defining and naming (logon mode) 123
defining sets 123
example of
associated with a CINIT 127
in a BIND area 130
processing of by an application program 127, 128
specifying 34, 793
using 123
session parameter fields
format (BIND image) 793
function management 795
profile 795, 819
session qualifier pair 139
session state control vector 747
session termination
and Chase request 217
by a secondary application program 560
by one of the session participants 109
stages of 84
terminate request types 86
SESSIONC
and XRF 532
for communication 24
for session establishment 24
in sending SDT requests 157
options 541
using
general description 532
to reject a BIND request 98
with CONTROL=BIND 98
network-qualified names 98
SESSIONC command 532
SESSKEY value on OPTCD operand
RPL 498
SESSPARM value on OPTCD operand
INQUIRE
description 423
possible restrictions 412
source of session parameters 414
RPL 498
Set and Test Sequence Numbers request (STSN)
need for SCIP exit routine to process 258
possible responses to 532
receiving 407
sending 542
using 167, 532
SETLOGON
ACB MACRF operand, interaction with 83
basic function of 25, 543

SETLOGON *(continued)*
examples of use 575
HOLD 84
LOGON exit routine scheduling 543
START 84
using 543
shortcut keys 899
SHOWCB
advantage of 267
basic function of 22, 551
errors and special conditions for 278
use and examples of 270
using 551
SHUTD (Shutdown Complete request)
in data-flow-control request 518
on CONTROL operand of SEND macroinstruction 527
on STYPE operand of SEND macroinstruction 527
Shutdown Complete request (SHUTD)
in data-flow-control request 518
on CONTROL operand of SEND macroinstruction 527
on STYPE operand of SEND macroinstruction 527
SIGDATA operand
RPL 504, 509
SEND 525
signal request 213
SIGNAL value on STYPE operand 526
SIMLOGON
basic function of 23, 555
defined 87
OPTCD=CONALL 90
OPTCD=CONANY 91
OPTCD=PASS
determining session parameters for 125
using 87, 555
simulated logon requests 555
single task with multiple ACBs 299
single-thread application program
characteristics of 36
definition of 36
example of
sample program 1 573
single-thread operation 36
SLU (secondary logical unit)
definition of 4
SNA (Systems Network Architecture)
key concepts for VTAM 1
LU (logical unit) 2
NAU (network addressable unit) 1
protocols
for ensuring orderly communication 209
specifying 217
using 201
PU (physical unit) 1
sense fields 679
SSCP (system services control point) 1
task association
exit routine 310
macroinstruction 311
SNA network interconnect vectors
host-subarea-network-name vector 63
host-subarea-PU-network-address vector 63
host-subarea-PU-network-name vector 63
maximum-subarea vector 64
network-name vector 63
SSCP-name vector 63
SNA protocol specifications 897
softcopy information xxix
SON (session outage notification) 105, 110
SON type codes 260
SONCODE 564
sources of SNA Initiate and Terminate requests 85
SPEC value on OPTCD operand 502
specific-mode
in a SEND or RECEIVE operation 174, 176
used to handle an inquiry 175
SRBEXIT operand 307
of ACB 381
SSCP (system services control point)
in SNA network 2
LU-LU session 4
role of, in VTAM 1
SSCP-LU session 4, 6
SSCP-PU session 4, 5
SSCP-SSCP session 4
SSENSEI field 510
SSENSEO field
for CLSDST request 401
for Logical Unit Status (LUSTAT) request 504
to represent a major class of error 525, 538
with requests and negative responses 510
SSENSMI field 510
SSENSMO field
for CLSDST request 401
for Logical Unit Status (LUSTAT) request 504
with requests and negative responses 510
with SNA-defined errors, how coded 525, 538, 564
stages of session establishment 83
stages of session termination
definition of 84
STANDARD value on CODESEL operand 519
Start Data Traffic request (SDT)
basic function of 7
in request flow 532
indication 117
need for SCIP exit routine to process 256
receiving 407
sending 532, 535
using 164
START value on OPTCD operand
RPL 501
SETLOGON 548
stop bracket initiative (SBI) 526
STOP value on OPTCD operand
RPL 501
SETLOGON 548
stopping logon request queuing 543, 550
storage key
ACB storage allocation 376
storage limitation
ACB data space 375
STSN (Set and Test Sequence Numbers request)
need for SCIP exit routine to process 258
possible responses to 532
receiving 407
sending 542
using 167, 532
STSN operand value 532
STYPE operand
RPL 505, 525
subtasks
using separate ACBs 298
using the same ACB 298
supervisor state, for use of authorized path 302

symbolic name
of a logical unit 115, 437
of an application program 375
SYN (synchronous handling) 502
SYN operand value 502
SYNAD exit routine
addressing mode 238, 263
advantage of 263
basic function of 13
coding 263, 290
coding, special requirements 230
executing
in SRB mode 308
in TCB mode 308
given control 321, 407
how to use 263
linkage conventions for 234, 263
not reentrant 234
parameters passed to 263
purpose of 263
reentrant 234
register usage 291
registers upon entry 263
synchronous handling (SYN) 502
synchronous operation
advantages of 45
characteristics of 40
errors for 282
returning to application under same SRB 499
versus asynchronous 169
synchronous request 169
syntax diagram, how to read xxv
SYSTEM operand value 441
system services control point (SSCP)
in SNA network 2
LU-LU session 4
role of, in VTAM 1
SSCP-LU session 4, 6
SSCP-PU session 4, 5
SSCP-SSCP session 4
system-sense information
sending 525, 538, 564
system-sense modifier information
sending 525, 538, 564
Systems Network Architecture (SNA)
key concepts for VTAM 1
LU (logical unit) 2
NAU (network addressable unit) 1
protocols
for ensuring orderly communication 209
specifying 217
using 201
PU (physical unit) 1
sense fields 679
SSCP (system services control point) 1
task association
exit routine 310
macroinstruction 311

T

target resource name 344
task association
description 310
of exit routines 310
of macroinstructions 311
task level error isolation 323
task termination 323
TCBEXIT operand 307
TCP/IP
online information xxxi
Technotes xxix
terminals
characteristics of LU type 0 3270 328
differences among LU type 0 3270 335
flow
deliver 339
forward 339
Terminate Cleanup request 84
Terminate Forced request 84
Terminate Orderly request 84
terminating affinities 79
terminating sessions with logical units 390
generic resources 79, 543
termination
address space 323
task 323
TERMSESS
basic function of 24
using 99, 560
network-qualified names 99
test request RUs, 3270 Information Display System
actions taken by the network 335
TESTCB
advantage of 267
basic function of 22
errors and special conditions for 278
testing OFLAGS field 277
use and examples of 271
using 565
testing
control block fields 565
multiple field values 565
processing options or option codes 565
third party Initiate and Terminate requests 85, 86
THRDPTY operand 400, 503
timeout CNM request unit format 352
TOPLOGON operand 414, 426
TPEND exit routine
closedown of VTAM 407
closing an application program 74
entry to, after HALT commands 263
executing
in SRB mode 307
in TCB mode 307
parameters available on entry to 265
reason codes 264
registers upon entry 265
user exit queues 309
with reason code 8 32
TPEND operand 407
trademark information 911
transmission control 331
transmission services
profile 795
usage field 797
type code
in PARMS operand of TERMSESS macroinstruction 564
UNBIND in RPL 500
UNBIND used on UNBIND RU 503

U

UNBIND request
  basic function of 7
  need for SCIP exit routine to process 256
  receiving 103
  SON codes 93, 260
  TERMSESS restrictions 100
USENSEI field 510
USENSEO field
  errors indicated by 527, 564
  for CLSSDT request 401
  for Logical Unit Status (LUSTAT) request 505
  how coded 527, 564
  when RPL-based macroinstruction is completed 510
user data 816
user data length 816
user exit queues 309
user RH option (USERRH)
  description 196
  example of using 200
  handling the Sense Data Included (SDI) indicator 199
  operating considerations 196
  operation for inbound RUs 198
  operation for outbound RUs 197
  relationship to NIB 199
user sense information
  receiving 402
  sending 527, 564
USERFLD field of the NIB 37
USERFLD operand
  of ACB 381
  of NIB 441
USERRH (user RH option)
  description 196
  example of using 200
  handling the Sense Data Included (SDI) indicator 199
  operating considerations 196
  operation for inbound RUs 198
  operation for outbound RUs 197
  relationship to NIB 199
USERRH field in the RPL
  relationship to the request/response header 198
using logon mode names and session parameters 123
using network-qualified names support 65
USS Messages
  national language code values 117

V

VARY command 529
vector list
  access method support vector 62
  component-ID vector 62
  function-list vector 62
  release-level vector 62
  resource-ID vector 63
VTAM
  domain 16
  exit routines 19
  FRR (functional recovery routine) 324
  general programming considerations 33
  interfacing with an application program 50
  keyword operands 19
  language 19
  macroinstruction differences
    across operating systems 295
VTAM *(continued)*
  macroinstructions
    conventions and descriptions 371
    summary of 20
  manipulative macroinstructions 19
  scheduling output 38
  SNA concepts 1
  special programming considerations 295
  VTAM-initiated HALT 76
VTAM, online information xxxi

W

WAREA operand 411

X

XRF (extended reference facility)
  and SESSIONC 532
  programming 149
  session requests 91
  terminating sessions 74
XRF session activation control vector (MVS only) 845

Y

YES value
  BRANCH operand 494, 518, 535
  LISTEND operand 436

Z

z/OS Basic Skills Information Center xxix
z/OS, documentation library listing 913

Communicating your comments to IBM

If you especially like or dislike anything about this document, use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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.

Send your comments to us in any of the following ways:

- To send comments by FAX, use this number: 1+919-254-1258
- To send comments electronically, use this address:
  - comsvrcf@us.ibm.com
- To send comments by post, use this address:

```
International Business Machines Corporation
Attn: z/OS Communications Server Information Development
P.O. Box 12195, 3039 Cornwallis Road
Department AKCA, Building 501
Research Triangle Park, North Carolina 27709-2195
```

Make sure to include the following information in your note:

- Title and publication number of this document
- Page number or topic to which your comment applies

© Copyright IBM Corp. 2000, 2013

Product Number: 5650-ZOS

Printed in USA

SC27-3674-00