z/VM
7.3

Connectivity

IBM

SC24-6267-73

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 339.

This edition applies to version 7, release 3 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-07-17

Contents

Figures... xiii

Tables...xix

About This Document..xxi

Intended Audience.. xxii
Syntax, Message, and Response Conventions...xxii
Where to Find More Information..xxv
Links to Other Documents and Websites.. xxv

How to provide feedback to IBM.. xxvii

Summary of Changes for z/VM: Connectivity.......................................xxix

SC24-6267-73, z/VM 7.3 (July 2025).. xxix
SC24-6267-73, z/VM 7.3 (December 2023)...xxx
SC24-6267-73, z/VM 7.3 (September 2022)..xxx

Part 1. Introduction to Connectivity.. 1

Chapter 1. Introduction to z/VM Connectivity..3
What is Connectivity?.. 3
TCP/IP Communications.. 4
OSA Adapters: OSA-Express and Network Express.. 5
HiperSockets... 6
Guest LAN Intercommunication among Virtual Machines.....................................6
APPC Communications between Programs..6
APPC Communications within a z/VM System..7
APPC Communications within a Collection of z/VM Systems................................ 8
APPC Communications between z/VM Systems.. 9
IUCV Communications between z/VM Systems..10
APPC Communications with an SNA Network...11
SNA Network Communications between TSAF or CS Collections.........................12
SNA Network Communications to Other Systems...12
Summary of APPC Communications...13

Chapter 2. Introduction to Connectivity Terminology.. 15
Virtual Network Terminology.. 15
What is QDIO and EQDIO?...16
What Is OSA-Express?...16
What Is Network Express?... 17
What are HiperSockets?... 17
What Is a Network Interface Card (NIC)?... 17
What Is a Guest LAN?... 17
What Is a Virtual Switch?...17
What Is a Virtual Switch Controller?.. 18
What Is a Global VSwitch?...18
What Is Transport Type?..18
What Is a Virtual LAN (VLAN)?.. 18
What Are Tagged and Untagged Frames?... 18

What Is VLAN Unaware?....18
What Is Port Type?....18
What Is an Access Port?....18
What Is a Trunk Port?....19
What Is a Native VLAN ID....19
What Is a Port VLAN ID (pvid)?....19
What Are Ingress Rules?....19
What Are Egress Rules?....19
What Is a Global VLAN ID?....19
What Is a Link Aggregation Port Group (LAG)?....19
What Is an Exclusive Port Group?....20
What Is a Shared Port Group?....20
What Is a Multi-VSwitch LAG Configuration?....20
What Is a LACP?....20
What Is a LAG Port Controller?....20
What is an UPLINK Port?....20
What Is HiperSockets Bridge Port?....20
What Is QEBSM?....21
What Is a HiperSockets Guest Port?....21
What Is a HiperSockets Bridge Capable Port?....21
What Is PMTUD?....21
What Is the Difference Between a Port Based and User Based Virtual Switch?....21
What is Directory Network Authorization (DNA)?....22
What Is a IVL Network?....23
What Is a IVL Domain?....23
What Is a IVL Virtual Switch?....23
What Is Virtual Switch Priority Queuing?....23
Systems Network Architecture Terminology....23
What Is an SNA Network?....23
What Is a Logical Unit?....24
What Is a Session?....24
What Is Session Security?....24
What Is a Transaction Program?....24
What Is a Conversation?....24
What Is a Mode Name?....25
What Is a Session Limit?....26
What Is Contention?....26
What Is Conversation Security?....26
What Is Negotiation?....27
z/VM Terminology....27
What Is *IDENT?....28
What Is a TSAF Collection?....28
What Is a CS Collection?....28
What Is a Domain?....29
What Is a VM Resource?....29
What Are Communications Partners?....31
What Is an AVS Gateway?....32
What Is a System Gateway?....33

Part 2. Planning Virtual Networks....37

Chapter 3. Networking Options in z/VM....39
Real Hardware Options....39
z/VM Network with OSA-Express Adapter....39
z/VM Network with Network Express Adapter....40
z/VM Network with HiperSockets....40
Advantages of Using Real Network Hardware....40

Virtual Networking Options....41
Guest LAN....41
Virtual Switch....41
Advantages of Using Virtual Networks....42
Guest LAN versus Virtual Switch....43
QDIO versus iQDIO Guest LAN....44

Chapter 4. Planning for Guest LANs and Virtual Switches....45
Guest LAN Environment....45
Virtual Switch Environment....46
UPLINK NIC....48
BRIDGEPORT....48
IPv6 Considerations....48
Link Aggregation....49
Considerations for Virtual Switch or Guest LAN Configuration....49
Transport Mode — IP or Ethernet....50
Media Access Control (MAC) address....50
Additional Considerations for Virtual Switch Configurations....54
Virtual Switch Controller Configuration Strategy....54
Virtual Switch Priority Queuing Function....55
OSA-Express Multiple Ports Function....56
Virtual Switch Management (SNMP)....57
Guest Port Traffic Forwarding Strategies for the Virtual Switch....57
Virtual Switch Configuration Strategy with VLAN....62
Virtual Switch Guest Configuration Considerations in IP Mode....66
Servicing Virtual Switch OSA adapters....67
Virtual Switch Failover....67
Virtual Switch Topology Considerations....70
Spanning Tree Protocol and the Virtual Switch....72
Configuring a Virtual Switch....74
Creating a Virtual Switch....74
Configuring Access to a Virtual Switch....75
Configuring an SNMP Subagent for a Virtual Switch....75
Creating a Simulated NIC....75
Coupling the Simulated NIC to the Virtual Switch....76
Configuring the Network Interface....76
Example — Dynamically Creating and Destroying a Virtual Switch....76
Virtual Switch Attributes....77
Automating Virtual Switch Configuration on IPL....77
Configuring a Guest LAN....78
Creating a Guest LAN segment....78
Creating a Simulated NIC on Each Virtual Machine....79
Coupling the Simulated NIC to the Guest LAN....79
Configuring the Network Interface for the Guest LAN....79
Example — Creating and Destroying a QDIO Guest LAN....80
Guest LAN Attributes....80
Automating Guest LAN Configuration on IPL....81

Chapter 5. Planning a z/VM Environment with OSA-Express and HiperSockets....83
OSA-Express Network....83
OSA-Express Multiple Ports Function....83
Interface Takeover for Local Area Networks....84
Setting up an OSA-Express Network....84
Defining Host Definitions for Your OSA-Express Adapter....84
Configuring OSA-Express Modes....84
Configuring Transport Mode....84
Sample z/VM TCP/IP Profile....85
HiperSockets Network....85

Setting up a HiperSockets Network....85
Defining IOCP for HiperSockets and Attaching the Devices....86
Using HiperSockets with TCP/IP.... 86
Verifying z/VM TCP/IP Connectivity....87
Verifying the Channel Path.... 87
Verifying Device Allocation.... 87
Verifying TCP/IP Connectivity....87

Chapter 6. Live Guest Relocation Networking Considerations....89
Real Networking Devices.... 89
Simulated Networking Connections (NICs)....91
Virtual Switch Equivalency....91
Relocation Eligibility Checks.... 91

Chapter 7. Bridging a HiperSockets LAN with a z/VM Virtual Switch....95
z/VM Guest Networking Support Paradigm Shift with the Virtual Switch HiperSockets Bridge....96
z/VM Virtual Switch Bridge Port (BRIDGEPORT)....96
High Availability for Bridged HiperSockets Channel (LAN)....97
Primary/Secondary Roles for Bridge Ports....97
Primary BRIDGEPORT Role.... 99
Secondary BRIDGEPORT Role....99
NIC Distribution for Bridge Ports....100
Construction of a Bridged HiperSockets LAN....100
IQD CHPID Definition.... 101
OSD and OSH CHPID Definitions....102
z/VM Virtual Switch Bridge Port Definition.... 103
Networking Connectivity in a Virtual Switch HiperSockets Bridge Configuration....109
z/OS Guest Considerations in a Bridged HiperSockets Configuration....110
Path MTU Discovery (PMTUD) Requirement for Bridge Capable Guests....111
Cross CEC Bridged HiperSockets Channel Network.... 111

Chapter 8. Virtual Switch Link Aggregation.... 113
Link Aggregation Overview.... 113
Link Aggregation Control Protocol (LACP)....113
Link Aggregation Port Group....114
Exclusive Port Group.... 115
Shared Port Group.... 115
Exclusive Link Aggregation vs Multi-VSwitch Link Aggregation....116
Exclusive Link Aggregation (default)....117
Multi-VSwitch LAG Configuration....118
Exclusive LAG Configuration....119
Configure the Partner Switch.... 120
Create an Exclusive Port Group....120
Configure the Exclusive Port Group to a VSwitch.... 121
Verify Exclusive LAG Configuration.... 121
Multi-VSwitch LAG Configuration.... 123
Link Aggregation Control Protocol Enhancements....124
Automatic OSA Device Allocation.... 124
Building a Multi-VSwitch LAG Configuration....125
Example of Sharing a LAG within the same z/VM System....129
Take-Over and Take-Back Recovery for a Multi-VSwitch LAG Configuration.... 131
Load Balancing within the Virtual Switch....131
Management Scope of a Load Balance Operation....132

Chapter 9. Global Virtual Switch.... 135
IVL (Inter-VSwitch Link) Overview....136
IVL Domain.... 136
IVL Virtual Switch.... 137

Additional IVL Controls and Diagnostic Aids.... 138
Deploying an IVL Network.... 139
Connectivity to the IVL Network.... 139
IVL Network Configuration Domain B VLAN 8.... 140
Verification of IVL Network Connectivity.... 140
Define a Global VSwitch.... 142
Take-Over and Take-Back Operations.... 143
Take-Over Sequence.... 144

Chapter 10. Troubleshooting a Virtual Switch or Guest LAN.... 147
Methods of Guest LAN Sniffing.... 147
Promiscuous Mode with a Guest Virtual Machine.... 147
TRSOURCE TYPE LAN Option.... 148
Keeping the Virtual Network Secure.... 148
Using Promiscuous Mode with Guest Virtual Machine.... 149
Authorizing a Guest for Promiscuous Mode.... 149
Activating Promiscuous Mode.... 151
Using TRSOURCE to TRACE a Guest LAN or Virtual Switch.... 154
CP TRSOURCE TYPE LAN Command.... 155
Interpreting the TRSOURCE Output.... 158
A Troubleshooting Example Using TRSOURCE.... 159

Part 3. APPC Server and Requester Virtual Machines.... 163

Chapter 11. Setting Up Server and Requester Virtual Machines.... 165
Communications Partners.... 165
Resource Managers.... 166
Summary of Resource Features.... 168
User Programs.... 169
Identifying Communications Partners.... 169
Preparing Virtual Machines to Manage Resources.... 173
Defining Unique Resource Names.... 173
Defining Unique User IDs.... 173
Managing Local, Global, and System Resources.... 174
Managing Private Resources.... 174
Preparing Virtual Machines to Request Connections to Resources.... 177
Authorizing Virtual Machines to Connect to Resources and Gateways.... 178
Setting Up a CMS Communications Directory.... 178
Using APPCPASS Directory Statements.... 181
Setting Up the SYSPROF EXEC.... 183

Part 4. TSAF Virtual Machine.... 185

Chapter 12. Preparing to Use the TSAF Virtual Machine.... 187
Setting Up the TSAF Virtual Machine.... 187
Loading TSAF Program Materials.... 188
Rebuilding the RUNTSAF MODULE.... 188
Creating the CP Directory Entry.... 188
IUCV Statements.... 188
OPTION Directory Control Statement.... 189
Other Directory Statements.... 190
Sample CP Directory Entry.... 190
Accessing TSAF Service Updates.... 191
Creating the TSAF PROFILE EXEC.... 191
Enabling the TSAF Message Repository.... 192
Setting Up Links for Communication.... 192
Defining Real Communication Links.... 192

Dedicating Links to TSAF....193
Updating the ATSLINKS FILE....194
Setting Up Centralized System Operations....194

Chapter 13. Setting Up TSAF Collections....195
TSAF Collection Structure....195
TSAF Collection Examples....195
Setting Up a TSAF Collection....195
Connecting to a Global Resource....196
Connecting to a Shared File System File Pool....198
Connecting to a Private Resource....200
Establishing an APPC Link....202
Supported Links....207
Setting Up Reliable Routes....208
TSAF Routing....209
How TSAF Dynamically Configures a TSAF Collection....209
Route Failure....210
When Two TSAF Collections Merge to Form One....210
Maintaining a Single TSAF Collection....210
Node-By-Node Scenario....211
Link-by-Link Scenario....211
Additional Considerations....212
Performance Considerations....212
Communication Path Characteristics....212
Line Performance Characteristics....212
APPC Link and VTAM Performance Considerations....213

Chapter 14. Operating the TSAF Virtual Machine....215
Overview of TSAF Commands....215
Using Online HELP for TSAF Commands....215
ADD LINK....217
DELETE LINK....220
QUERY....222
RUNTSAF....227
SET ETRACE....230
STOP TSAF....233

Chapter 15. Generating TSAF Accounting and Link Statistics....235
Initialization Accounting Record....235
Session Accounting Record....236
Link Statistics Record....237
Termination Accounting Record....237

Chapter 16. Collecting TSAF Problem Diagnosis Information....239
Collecting Abend Information....239
Using the Console Log....240
Using TSAF Dumps to Diagnose Problems....240
Using System Trace Data to Diagnose Problems....241
Using Interactive Service Queries....241

Part 5. AVS Virtual Machine....243

Chapter 17. Preparing to Use the AVS Virtual Machine....245
Defining the AVS Virtual Machine to GCS....245
Setting Up the AVS Virtual Machine....245
Updating the AVS Virtual Machine's CP Directory Entry....246
IUCV Statements....246

OPTION Directory Control Statement....246
Other Directory Statements.... 247
Sample CP Directory Entry.... 247
Installing AVS.... 248
Preparing the AVS Message Repository....248
Defining AVS to VTAM....249
Logon Mode Table....249
Application Program Major Node.... 249
Performance Considerations....251
Security Level Considerations.... 251
Preparing the AVS Accounting Module.... 251
Setting Up the AVS Virtual Machine Profile.... 252
Creating the AGWPROF GCS Exec.... 253
Setting Up the AVS Virtual Machine for Autologging....253
Setting Up Centralized System Operations.... 253

Chapter 18. Using AVS....255
Understanding the SNA Network View....255
Global Gateways....256
Private Gateways.... 256
AVS Examples....257
Inbound Connection to Global Resources.... 257
Inbound Connection to Private Resources through Dedicated Private Gateways....259
Inbound Connection to Private Resources through Nondedicated Private Gateways.... 261
Outbound Connection to a Global Resource in Another TSAF Collection....263
Outbound Connection to a Shared File System File Pool....265

Chapter 19. Operating the AVS Virtual Machine....269
Overview of AVS Commands....269
Using Online HELP for AVS Commands.... 269
AGW ACTIVATE GATEWAY.... 271
AGW ADD USERID....273
AGW CNOS....274
AGW DEACTIVE CONV.... 277
AGW DEACTIVE GATEWAY....278
AGW DELETE USERID.... 279
AGW QUERY.... 280
AGW QUIESCE....285
AGW SET ETRACE....286
AGW SET ITRACE.... 287
AGW START.... 288
AGW STOP.... 289

Chapter 20. Generating AVS Accounting Information....291
Initialization Accounting Record....291
Conversation Accounting Records....291
Conversation Start Accounting Record.... 292
Conversation Active and Conversation End Accounting Records.... 292
Termination Accounting Record.... 293

Chapter 21. Collecting AVS Problem Diagnosis Information.... 295
Collecting Abend Information....295
Using AVS Dumps to Diagnose Problems....295
Using System Trace Data to Diagnose Problems.... 296
Using Interactive Queries.... 296
Using AVS Error Messages.... 297
Using VTAM Trace Facilities.... 297

Part 6. Planning for ISFC..299

Chapter 22. Planning a CS Collection..301
CS Collection Structure..301
Planning the CS Collection Configuration..301
Defining Unique Identifiers..301
Defining the VM Domain Controller Name..301
Defining Links in a CS Collection..302
Reducing CTC Links with Broadcast Routing..302
Operating with TSAF and SNA Network Communications..303
Managing Resources..303
Accessing Resources..303
Security Considerations..303
System Definitions..304
Migrating from a TSAF Collection to a CS Collection..304
Migration Considerations..304
Migrating from TSAF to ISFC..305

Chapter 23. Operating the VM Domain Controller..309
Overview of Commands..309
Generating Accounting Information..309
Forwarding ISFC Information to a Central Site..310

Appendix A. IUCV Directory Control Statement.. 311
Specifying IUCV Statements..311
Authorizing Connections to Resources..311
IUCV Statement Syntax (Connection Authority)..311
Authorizing Resource Management..312
IUCV Statement Syntax (Resource Management)..312
Identifying Multiple Resources..313
Authorizing the AVS Virtual Machine to Manage Gateways..314
IUCV Statement Syntax (Gateway Management)..314
Identifying Multiple Gateways..315
Examples of Authorizing Virtual Machines..315

Appendix B. Connectivity Summary..319
Connections within a TSAF or CS Collection..319
Access Security Verification..321
Outbound Connections to the SNA Network..321
Inbound Connections from the SNA Network..322
Access Security Verification..323

Appendix C. Introduction to Service Pool Support..325
What Is a Service Pool?..325
Why Have a Service Pool?..327
Considerations for Writing and Installing a CMR..327
Interactions between AVS and a CMR..327
ACTIVATE Event..328
DEACTIVATE Event..328
INBOUND ALLOCATE Event..328
OUTBOUND ALLOCATE Event..329
DEALLOCATE Event..330
ATTENTION LOSS Event..330
ABEND Event..331
Queued Allocates..331

Appendix D. POSIX Security Values and Connectivity.. 333
Introduction..333
File Access..333
File Execution...333

Appendix E. Secure APPC Communications..335
What Is Authentication?...335
What Is Authorization?...335
The Importance of Unique User IDs..336
Using SECURITY(SAME) Safely...336

Appendix F. Creating a VSwitch Controller...337

Notices...339
Programming Interface Information..340
Trademarks..340
Terms and Conditions for Product Documentation...340
IBM Online Privacy Statement..341

Bibliography...343
Where to Get z/VM Information..343
z/VM Base Library...343
z/VM Facilities and Features..344
Prerequisite Products..346
Related Products..346

Index... 347

Figures

1. z/VM Guest LAN....6

2. z/VM Connectivity Support for APPC/VM and CPI Communications....7

3. APPC Communications within One z/VM System....8

4. Communication in a TSAF Collection of z/VM Systems....9

5. APPC/VM Communication in a CS Collection....10

6. IUCV Communication in a CS Collection....11

7. Communication between Two Collections in the SNA Network....12

8. Communication between the TSAF Collection and another System in the SNA Network....13

9. Summary of APPC Communications....13

10. A Session and a Conversation....25

11. Using a System Gateway to Access System Resources....34

12. Using a System Gateway to Access a Private Resource Manager....34

13. Using a System Gateway to Access Resources in an Adjacent Collection....35

14. Guest LAN Environment....46

15. Virtual Switch Environment....48

16. Virtual Switch in a Link Aggregation Environment....49

17. SNMP Connectivity for the Network Management System....57

18. Typical IBM Z CPC Configuration....58

19. Virtual Switch Port Isolation....59

20. z/VM VSWITCH-initiated VEPA Mode....61

21. Failover of a Virtual Switch that Uses OSA-Express Devices....68

22. Failover of a Virtual Switch that Uses Network Express Devices....69

23. Multiple VLAN Environment GIF....71

24. Virtual Switch with Remote Guest LAN.. 72

25. Redundant network example..73

26. Sample z/VM TCP/IP Profile... 85

27. Bridged HiperSockets channel... 96

28. HiperSockets Channel with Primary and Secondary VSwitch Bridge Ports.. 98

29. Query of VSwitch A showing Bridge Port data...99

30. Query of VSwitch B showing Bridge Port data... 99

31. IQD HiperSockets EXTERNAL_BRIDGED Channel with bridge capable ports....................................101

32. OSD CHPID to provide external network connectivity for virtual switch.. 102

33. OSH CHPID to provide external network connectivity for virtual switch.. 103

34. Instantiated virtual switch with no external connectivity.. 104

35. Definition of Activation of virtual switch Uplink port... 105

36. Virtual Switch Bridged HiperSockets LAN...106

37. Query VSwitch VSW output part 1...108

38. Query VSwitch VSW output part 2...109

39. Virtual switch HiperSockets Bridge connectivity examples.. 110

40. Path MTU Virtual Switch setting and value... 111

41. HiperSockets LAN spanning multiple CECs.. 112

42. Exclusive Link Aggregation Configuration...117

43. Multi-VSwitches LAG Configuration.. 118

44. Exclusive LAG Configuration..120

45. Exclusive Port Group...121

46. Query of Exclusive Port Group ANG.. 121

47. Query of Active Exclusive Port Group ANG.. 122

48. Display of Query Port Group ANG Details.. 123

49. Global VSwitches that Share a LAG Port Group..125

50. Shared Port Group ANG..126

51. Query Port Group Display for ANG.. 127

52. VSwitch Connections to a Shared Port Group..128

53. Display of Query Shared Port Group ANG Details..129

54. Sharing a Port Group within the same z/VM System.. 130

55. IVL VSwitch SUE on System CASEY..139

56. IVL Domain B Network..140

57. Query VSwitch SUE display.. 141

58. Query VMLAN with IVL Domain B..141

59. Global VSwitch PEGGY with shared port group ANG..142

60. Example of a Multi-VSwitch LAG Configuration.. 143

61. LAG Port Controller Take-Over Sequence..144

62. VLAN Authority Honored.. 148

63. Guest Promiscuous Mode..149

64. Example of Granting and Revoking Promiscuous Mode Authority on a Guest LAN.............................150

65. Query VSWITCH PRO output.. 150

66. Query LAN PRO output.. 151

67. Query NIC DETails..152

68. Activating Promiscuous Mode.. 152

69. Deactivating Promiscuous Mode.. 152

70. The Options Line from a Query NIC DETails..153

71. Using tcpdump with Promiscuous Mode on the Linux Guest.. 154

72. Possible Network Problem Revealed by Promiscuous Mode and tcpdump....................................... 154

73. Sniffer Mode using TRSOURCE..155

74. Example of z/VM Virtual Network....157

75. Example of TRSOURCE with TYPE LAN option.... 157

76. Enabling a TRSOURCE trace.... 157

77. Disabling a TRSOURCE trace.... 158

78. Processing a TRSOURCE trace using TRACERED....158

79. Raw Hex Trace File Processed by TRACERED....158

80. Layout of Raw Hex Data....159

81. Collecting Data Using TRSOURCE.... 159

82. TRSOURCE Data from DEBUG FILE A (Part 1 of 2).... 160

83. TRSOURCE Data from DEBUG FILE A (Part 2 of 2).... 160

84. Using IPFORMAT to decode the TRSOURCE Data....161

85. IPFORMAT Decoding of First Trace Packet.... 161

86. IPFORMAT Decoding of Second Trace Packet.... 162

87. Communications between a User Program and a Resource Manager.... 165

88. Resource Name Space in a TSAF Collection.... 166

89. A TSAF Virtual Machine Running in a z/VM System....187

90. Sample CP Directory Entry for the TSAF Virtual Machine....191

91. Sample PROFILE EXEC for the TSAF Virtual Machine.... 192

92. TSAF Virtual Machine Identifying Itself.... 196

93. TSAF Virtual Machines Exchanging Information....196

94. Resource Manager Requesting to Manage a Global Resource.... 197

95. A Local Program Accessing a Global Resource....197

96. Remote Program Accessing a Global Resource....198

97. Sending and Receiving....198

98. File Pool Server Requesting to Manage a File Pool....199

99. Local User Accessing the File Pool....199

100. Remote User Accessing the File Pool....200

101. Processing CMS Commands....200

102. Resource Manager Preparing to Manage a Private Resource....201

103. Local Program Accessing a Private Resource....201

104. Remote Program Accessing a Private Resource....202

105. Sending and Receiving Data....202

106. System A and System B....203

107. Requesting a Connection to the TSAFb Virtual Machine....206

108. Requesting a Connection to the TSAFa Virtual Machine....206

109. System A and System B Form a TSAF Collection....207

110. Accessing the Resource Manager on System B....207

111. Sample TSAF Collection with Various Links....208

112. A Sample TSAF Collection....208

113. A TSAF Collection....209

114. More Reliable TSAF Collection....209

115. Remote Systems Connected by Two Physical VTAM Controlled Links....213

116. Logically Fully-Connected TSAF Collection....214

117. TSAF Console Log Example....240

118. An AVS Virtual Machine Running in a z/VM System....245

119. Sample AVS Virtual Machine CP Directory Entry....248

120. VTAM APPL Statement Example....249

121. PROFILE GCS Example....252

122. AGWPROF GCS Example....253

123. Gateway Overview....256

124. Defining a Global Gateway... 258

125. Remote APPC Transaction Program Issuing a Request...258

126. Remote APPC Transaction Program Communicating with a Global Resource................................. 259

127. Defining a Dedicated Private Gateway.. 260

128. Remote APPC Transaction Program Issuing a Request...260

129. Remote APPC Transaction Program Communicating with a Private Resource.................................261

130. AVS Operator Defining a Nondedicated Private Gateway...262

131. Remote APPC Transaction Program Issuing a Request...262

132. Remote APPC Transaction Program Communicating with a Private Resource.................................263

133. Defining Global Gateways... 264

134. APPC/VM Program Requesting to Connect to a Global Resource in a Remote TSAF Collection........264

135. APPC/VM Program Communicating with a Global Resource in a Remote TSAF Collection.............. 265

136. Defining Global Gateways... 266

137. User Requesting Access to a File Pool in a Remote TSAF Collection..267

138. User Accessing a File Pool in a Remote TSAF Collection.. 267

139. TSAF Collection with CTC Links...305

140. Forming a CS Collection and a TSAF Collection...306

141. Adjacent TSAF and CS Collections... 307

142. A CS Collection and Partitioned TSAF Collections...307

143. TSAF Collection with Authorized Global Resource Managers and User Programs............................316

144. TSAF Collection with Authorized Local and Private Resource Managers and User Programs........... 317

145. TSAF Collection with an AVS Virtual Machine..318

146. Service Pool Environment...325

147. AVS Interface with CMR (Service Pool Manager)...326

Tables

1. Examples of Syntax Diagram Conventions....xxii

2. User and Port Style Management Differences....22

3. Host Program Definition Summary for z/VM.... 84

4. Maximum frame size and TCP/IP maximum transmission unit (MTU) HiperSocketsmaximum transmission unit (MTU)....86

5. VSWITCH attributes that are compared for relocation eligibility.... 91

6. Promiscuous_Denied Keyword Reasons and Solutions....153

7. Summary of Resource Features....168

8. Identifying the Target of a Connection Request....170

9. Contents of the $SERVER$ NAMES File.... 176

10. Contents of a CMS Communications Directory File.... 180

11. IUCV Statements to Authorize Connections to Resources....312

12. Connection to Resources within the TSAF or CS Collection.... 320

13. Outbound Connection to Resources in the SNA Network.... 322

14. Inbound Connection to Resources in a TSAF or CS Collection....322

15. AVS Events that Cause Interactions between AVS and the CMR.... 326

About This Document

This document provides an introduction to the facilities of IBM z/VM that allow z/VM systems, application programs, and guests to establish logical connections to other systems, application programs, and guests. It describes the following z/VM connectivity facilities:

- Transmission Control Protocol/Internet Protocol (TCP/IP) for z/VM
- z/VM virtual networks (guest LANs and virtual switches)
- Advanced Program-to-Program Communications (APPC)
- Transparent Services Access Facility (TSAF)
- APPC/VM VTAM® Support (AVS)
- Inter-System Facility for Communications (ISFC)

If you are...	Then use...
Responsible for planning and implementing networks for z/VM	• Part 1, “Introduction to Connectivity,” on page 1 • Part 2, “Planning Virtual Networks,” on page 37
An application programmer who uses APPC to write programs or a system administrator in charge of setting up virtual machines in which APPC applications will run	• Part 1, “Introduction to Connectivity,” on page 1 • Part 3, “APPC Server and Requester Virtual Machines,” on page 163 • Appendix A, “IUCV Directory Control Statement,” on page 311 • Appendix D, “POSIX Security Values and Connectivity,” on page 333 • Appendix E, “Secure APPC Communications,” on page 335
An application programmer who uses APPC to write programs	The *z/VM: CMS Application Development Guide* for information on: • The Advanced Program-to-Program Communications/Virtual Machine (APPC/VM) programming interface • The Common Program Interface (CPI) Communications (also known as SAA® communication interface) programming interface
A system operator or system administrator in charge of running the TSAF virtual machine	• Part 1, “Introduction to Connectivity,” on page 1 • Part 3, “APPC Server and Requester Virtual Machines,” on page 163 • Part 4, “TSAF Virtual Machine,” on page 185

If you are...	Then use...
A system operator or system administrator in charge of running the AVS virtual machine	• Part 1, “Introduction to Connectivity,” on page 1 • Part 3, “APPC Server and Requester Virtual Machines,” on page 163 • Part 5, “AVS Virtual Machine,” on page 243
A system operator or system administrator in charge of planning and running ISFC	• Part 1, “Introduction to Connectivity,” on page 1 • Part 3, “APPC Server and Requester Virtual Machines,” on page 163 • Part 6, “Planning for ISFC,” on page 299

Intended Audience

This document is for anyone responsible for planning, setting up, running, and maintaining z/VM connectivity facilities.

You need to have a general knowledge of the z/VM system and a basic knowledge of application programming. Knowledge of TCP/IP, SNA, and VTAM concepts is also helpful.

Syntax, Message, and Response Conventions

The following topics provide information on the conventions used in syntax diagrams and in examples of messages and responses.

How to Read Syntax Diagrams

Special diagrams (often called *railroad tracks*) are used to show the syntax of external interfaces.

To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

- The ►►—— symbol indicates the beginning of the syntax diagram.
- The ——► symbol, at the end of a line, indicates that the syntax diagram is continued on the next line.
- The ►—— symbol, at the beginning of a line, indicates that the syntax diagram is continued from the previous line.
- The ——►◄ symbol indicates the end of the syntax diagram.

Within the syntax diagram, items on the line are required, items below the line are optional, and items above the line are defaults. See the examples in Table 1 on page xxii.

Table 1. Examples of Syntax Diagram Conventions

Syntax Diagram Convention	Example
Keywords and Constants A keyword or constant appears in uppercase letters. In this example, you must specify the item KEYWORD as shown. In most cases, you can specify a keyword or constant in uppercase letters, lowercase letters, or any combination. However, some applications may have additional conventions for using all-uppercase or all-lowercase.	►►— KEYWORD —►◄

Table 1. Examples of Syntax Diagram Conventions (continued)

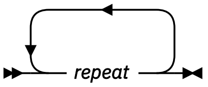
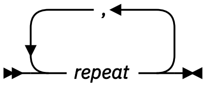
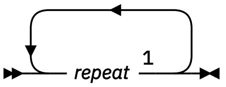
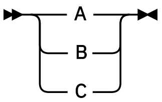
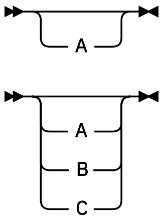
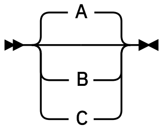
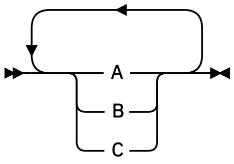
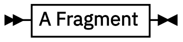
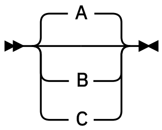
Syntax Diagram Convention	Example
Abbreviations Uppercase letters denote the shortest acceptable abbreviation of an item, and lowercase letters denote the part that can be omitted. If an item appears entirely in uppercase letters, it cannot be abbreviated. In this example, you can specify KEYWO, KEYWOR, or KEYWORD.	►► KEYWOrd ►◄
Symbols You must specify these symbols exactly as they appear in the syntax diagram.	***** Asterisk **:** Colon **,** Comma **=** Equal Sign **-** Hyphen **()** Parentheses **.** Period
Variables A variable appears in highlighted lowercase, usually italics. In this example, *var_name* represents a variable that you must specify following KEYWORD.	►► KEYWOrd — *var_name* ►◄
Repetitions An arrow returning to the left means that the item can be repeated. A character within the arrow means that you must separate each repetition of the item with that character. A number (1) by the arrow references a syntax note at the bottom of the diagram. The syntax note tells you how many times the item can be repeated. Syntax notes may also be used to explain other special aspects of the syntax.	   Notes: [1] Specify *repeat* up to 5 times.
Required Item or Choice When an item is on the line, it is required. In this example, you must specify A. When two or more items are in a stack and one of them is on the line, you must specify one item. In this example, you must choose A, B, or C.	►► A ►◄ 

Table 1. Examples of Syntax Diagram Conventions (continued)

Syntax Diagram Convention	Example
Optional Item or Choice When an item is below the line, it is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	
Defaults When an item is above the line, it is the default. The system will use the default unless you override it. You can override the default by specifying an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	
Repeatable Choice A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	
Syntax Fragment Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named "A Fragment."	 **A Fragment** 

Examples of Messages and Responses

Although most examples of messages and responses are shown exactly as they would appear, some content might depend on the specific situation. The following notation is used to show variable, optional, or alternative content:

xxx
: Highlighted text (usually italics) indicates a variable that represents the data that will be displayed.

[ ]
: Brackets enclose optional text that might be displayed.

{ }
: Braces enclose alternative versions of text, one of which will be displayed.

|
: The vertical bar separates items within brackets or braces.

...
: The ellipsis indicates that the preceding item might be repeated. A vertical ellipsis indicates that the preceding line, or a variation of that line, might be repeated.

Where to Find More Information

For more information about z/VM functions, see the documents listed in the "Bibliography" on page 343.

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See How to send feedback to IBM for additional information.

Summary of Changes for z/VM: Connectivity

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (|) to the left of the change.

SC24-6267-73, z/VM 7.3 (July 2025)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.3.

[7.3 VM66823, VM66853, VM66857] z/VM support for the IBM z17 family

With the PTFs for APARs VM66823 and VM66853 (7.3 CP) and VM66857 (7.3 RACF), z/VM 7.3 provides support for the IBM z17 family. For more information, see z/VM support for the IBM z17 family in the *z/VM: Migration Guide*.

Of particular importance for z/VM connectivity is the following support:

- Real device support — Allows the following to be attached for guest exploitation:
  - Enhanced QDIO (OSH) devices.
  - CL6 CHPID coupling adapter.
  - Network Express Hybrid (NETH) devices.
- Guest exploitation support for EQDIO devices — Allows guests to directly use the OSH functions of the Network Express Adapter.
- Extension of CP VSWITCH logic for Network Express Adapter EQDIO support within the z/VM VSwitch — Allows customers to configure the VSwitch to take advantage of lower latency and higher bandwidths that are provided by networking EQDIO devices within their data center. The VSwitch EQDIO exploitation includes QDIO-to-EQDIO translation, which allows guests that do not support EQDIO to directly take advantage of this networking support.

Many topics explain concepts that apply to the new Network Express adapters in the same way that they apply to the OSA-Express adapters. These topics are updated to include mention of Network Express or EQDIO where previously only OSA-Express or QDIO was mentioned. In some cases, the generic term "OSA" is used to indicate both OSA-Express and Network Express.

The following topics are created or updated to identify differences in support between OSA-Express adapters and Network Express adapters.

In the Chapter 1, “Introduction to z/VM Connectivity,” on page 3 section:

- “OSA Adapters: OSA-Express and Network Express” on page 5

In the Chapter 2, “Introduction to Connectivity Terminology,” on page 15 section:

- “Virtual Network Terminology” on page 15
- “What is QDIO and EQDIO?” on page 16
- “What Is OSA-Express?” on page 16
- “What Is Network Express?” on page 17
- “What Is a Network Interface Card (NIC)?” on page 17
- “What Is a Virtual Switch Controller?” on page 18

In the Chapter 3, “Networking Options in z/VM,” on page 39 section:

- “z/VM Network with Network Express Adapter” on page 40
- “Virtual Switch” on page 41

In the Chapter 4, “Planning for Guest LANs and Virtual Switches,” on page 45 section:

- Chapter 4, “Planning for Guest LANs and Virtual Switches,” on page 45
- “Guest LAN Environment” on page 45
- “Virtual Switch Environment” on page 46
- “Transport Mode — IP or Ethernet” on page 50
- “Additional Considerations for Virtual Switch Configurations” on page 54
- “Virtual Switch Priority Queuing Function” on page 55
- “IVL Virtual Switch Priority Queuing” on page 55
- “OSA-Express Multiple Ports Function” on page 56
- “Guest Port Traffic Forwarding Strategies for the Virtual Switch” on page 57
- “VEPA Mode” on page 60
- “Servicing Virtual Switch OSA adapters” on page 67
- “Virtual Switch Failover” on page 67
- “VSwitch Queue Hang Detection” on page 70
- “Virtual Switch Topology Considerations” on page 70
- “Virtual Machine Routing for QDIO” on page 71
- “Creating a Simulated NIC on Each Virtual Machine” on page 79
- “Coupling the Simulated NIC to the Guest LAN” on page 79
- “Example — Creating and Destroying a QDIO Guest LAN” on page 80

In the Chapter 5, “Planning a z/VM Environment with OSA-Express and HiperSockets,” on page 83 section:

- Chapter 5, “Planning a z/VM Environment with OSA-Express and HiperSockets,” on page 83
- “HiperSockets Network” on page 85

In the Chapter 6, “Live Guest Relocation Networking Considerations,” on page 89 section:

- “Real Networking Devices” on page 89
- “Relocation Eligibility Checks” on page 91

In the Chapter 7, “Bridging a HiperSockets LAN with a z/VM Virtual Switch,” on page 95 section:

- “Construction of a Bridged HiperSockets LAN” on page 100
- “OSD and OSH CHPID Definitions” on page 102
- “z/VM Virtual Switch Bridge Port Definition” on page 103

In the Appendix F, “Creating a VSwitch Controller,” on page 337 section:

- Appendix F, “Creating a VSwitch Controller,” on page 337

SC24-6267-73, z/VM 7.3 (December 2023)

This edition includes terminology, maintenance, and editorial changes.

SC24-6267-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

Part 1. Introduction to Connectivity

The topics in this part offer an introduction to z/VM connectivity. The topics explain the components, protocols, and technologies that enable communication between virtual machines, systems, and networks within the z/VM environment. Key networking options like TCP/IP, APPC/VM, CPI Communications, SNA integration via AVS and VTAM, virtual channel-to-channel adapters, guest LANs, virtual switches, OSA adapters, and HiperSockets are covered. The information also addresses program-to-program communication across systems using TSAF and ISFC for transparent routing and SNA interoperability, along with guidance on planning, setup, operation, troubleshooting, and security for managing complex virtualized network environments on IBM Z® systems.

This part contains the following chapters:

- Chapter 1, "Introduction to z/VM Connectivity," on page 3 describes the communication services and connectivity features of z/VM and other related IBM products.
- Chapter 2, "Introduction to Connectivity Terminology," on page 15 defines the terms used in z/VM connectivity, the SNA network, and the Advanced Program-to-Program Communications (APPC) protocol.

Related Information: See the following sources for more information on the tasks in this section:

Task	Source
Using z/VM connectivity solutions	*z/VM: General Information*
Planning and customizing TCP/IP	*z/VM: TCP/IP Planning and Customization*
Using TCP/IP	*z/VM: TCP/IP User's Guide*
Programming with TCP/IP functions and routines	*z/VM: TCP/IP Programmer's Reference*
Programming with APPC/VM functions and routines	*z/VM: CP Programming Services*
Programming with SAA CPI-Communications routines	*Common Programming Interface Communications Reference (https://publibfp.dhe.ibm.com/epubs/pdf/c2643999.pdf)*
Understanding Systems Network Architecture concepts	*SNA Transaction Programmer's Reference Manual for LU Type 6.2* *SNA Technical Overview*
Planning and customizing guest LANs and virtual switches	*z/VM: CP Commands and Utilities Reference* *z/VM: CP Planning and Administration*

Chapter 1. Introduction to z/VM Connectivity

The topics in this chapter provide an introduction to z/VM connectivity, including the components, protocols, and facilities that enable communication between virtual machines, systems, and networks within and beyond the z/VM environment. The topics cover key technologies such as TCP/IP communications, APPC/VM, CPI Communications, and SNA network integration via AVS and VTAM.

Various networking options including virtual channel-to-channel adapters, guest LANs, virtual switches, OSA adapters (OSA-Express and Network Express), HiperSockets, and their roles in facilitating efficient data exchange are explained. z/VM support for program-to-program communication is described, both within a single system and across collections of systems using TSAF and ISFC, which enables transparent routing and interoperability with SNA networks.

This chapter also outlines the planning, setup, operation, troubleshooting, and security considerations for connectivity features. The information provides a foundational guide for managing complex virtualized network environments on IBM Z systems.

What is Connectivity?

In general terms, connectivity is the ability to connect systems or application programs. Ideally, these connections are established without requiring many changes to the applications or the systems on which they run. Application programs may need to communicate with each other to complete transactions or to effectively balance resources at an installation.

However, application programs are often written in different programming languages and processors may use different operating systems or they may be in different locations. To enable communications, application programs must follow common rules and a physical connection must be established between the processors on which the programs run.

For communications among guest systems within a z/VM image, you can define a virtual network that deploys Internet Protocol (Layer 3) or Ethernet (Layer 2) as the transport. Either of these transports supports Transmission Control Protocol/Internet Protocol (TCP/IP) communications. The Ethernet transport also provides the ability to support non-IP based applications on the virtual network (such as SNA, IPX, etc.).

IBM's Systems Network Architecture (SNA) defines a set of communications functions and protocols for sending data between systems. The set of protocols that application programs most commonly use is called LU Type 6.2 or Advanced Program-to-Program Communications (APPC). The z/VM implementation of the APPC communications functions is called Advanced Program-to-Program Communications/VM (APPC/VM).

IBM's Systems Application Architecture® (SAA) also defines a set of programming languages and common programming interfaces that are common across different operating systems. Application programs that are written in SAA languages and use only SAA common programming interfaces are portable between different computer systems. The interfaces to SAA functions for functions that use APPC are called Common Programming Interfaces (CPI) Communications. CPI Communications is a common high-level language and REXX programming interface for APPC.

z/VM provides the following components and facilities that enable logical connections between systems and application programs that use TCP/IP, APPC/VM, or CPI-Communications protocols:

- APPC/VM VTAM Support (AVS)
- Control Program (CP)
- Conversational Monitor System (CMS)
- Inter-System Facility for Communications (ISFC)
- TCP/IP for z/VM
- Transparent Service Access Facility (TSAF)

These components and facilities support the APPC/VM and CPI-Communications protocols between application programs. Application programs that use these protocols can use the logical connections provided by z/VM to communicate and exchange data.

CMS and CP support communications between applications that reside on the same z/VM system. CMS supports CPI-Communications protocols for application programs and CP supports APPC/VM communications between application programs. When application programs reside in a collection of systems, TSAF and ISFC provide the services necessary to route communications between the application programs. AVS provides services that let application programs communicate with other programs on z/VM or non-z/VM systems in an SNA network. TCP/IP supports communications between networks through universal communication services (see "TCP/IP Communications" on page 4 below for more information).

TCP/IP Communications

TCP/IP provides connectivity and gateway functions that handle the physical interfaces and the routing of data. TCP/IP allows you to build an interconnection between networks through universal communication services. To be able to communicate between networks, addresses are assigned to each host on the network called the IP address.

Two virtual connectivity options for connecting one or more virtual machines (VM guests) are virtual channel-to-channel adapters (CTCA) and the Inter-User Communications Vehicle (IUCV) facility. These virtual interfaces are classified as point-to-point connections. While the bandwidth of point-to-point connections is considerable and thus affords the rapid movement of large amounts of data between guests, these interfaces have a number of drawbacks.

Using CTCA links as an example, in order for two guests to communicate with each other, you must define CTCA device pairs in each guest and couple those devices between the guest machines. Another requirement when using point-to-point connections is the definition of static routing statements in each guest that needs to communicate with any other guest in the system. Finally, another limitation is that if one side of the point-to-point connection goes down, it is often difficult to subsequently reconnect the two guests. Frequently, one of the Linux® guest machines has to be rebooted in order to reestablish the connection.

CP provides a virtual connectivity feature known as guest LAN (and an extension of a guest LAN called a virtual switch). This feature allows you to create multiple virtual LAN segments within a z/VM environment.

Note: While the structures and simulated devices related to the guest LAN under z/VM are "virtual", we use the term guest LAN and not Virtual LAN, because the term Virtual LAN (VLAN) has a different meaning in the networking world.

There is no architectural limit on the number of guest LAN segments that you can create. However, you are limited by the amount of machine resources you have available.

In contrast to point-to-point connections, individual guest machines create a virtual network interface card (NIC) to connect to a guest LAN. They can then connect this NIC to the LAN and communicate with other guests using standard TCP/IP protocols.

TCP/IP provides the following functions for a z/VM network:

- Connectivity and gateway functions, which handle the physical interfaces and routing of data.
- Server functions, which provide a service to a client (for example, to send or transfer a file).
- Client functions, which request a certain service from a server anywhere in the network.
- Application Programming Interfaces, which allow you to write your own client/server applications.

TCP/IP network communication allows application programs to talk with each other without regard to the hardware and operating systems where they are run. This allows application programs to communicate independently of their physical network connections.

OSA Adapters: OSA-Express and Network Express

OSA Adapter

Open Systems Adapter (OSA) is a class of networking adapters. An OSA adapter is an integrated hardware feature that supports many networking transport protocols and provides connectivity from an IBM Z system to an external Local Area Network (LAN). An OSA adapter integrates the control unit and device into the same hardware. It does so by placing both elements on a single card that directly connects to the central processor complex I/O bus. The adapter improves data transfer speed and efficiency for TCP/IP traffic when compared with CCW-based I/O.

z/VM supports the real OSA adapter, which provides connectivity to an external LAN. z/VM provides a virtualized OSA interface for z/VM guests, which supports virtual switches and virtual network interface cards (NICs).

OSA adapters include the OSA-Express and Network Express families of adapters. When the term "OSA" appears without modification, it is a general term that can include OSA-Express adapters or Network Express adapters or both.

Exploitation of a particular OSA function by z/VM might require a particular OSA adapter.

OSA-Express and QDIO

OSA-Express is a generation of Open Systems Adapter (OSA) technology that uses Queued Direct I/O (QDIO) architecture. QDIO allows a host to directly exchange data with an I/O device without using traditional I/O instructions. Data transfer is initiated and completed by referencing main storage directly via a set of data queues by both the I/O device and the host. Once the host establishes and activates the data queues, minimal processor intervention is required to complete the direct exchange of data.

With the OSA-Express adapter, the virtual OSA interface that z/VM provides to z/VM guests supports only QDIO mode. The uplink port to an external LAN also supports only QDIO mode.

OSA-Express supports both IP and Ethernet transport modes.

The CHPID type that supports QDIO is OSD (OSA direct).

Network Express and EQDIO

The Network Express family of network adapters extends OSA technology. Network Express is introduced with the IBM z17 family of processors and is optimized for mainframe Ethernet local area network (LAN) connections. The Network Express adapter leverages the IBM On Chip I/O Processor and supports Enhanced Queued Direct I/O (EQDIO) architecture.

The z/VM VSwitch EQDIO exploitation includes QDIO-to-EQDIO translation, which allows z/VM guests that do not support EQDIO to use and benefit from the higher bandwidth and lower latency of Network Express adapters.

The uplink port to an external LAN supports EQDIO mode.

The CHPID type that supports EQDIO is OSH (OSA hybrid).

A virtual switch that uses a Network Express (EQDIO) adapter as the uplink device does not require a virtual switch controller because the Network-Express adapter is controlled by the z/VM Control Program (CP).

Network Express converges RoCE and OSA networking features into a single network feature.

Network Express uses only Ethernet transport mode.

Network Express supports Remote Direct Memory Access (RDMA) over Converged Ethernet Express.

HiperSockets

HiperSockets is an extension to the QDIO Hardware Facility that provides a microcode-only vehicle for IP inter-program communications (IPC). This is sometimes also referred to as iQDIO. Communications between programs is accomplished with TCP/IP socket connections. Using HiperSockets, a program has the ability not only to directly communicate with a program running within the same logical partition (LP), but also to communicate across any logical partition within the same Central Electronics Complex (CEC) or processor.

Guest LAN Intercommunication among Virtual Machines

z/VM supports the function called guest LAN for intercommunication among virtual machines. See Figure 1 on page 6 for an example. z/VM simulates both the HiperSockets function and OSA-Express (QDIO) adapters for communication among virtual machines without the need for HiperSockets microcode or an OSA-Express. A group of virtual machines can use each guest LAN to communicate among the group, independent of other groups of virtual machines on another guest LAN.

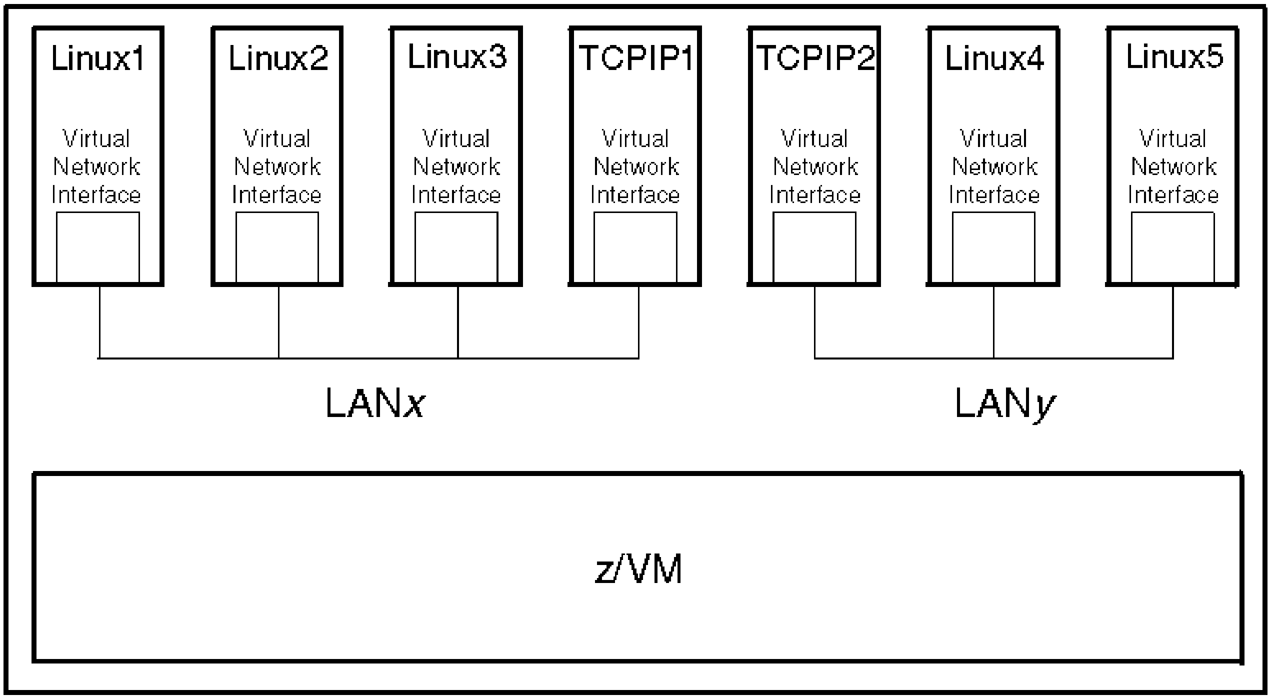

Linux1
Virtual Network Interface
Linux2
Virtual Network Interface
Linux3
Virtual Network Interface
TCPIP1
Virtual Network Interface
TCPIP2
Virtual Network Interface
Linux4
Virtual Network Interface
Linux5
Virtual Network Interface
LAN*x*
LAN*y*
z/VM

Figure 1. z/VM Guest LAN

APPC Communications between Programs

z/VM uses APPC as its program-to-program communication protocol. APPC/VM is the z/VM implementation of the APPC communications functions. APPC is divided into functional pieces called sets. Most of the communication functions APPC provides are contained in what is known as the base set. This base set of functions is implemented by all APPC products. Additional APPC functions are defined in option sets that APPC products may choose to implement.

An assembler language programming interface is provided for APPC/VM. Like each product that implements APPC, APPC/VM defines its own set of functions and protocols that map to those defined by APPC. APPC/VM provides the base set (excluding mapped conversation) of APPC functions and many of the option sets. See *z/VM: CP Programming Services* for a list of APPC functions provided by APPC/VM and a mapping of the APPC/VM functions to the APPC architecture.

z/VM, through CMS, also supports the SAA CPI-Communications high-level language and REXX programming interface for APPC. The CPI Communications programming interface provides functions similar to those provided by APPC/VM. See the *Common Programming Interface Communications*

Reference (https://publibfp.dhe.ibm.com/epubs/pdf/c2643999.pdf) for information about the CPI-Communications routines.

Figure 2 on page 7 shows how programs can use the CPI-Communications or the APPC/VM programming interfaces for program-to-program communication. Using the services provided by z/VM, an APPC/VM or CPI-Communications program can communicate with other programs on:

- The same z/VM system
- Different z/VM systems
- Non-z/VM systems

Note: Unless otherwise stated in this document, the term APPC program refers to programs using the APPC/VM assembler language programming interface or the CPI Communications high-level language and REXX programming interface.

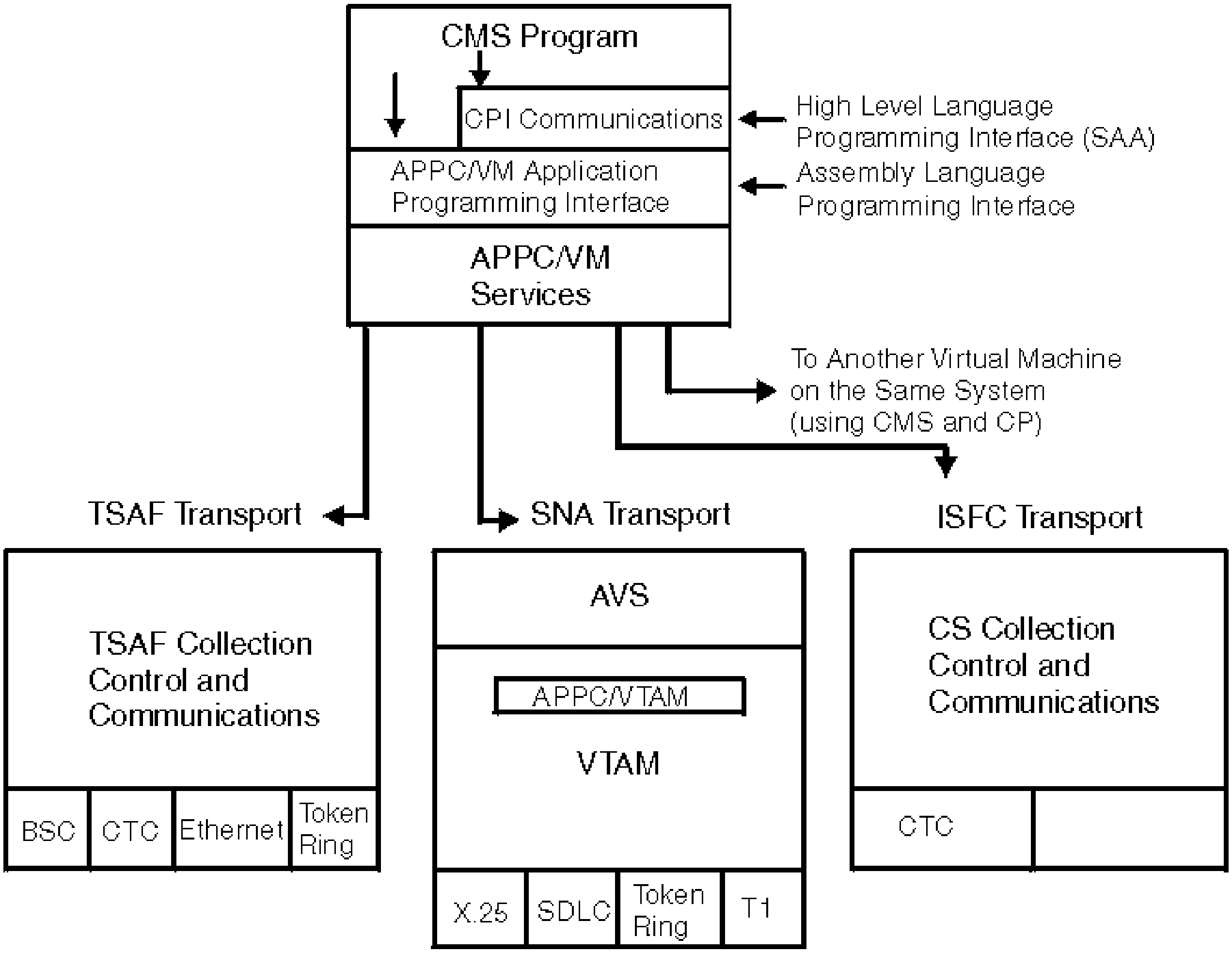

CMS Program
CPI Communications
High Level Language Programming Interface (SAA)
APPC/VM Application Programming Interface
Assembly Language Programming Interface
APPC/VM Services
To Another Virtual Machine on the Same System (using CMS and CP)
TSAF Transport
SNA Transport
ISFC Transport
TSAF Collection Control and Communications
BSC
CTC
Ethernet
Token Ring
AVS
APPC/VTAM
VTAM
X.25
SDLC
Token Ring
T1
CS Collection Control and Communications
CTC

Figure 2. z/VM Connectivity Support for APPC/VM and CPI Communications

APPC Communications within a z/VM System

An APPC/VM program running in your virtual machine can request services from a program running in another virtual machine in the same system. CMS and CP handle communications between the virtual machines. See Part 3, “APPC Server and Requester Virtual Machines,” on page 163 for information about setting up the CMS virtual machines in which APPC/VM programs run.

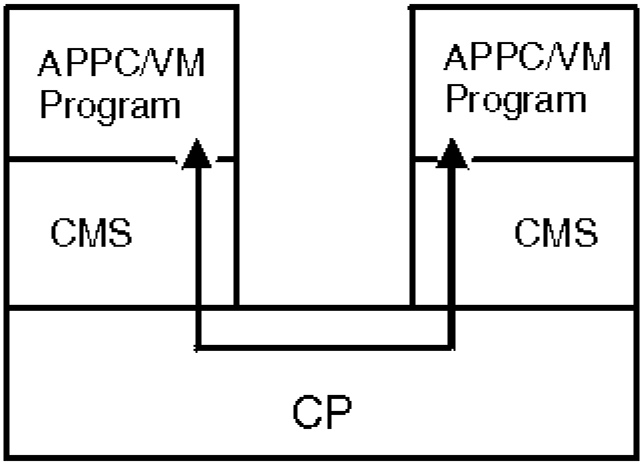


Figure 3. APPC Communications within One z/VM System

For example, Figure 3 on page 8 shows how data is exchanged between APPC/VM programs running on the same system. Each tower represents a CMS virtual machine running on CP; each APPC/VM program runs in a CMS virtual machine. When the APPC/VM programs exchange data, the communications requests pass through CMS and CP on the system.

APPC Communications within a Collection of z/VM Systems

TSAF and ISFC, which are provided with z/VM, provide communication services between application programs that reside in a collection of z/VM systems. z/VM supports TSAF collections, which are created by TSAF, and Communication Services (CS) collections, which are formed using ISFC. This section provides an overview of TSAF; see Part 4, “TSAF Virtual Machine,” on page 185 for more information about the TSAF virtual machine. For an introduction to ISFC, see “APPC Communications between z/VM Systems” on page 9.

The TSAF component routes communication requests between application programs that reside on other systems. This routing is transparent to the application program; that is, communications between the application programs proceed as if the programs were running on the same system.

TSAF runs in a virtual machine on a z/VM system. When a TSAF virtual machine establishes a link to a TSAF virtual machine on another system, a TSAF collection is formed. A TSAF collection may contain a maximum of eight z/VM systems. APPC programs on a z/VM system can communicate with other programs on other systems within the TSAF collection.

For example in Figure 4 on page 9, two systems have formed a TSAF collection. An APPC program requests to access a database that is managed by an APPC program on another system. This request passes through CMS, CP, and the TSAF virtual machine to the other system and on to the database server program. Because the TSAF virtual machine provides a path to the other system, the APPC programs can communicate without knowing each other's location.

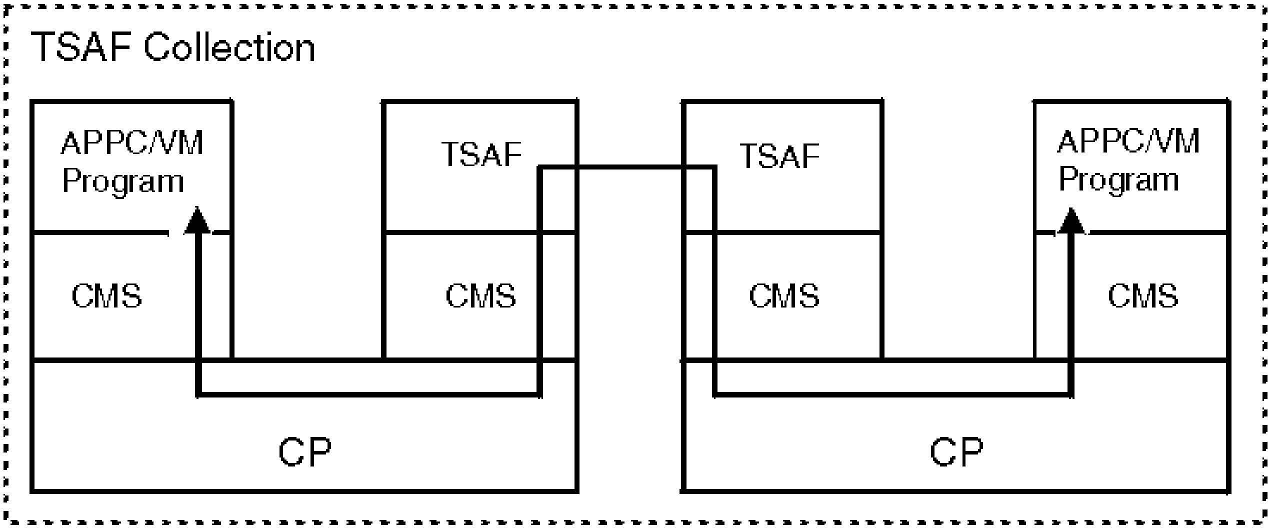


Figure 4. Communication in a TSAF Collection of z/VM Systems

APPC Communications between z/VM Systems

z/VM also provides communications support using ISFC, which is a function provided in CP. Using the services that ISFC provides, application programs running on a z/VM system can exchange data with programs that run on other z/VM systems; this is similar to the communications support provided by TSAF.

In a CS collection, z/VM systems that run ISFC are called z/VM domain controllers. A domain controller acts as a program-to-program communications gateway between the z/VM systems in the CS collection.

As Figure 5 on page 10 shows, programs on z/VM systems can be written to the CPI Communications or the APPC/VM assembler language programming interface.

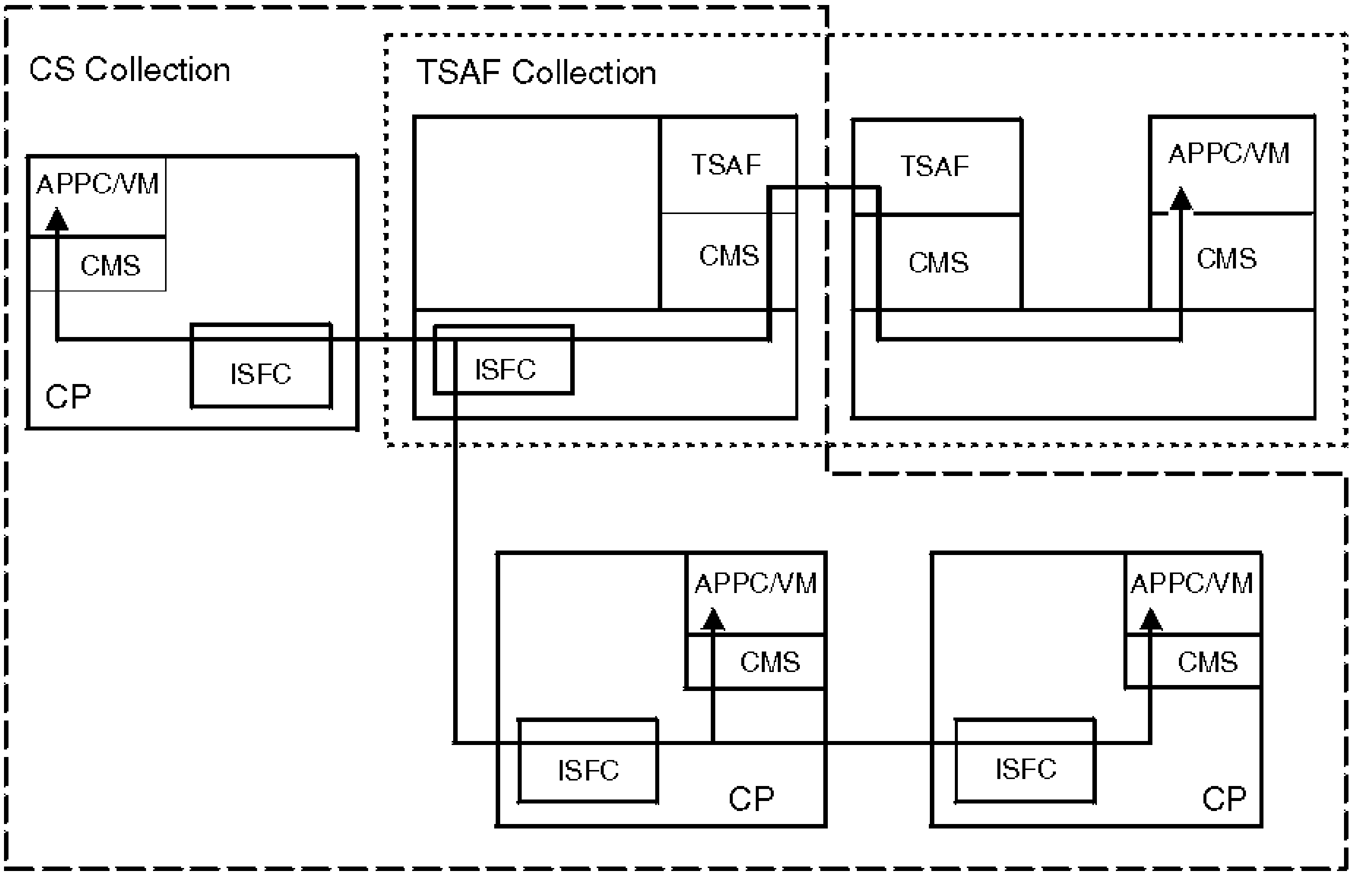


Figure 5. APPC/VM Communication in a CS Collection

See Part 6, “Planning for ISFC,” on page 299 for more information about ISFC and CS collections.

IUCV Communications between z/VM Systems

ISFC also allows IUCV programs to communicate with IUCV programs on other z/VM systems.

As Figure 6 on page 11 shows, programs on z/VM systems can be written to the IUCV assembler language programming interface.

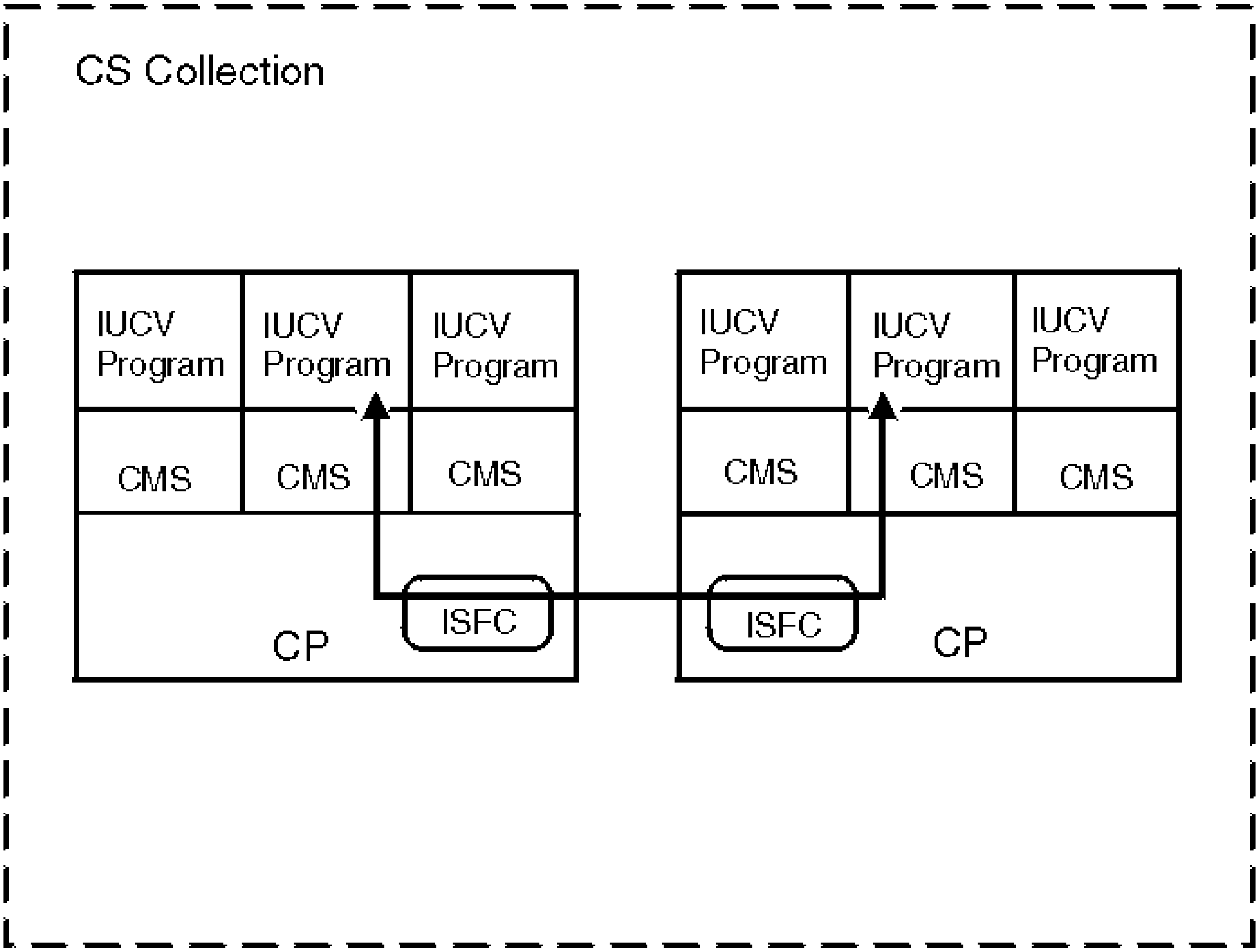


Figure 6. IUCV Communication in a CS Collection

See Part 6, "Planning for ISFC," on page 299 for more information about ISFC and CS collections.

APPC Communications with an SNA Network

The AVS component of z/VM works with the Advanced Communications Function for Virtual Telecommunications Access Method (ACF/VTAM) licensed program to provide communication services between z/VM and non-z/VM systems in the SNA network.

ACF/VTAM (which is referred to as VTAM) controls telecommunications activity and interprocessor communication in the SNA network. VTAM directs data transfer between programs and devices, such as terminals, and between different programs. On z/VM, VTAM runs in a virtual machine in a Group Control System (GCS) group. GCS, which is a component of z/VM, is a virtual machine supervisor that manages multiple tasks. It manages multiple subsystems that support an SNA network and provides an interface between these subsystems and CP.

AVS provides the connection between ACF/VTAM and APPC/VM. This means that your APPC/VM application programs running in a TSAF or CS collection can communicate with APPC and APPC/VM programs running on a system in the SNA network. AVS and VTAM must run in the same GCS group on a z/VM system. Together, AVS and VTAM enable APPC/VM programs in the TSAF or CS collection to communicate with:

- Other APPC/VM programs residing in other z/VM systems within the SNA network
- APPC programs residing in non-z/VM systems in the SNA network

See Part 5, "AVS Virtual Machine," on page 243 for more information about the AVS virtual machine.

SNA Network Communications between TSAF or CS Collections

APPC programs running on a z/VM system in a TSAF or CS collection can also communicate with APPC programs that run in a TSAF or CS collection in the SNA network. AVS, GCS, and VTAM, working together in a GCS group, provide the SNA communications services. TSAF and ISFC enable systems to form a TSAF or CS collection, respectively.

As Figure 7 on page 12 shows, AVS and VTAM connect two collections (each made up of one z/VM system) in the SNA network. An APPC/VM program, running in a virtual machine in a TSAF or CS collection, requests data from a database. The database server program is located in a virtual machine in another TSAF or CS collection. AVS translates information between APPC/VM and APPC/VTAM (the VTAM implementation of APPC). VTAM provides a path between the two collections in the SNA network. Because VTAM, AVS, and the database server use the APPC protocol, the application programs can communicate.

When the application program requests data, the request is passed to AVS. The request then passes over the path established by VTAM to the TSAF or CS collection in the SNA network. AVS translates the request from APPC/VTAM to APPC/VM. The request then passes to the virtual machine that is running the database server program. In a TSAF collection, the request is not routed through a TSAF virtual machine if the AVS virtual machine and the database server reside on the same system.

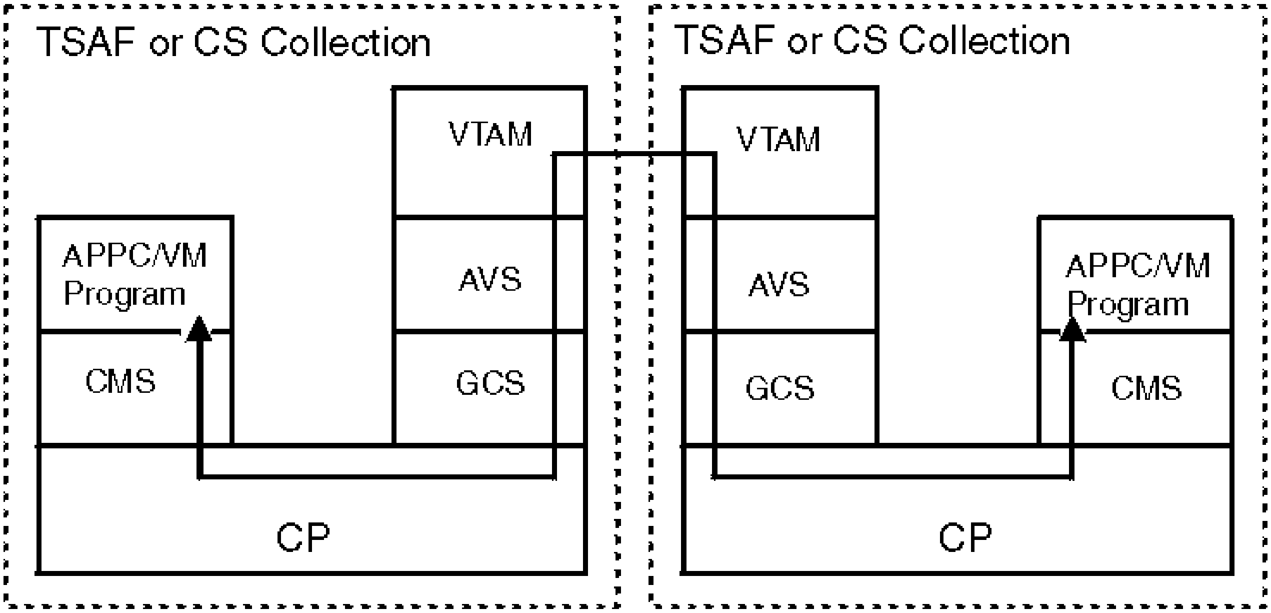


Figure 7. Communication between Two Collections in the SNA Network

SNA Network Communications to Other Systems

AVS and VTAM also enable z/VM systems in a TSAF or CS collection to connect to other systems in the SNA network that support the LU 6.2 protocol. AVS translates information between the APPC/VM and APPC/VTAM protocols. VTAM controls the path (an LU 6.2 session) between the collection and the system in the SNA network.

For example in Figure 8 on page 13, an APPC/VM program in a TSAF collection receives a request from an APPC program that is running in a non-z/VM system in the SNA network. VTAM receives the APPC request sent to the TSAF collection and AVS translates the request from APPC/VTAM to APPC/VM. TSAF then routes the connection to the APPC/VM program because AVS and the program are not in the same system.

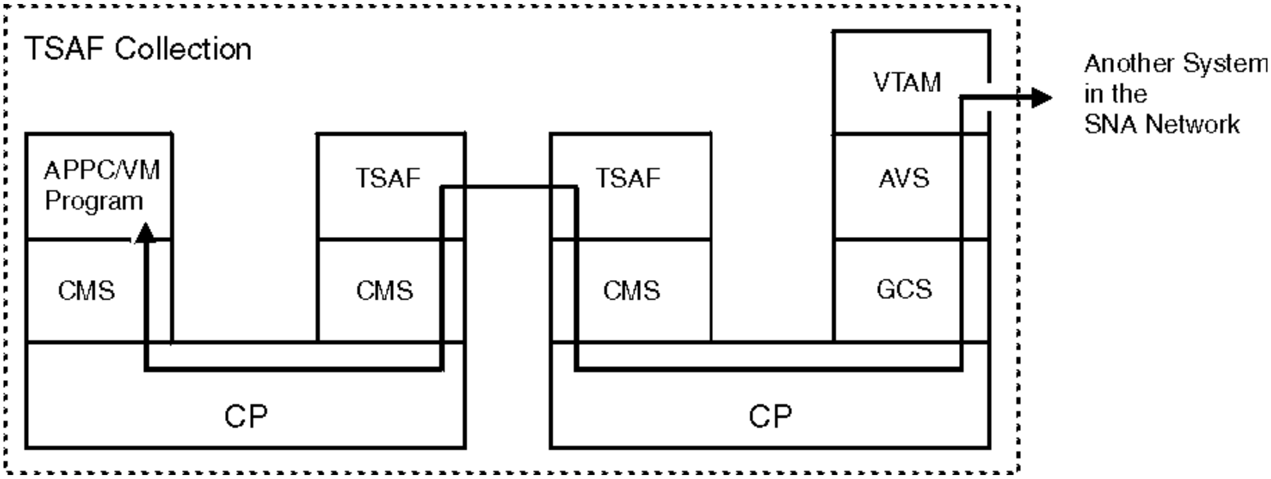


Figure 8. Communication between the TSAF Collection and another System in the SNA Network

Summary of APPC Communications

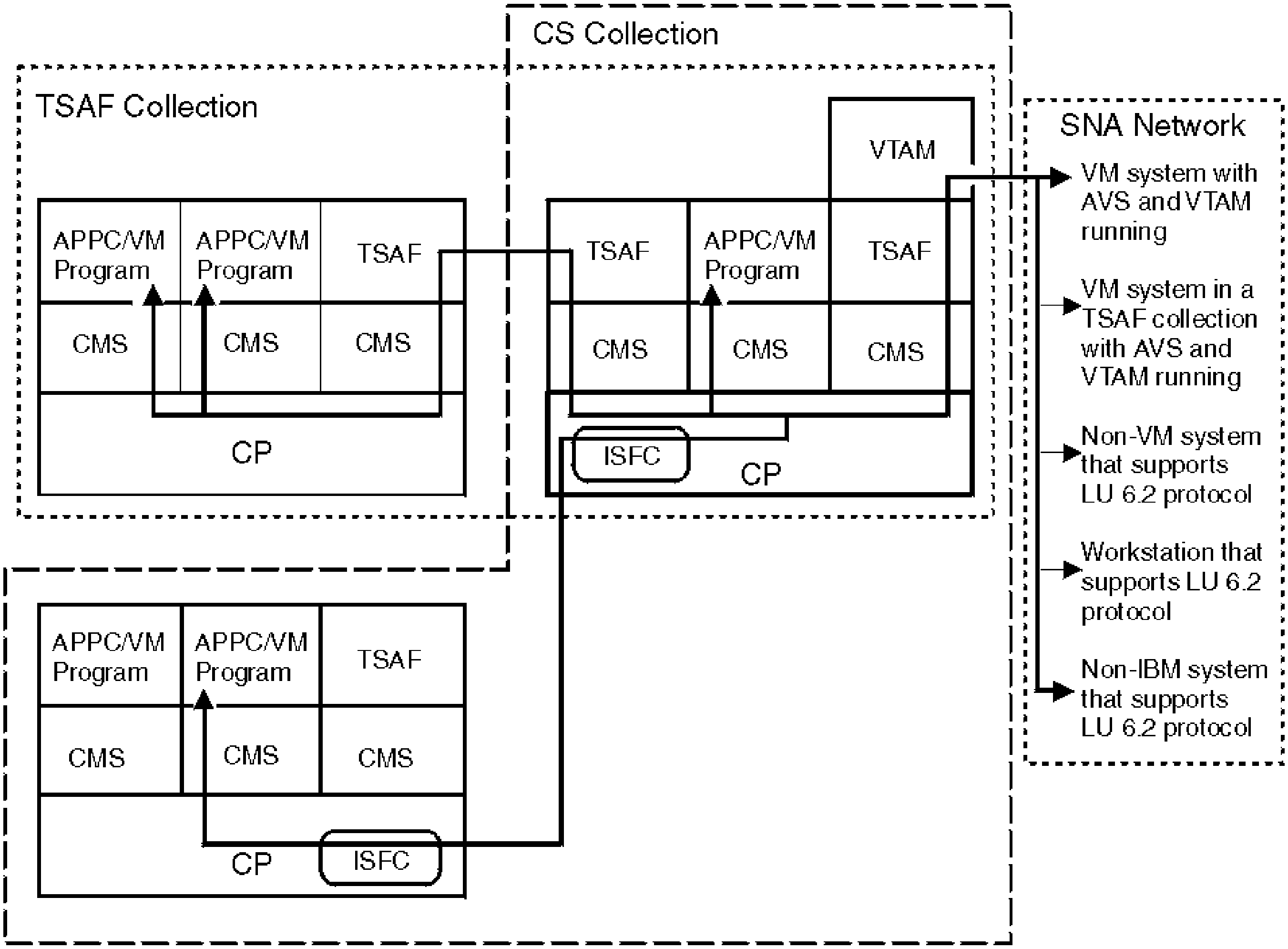

CS Collection
TSAF Collection
APPC/VM Program
APPC/VM Program
TSAF
CMS
CMS
CMS
CP
TSAF
APPC/VM Program
TSAF
VTAM
CMS
CMS
CMS
ISFC
CP
SNA Network
VM system with AVS and VTAM running
VM system in a TSAF collection with AVS and VTAM running
Non-VM system that supports LU 6.2 protocol
Workstation that supports LU 6.2 protocol
Non-IBM system that supports LU 6.2 protocol
APPC/VM Program
APPC/VM Program
TSAF
CMS
CMS
CMS
CP
ISFC

Figure 9. Summary of APPC Communications

As Figure 9 on page 13 shows, APPC/VM, CPI Communications, and the connectivity support provided by z/VM enable application programs on your system to communicate with application programs in the following locations:

- The same z/VM system

- Another z/VM system in the same TSAF or CS collection
- z/VM system in an SNA network that has AVS and VTAM running
- z/VM system, running AVS and VTAM, in another TSAF or CS collection
- Workstations in an SNA network that support the LU 6.2 APPC protocol
- Systems in an SNA network that support the LU 6.2 APPC protocol

Chapter 2. Introduction to Connectivity Terminology

The topics in this chapter are a comprehensive guide to connectivity terminology and concepts used in IBM z/VM environments. The information is a foundational reference for understanding and planning network connectivity, resource management, and secure communications within and across z/VM systems and SNA networks.

The focus is on virtual networking and APPC communications supported by TSAF, AVS, and ISFC. Key virtual network components such as QDIO/EQDIO architectures, OSA-Express and Network Express adapters, HiperSockets, NICs, guest LANs, virtual switches, VLANs, port types, link aggregation (LAG), and related protocols like LACP are defined. Systems Network Architecture (SNA) terms are covered, including logical units, sessions, transaction programs, conversations, security levels, and negotiation processes. z/VM-specific terms such as *IDENT service, TSAF and CS collections, domains, resource types (local, global, system, private), communication partners, resource managers, user programs, and various gateway types (AVS global/private gateways and system gateways) are explained.

Virtual Network Terminology

The following virtual networking terms are briefly described in this section.

- QDIO and EQDIO
- OSA-Express
- Network Express
- HiperSockets
- Network interface card (NIC)
- Guest LAN
- Virtual Switch (VSwitch)
- Virtual Switch Controller
- Global Virtual Switch
- Transport type
- Virtual LAN (VLAN)
- Tagged and untagged frames
- VLAN unaware
- Port type
- Access port
- Trunk port
- Native VLAN ID
- Port VLAN ID (pvid)
- Ingress rules
- Egress rules
- Global VLAN ID
- Link Aggregation Port Group (LAG)
- Exclusive Port Group
- Shared Port Group
- Multi-VSwitch LAG Configuration
- LACP
- LAG Port Controller

- UPLINK port
- HiperSockets Bridge port
- QEBSM
- HiperSockets guest port
- HiperSockets Bridge Capable Port
- PMTUD
- Difference between a Port Based and a User Based Virtual Switch
- Directory Network Authorization (DNA)
- IVL (Inter-VSwitch Link) Network
- IVL Virtual Switch
- IVL Domain
- Virtual Switch Priority Queuing

What is QDIO and EQDIO?

QDIO

QDIO is Queued Direct I/O architecture. QDIO allows a host to directly exchange data with an I/O device without using traditional I/O instructions. Data transfer is initiated and completed by referencing main storage directly via a set of data queues by both the I/O device and the host. Once the host establishes and activates the data queues, minimal processor intervention is required to complete the direct exchange of data.

A virtual switch that uses an OSA-Express adapter requires a virtual switch controller.

OSA-Express adapters operate in QDIO mode.

QDIO uses CHPID type OSD (OSA direct).

EQDIO

EQDIO is Enhanced QDIO architecture. Like QDIO, EQDIO allows a host to directly exchange data with an I/O device without using traditional I/O instructions.

A z/VM virtual switch that uses a Network Express adapter does not require a virtual switch controller.

Network Express adapters operate in EQDIO mode, which provides enhanced connectivity.

EQDIO uses CHPID type OSH (OSA hybrid).

The Network Express adapter leverages the IBM On Chip I/O Processor, which is available with the IBM IBM z17 family processor family and later processor families.

What Is OSA-Express?

OSA-Express is a generation of Open Systems Adapter (OSA) technology that uses Queued Direct I/O (QDIO) architecture. QDIO allows a host to directly exchange data with an I/O device without using traditional I/O instructions. Data transfer is initiated and completed by referencing main storage directly via a set of data queues by both the I/O device and the host. Once the host establishes and activates the data queues, minimal processor intervention is required to complete the direct exchange of data.

With the OSA-Express adapter, the virtual OSA interface that z/VM provides to z/VM guests supports only QDIO mode. The uplink port to an external LAN also supports only QDIO mode.

OSA-Express supports both IP and Ethernet transport modes.

The CHPID type that supports QDIO is OSD (OSA direct).

For additional information about functions that are supported by various OSA-Express adapters and z/VM environments, see Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf).

What Is Network Express?

The Network Express family of network adapters extends OSA technology. Network Express is introduced with the IBM z17 family of processors and is optimized for mainframe Ethernet local area network (LAN) connections. The Network Express adapter leverages the IBM On Chip I/O Processor and supports Enhanced Queued Direct I/O (EQDIO) architecture.

The z/VM VSwitch EQDIO exploitation includes QDIO-to-EQDIO translation, which allows z/VM guests that do not support EQDIO to use and benefit from the higher bandwidth and lower latency of Network Express adapters.

The uplink port to an external LAN supports EQDIO mode.

The CHPID type that supports EQDIO is OSH (OSA hybrid).

A virtual switch that uses a Network Express (EQDIO) adapter as the uplink device does not require a virtual switch controller because the Network-Express adapter is controlled by the z/VM Control Program (CP).

Network Express converges RoCE and OSA networking features into a single network feature.

Network Express uses only Ethernet transport mode.

Network Express supports Remote Direct Memory Access (RDMA) over Converged Ethernet Express.

What are HiperSockets?

HiperSockets network adapters simulate queued direct I/O (QDIO) network adapters and provide high-speed TCP/IP communication between guest systems within and across logical partitions (LPARs). As with OSA-Express, z/VM virtualizes its capability within an LPAR with guest LANs and NICs.

What Is a Network Interface Card (NIC)?

A Network Interface Card (NIC) is another term for "network adapter". z/VM supports the following real network adapters:

- Network-Express (EQDIO mode)
- OSA-Express (QDIO mode)
- HiperSockets (QDIO mode)

A real NIC can attach to a z/VM virtual switch that operates in the same mode. A real NIC can also attach directly to a z/VM guest.

z/VM also provides simulated NICs (also called virtual NICs) to z/VM guests.

z/VM simulates NICs that operate in QDIO mode and NICs that operate in EQDIO mode.

What Is a Guest LAN?

A guest LAN represents a simulated LAN segment that can be connected to simulated network interface cards. There are two types of LAN segments: OSA-Express and HiperSockets. Each guest LAN is isolated from other guest LANs on the same system (unless some member of one LAN group acts as a router to other groups).

What Is a Virtual Switch?

A virtual switch is a special type of guest LAN that provides external LAN connectivity through an OSA adapter without the need for a routing virtual machine.

What Is a Virtual Switch Controller?

A virtual switch Controller is a z/VM TCP/IP Server specifically configured to provide the configuration and management required by the OSA-Express adapters that are configured to a virtual switch. This includes all link recovery operations for a virtual switch in the event of an OSA-Express Uplink port failure or a HiperSockets bridge port failure. Communications between the controller and the virtual switch is via an IUCV *VSWITCH connection. The z/VM operating system ships four preconfigured virtual switch controllers: DTCVSW1, DTCVSW2, DTCVSW3 and DTCVSW4.

Note: A Network Express (EQDIO) adapter does not require a virtual switch controller. The controller functions for EQDIO are managed within the virtual switch.

What Is a Global VSwitch?

A Global VSwitch is collection of virtual switches that share the same name and the same networking characteristics. This collection of virtual switches spans multiple systems running z/VM but logically operates as a single virtual switch. Virtual switches that are targeted to share a port group of OSA adapters (LAG) must be configured as members of a Global virtual switch.

What Is Transport Type?

Transport type is the method by which a virtual switch or guest LAN identifies, manages, and transports data through the z/VM LAN fabric. The IP transport type is Network (Layer 3) based, where the IP packet is used as the point of reference for source and destination IP addresses in transporting IP packets on the LAN. The Ethernet transport type is Data Link (Layer 2) based, where the Ethernet frame is used as the point of reference for source and destination Media Access Control (MAC) addresses in transporting Ethernet frames on the LAN.

What Is a Virtual LAN (VLAN)?

As defined by IEEE 802.1Q, a "Virtual LAN" is created by assigning artificial LAN identifiers (VLAN IDs) to the datagrams exchanged through the physical network. Hosts on the same Virtual LAN may represent a subset of the hosts on the physical network. An efficient VLAN configuration increases network traffic flow and reduces overhead by allowing the network to be organized by traffic patterns rather than by physical locations.

What Are Tagged and Untagged Frames?

"Tagged" and "untagged" are terms that refer to the presence or absence of fields in Ethernet frames in support of IEEE 802.1Q VLANs.

What Is VLAN Unaware?

"VLAN unaware" is a classification for a networking device that indicates it does not support the IEEE 802.1Q VLAN specifications for VLAN memberships and VLAN frame formats. These devices ignore the additional fields within the Ethernet frame that carry VLAN specific semantics.

What Is Port Type?

"Port type" is a classification of the type of connection (virtual port) provided to a guest virtual machine when it is coupled to a virtual switch instance. The port type defines the ingress and egress rules that are enforced by the virtual switch for this specific virtual machine connection.

What Is an Access Port?

An "access port" is a type of connection on a switch that is used to connect a guest virtual machine that is VLAN unaware. This port provides the virtual machine with connectivity through a switch that is VLAN aware without requiring it to support VLAN tagging.

What Is a Trunk Port?

A "trunk port" is a type of connection on a switch that is used to connect a guest virtual machine that is VLAN aware. Generally, all frames that flow through this port are VLAN tagged. The exception to this is when a trunk port is granted access to the untagged VLAN set (native VLAN ID).

What Is a Native VLAN ID

A native VLAN ID, (usually VLAN ID 0001) is deployed internally by the virtual switch to associate or flow untagged frames through the switching fabric. Only those guests that are configured for the native VLAN ID will receive or send untagged frames.

What Is a Port VLAN ID (pvid)?

A Port VLAN ID (pvid) is a default VLAN ID that is assigned to an access port to designate the virtual LAN segment to which this port is connected. The pvid places the port into the set of ports that are connected under the designated VLAN ID. Also, if a trunk port has not been configured with any VLAN memberships, the virtual switch's Port VLAN ID (pvid) becomes the default VLAN ID for the ports connection.

What Are Ingress Rules?

The ingress rules are a set of rules for processing a frame or packet that is received on a switch port. These rules enforce the VLAN tagging standards based on the actual port type defined. These rules apply only to inbound data on a switch port.

What Are Egress Rules?

The egress rules are a set of rules for processing a frame or packet that is sent out on a switch port. These rules ensure that the proper VLAN tagging standards are applied to the outbound data based on the actual port type defined. These rules apply only to outbound data on a switch port.

What Is a Global VLAN ID?

A GLOBAL VLAN ID is OSA's VLAN support to provide access to a virtual LAN segment for a VLAN unaware host so the host can receive and send its network traffic. This host does not tag its outbound frames nor receive tagged inbound frames. The GLOBAL VLAN ID participates on the VLAN transparently with OSA handling all the tagging work (VLAN-unaware). A host device driver can register a Global VLAN ID with the OSA adapters. Typically each host defines only one Global VLAN ID per connection. Some device drivers allow configuration of one VLAN ID for IPv4 and a second VLAN ID for IPv6. The OSA-Express will use the Global VLAN ID to tag frames and send out Gratuitous ARP requests (ARP requests to check for duplicate IP addresses) on behalf of the host. The NIC simulation in z/VM also provides this support, which is separate from the virtual switch support. The Global VLAN ID processing for the virtual NIC is performed prior to any virtual switch port ingress processing and after virtual switch port egress processing.

One example of this is in z/VM. You can specify the VLAN keyword on a LINK configuration statement for a QDIOETHERNET link defined as a trunk port to register a Global VLAN ID. (When the QDIOETHERNET link is specified as an access port, you should not specify the VLAN keyword to register a Global VLAN ID.)

To reduce complexity and host TCP/IP configuration changes when configuring a virtual switch host connection, it is recommended that you do not configure a global VLAN ID for a host that will be connected to a trunk port. Instead, connect the host to an access port and authorize it for the desired VLAN ID. This assigns a port VLAN ID (pvid) for the access port and all VLAN operations occur within the virtual switch.

What Is a Link Aggregation Port Group (LAG)?

A Link Aggregation port group is two or more links that are grouped together to appear as a single logical link. In z/VM these links are OSA adapters that are grouped on a virtual switch and given a groupname. From the virtual switch perspective the group is treated as a single link for external connectivity. A port group on the virtual switch is synonymous with a LAG (Link Aggregation Group) defined by IEEE 802.1ax or IEEE 802.3ad.

What Is an Exclusive Port Group?

An exclusive port group is a Link Aggregation Group (LAG) of OSA adapters that is defined so that one virtual switch within a single z/VM system can be configured to use the group. Essentially the adapters are dedicated for the sole use of a single VSwitch.

What Is a Shared Port Group?

A Shared port group is a grouping (set) of OSA adapters that are defined so that one or more Global virtual switches and Global virtual switch members can share the OSA adapters that make up the port group that forms a Multi-VSwitch LAG configuration. This grouping provides a single point of control for OSA port management across multiple Global virtual switches that share the same physical LAG Configuration.

What Is a Multi-VSwitch LAG Configuration?

Multi-VSwitch LAG (Link Aggregation Group) provides the ability for a port group of OSA adapters to span multiple virtual switches within or between multiple z/VM systems. This configuration allows adapters that are configured in a port group to be shared between multiple z/VM virtual switches that require access to the same LAG.

Sharing a LAG with multiple virtual switches increases utilization of the adapter by allowing it to handle larger traffic loads, especially in configurations that fully utilize the adapter.

What Is a LACP?

LACP (Link Aggregation Control Protocol) is a protocol that supports the exchange of link aggregation information between two switches of the said aggregation. This exchange of Link Aggregation control information provides for the mutual confirmation of the link aggregation configuration and the detection of a link failure within the link aggregation group (LAG) by both switches. LACP is defined in IEEE 802.1ax or IEEE 802.3ad.

What Is a LAG Port Controller?

A LAG Port Controller is a VSwitch operational role of a specific port within an Uplink Port defined in a LACP group by a VSwitch SET PORT GROUP Command. Each port within the port group has the ability to establish and manage a common LACP protocol with its partner switch. This LACP management function is known internally as a LAG Port Controller. Every VSwitch network connection sharing the same physical LAG is capable of being a LAG Port Controller. Although there is only one active LAG Port Controller per physical port within the LAG.

Note: This controller is not to be confused with a VSwitch Controller which is a completely separate and distinct type of controller.

What is an UPLINK Port?

The UPLINK port is a special port that is used to connect the virtual switch to a physical switch. The UPLINK port is a bridge from the virtual switch's simulated network to a physical network. Typically, the UPLINK port is an OSA adapter or a group of such adapters. Alternatively, the UPLINK port can be a virtual switch guest port that is set with the UPLINK NIC option on the SET VSWITCH command.

What Is HiperSockets Bridge Port?

A HiperSockets Bridge Port is a special purpose port that is used to connect the virtual switch to a HiperSockets channel. The connection bridges the virtual switch's simulated network to a HiperSockets network. If the virtual switch also has an active OSA UPLINK port, the HiperSockets channel will also be bridged to the external physical network. Such a bridge provides external network connectivity for the HiperSockets Bridge Capable ports.

What Is QEBSM?

QDIO Enhanced Buffer State Management Facility (QEBSM) is an extension to QDIO architecture. This facility provides interpretive execution by machine firmware to perform QDIO data transfers in a virtual machine. Prior to this new facility, z/VM had to mediate QDIO data transmissions between a virtual machine and a network adapter. With QEBSM, firmware and not z/VM, is involved with a typical QDIO data transfer when the guest operating system or device driver supports the facility. QIOASSIST must be turned on in the z/VM hypervisor for firmware to perform the interpretive execution for the virtual machine.

What Is a HiperSockets Guest Port?

A HiperSockets guest port is real networking connection using real HiperSockets devices (IQD) attached to a virtual machine. The network connection established on a HiperSockets guest port might not be bridge capable.

What Is a HiperSockets Bridge Capable Port?

A Bridge Capable port is a network connection that can be bridged by a virtual switch using its HiperSockets bridge port. An established network connection on a HiperSockets Bridge Capable port is able to communicate directly with a simulated network connection on the virtual switch or an external network connections through the virtual switch Uplink port. Only network connections established using QEBSM will make a HiperSockets guest port into a HiperSockets Bridge Capable port. Network connections not established using QEBSM will not be able to communicate (bridged) with the virtual switch. Bridge Capable ports can only exist on a IQD CHPID that has been defined with channel parameter EXTERNAL_BRIDGED.

What Is PMTUD?

Path MTU Discovery (PMTUD) is a standardized Internet Engineering Task Force (IETF) technique for determining an acceptable Maximum Transmission Unit (MTU) between two IP network connections. The goal of this technique is to discover the largest size datagram that does not require fragmentation anywhere along the path between the source and destination. This discovered datagram size is known as the Path Maximum Transmission Unit (PMTU).

What Is the Difference Between a Port Based and User Based Virtual Switch?

Operational differences between USERBASED and PORTBASED VSwitch have been eliminated. A system administrator has the option to either manage the VSwitch by user, by a specific port number or a combination of the two methods. This means, for example, a user interface can be configured with NICDEF PORTNUMBER, even if the device will be coupled to a USERBASED VSWITCH. Furthermore, a user port can be configured with a VSWITCH GRANT operation, even if the device will be coupled to a PORTBASED VSWITCH. This makes it convenient for the network administrator to manage a given VSwitch by user or port without immediately changing all of the GRANT commands to PORTNUMBER syntax (or to NICDEF options).

Although a VSwitch can be managed by user or port, declaring a VSwitch as either PORTBASED or USERBASED managed affects how live guest relocation is performed for a NIC connected to the VSwitch. Essentially, USERBASED or PORTBASED determines the NIC characteristics that will be checked and preserved across a Live Guest Relocation. See Chapter 6, “Live Guest Relocation Networking Considerations,” on page 89

The following describes the difference between user and port style management:

Table 2. User and Port Style Management Differences

Method	Description
User Oriented Management	Configuring NIC connectivity on the VSwitch is done on a user ID basis using the SET VSWITCH GRANT and REVOKE commands. If a guest has multiple connections to the virtual switch, all connections have the exact same attributes (port type, promiscuous, VLAN id, and so-on). Port numbers for all NICs will be assigned by z/VM.
Port Oriented Management	Configuring NIC connectivity on the VSwitch is done on a port basis. Each port is defined and configured with the SET VSWITCH PORTNUMBER command or NICDEF directory statement (with or without a portnumber). Connectivity to a specific port number can be specified on the COUPLE command. A guest can have multiple ports with distinct configurations connected to the same virtual switch.

What is Directory Network Authorization (DNA)?

With DNA, a system administrator can configure and consolidate a virtual NIC device and its network properties in a secure, centralized location in z/VM's User Directory.

The following NICDEF operands support DNA:

- PORTNUMBER *portnum*
- PORTTYPE ACCESS|TRUNK
- VLAN *vidset*
- PQUPLINKTX *priority*
- PROMISCUOUS | NOPROMISCUOUS

When a network configuration is added to the NICDEF statement, the MODIFY VSWITCH statement (SYSTEM CONFIG) and CP SET VSWITCH command can be eliminated. In this case, DNA provides the grant authorization methods previously provided by these commands. The network administrator can manage each user connection entirely within the user directory.

By default, DNA is enabled for use on z/VM but can be turned off by the system administrator with a SET VMLAN DNA CP command dynamically or during IPL with the VMLAN DNA System CONFIG statement. When DNA is disabled, all DNA operands will be ignored on the NICDEF statement. All connectivity and authorization must be performed using SET VSWITCH CP command or MODIFY VSWITCH System CONFIG statement.

When an ESM is used to manage a VSwitch, the ESM is the ultimate authority to grant access and authorize NIC characteristics for a VSwitch connection. An ESM supporting DNA will find sufficient information in the directory to fully configure network access via its own native mechanism, but all authorization for these resources are controlled by the ESM and not the VSwitch's Access Control List. The current setting of VMLAN DNA will not prevent an ESM from retrieving or using this information.

It is important to note that:

1. If NICDEF LAN is configured (even without any of these new options) CP will accept this as permission to connect to the designated network. In a simple network configuration (VLAN UNAWARE) it is no longer necessary to add NICDEF LAN operands in USER DIRECT and also add a MODIFY VSWITCH statement in SYSTEM CONFIG. In a more complex network configuration (VLAN AWARE) the MODIFY VSWITCH statements (or SET VSWITCH commands) should only be eliminated when network attributes are fully encoded on the NICDEF statement (and USER DIRECT is placed online).
2. If PORTNUMBER, PORTTYPE, VLAN, PQUPLINKTX, or a PROMISCUOUS option is found, the NICDEF LAN option is required, and CP will only allow connections to the designated network. Any attempt to couple to a different network will fail.

3. If PORTTYPE, VLAN, or a PROMISCUOUS option is found, the NICDEF is assumed to contain the complete configuration for this device. This NICDEF configuration will override any prior VSWITCH configuration for the user or port, so the administrator should fully configure the network here.
4. If NICDEF LAN option is specified with no other DNA operands and the user was authorized with a SET VSWITCH GRANT, the network characteristics specified by the previous GRANT will be used. Otherwise, the VSwitch default network characteristics will be used.

What Is a IVL Network?

The IVL (Inter-VSwitch Link) is a virtual switch to virtual switch Ethernet LAN segment providing a communications control and data plane between multiple z/VM systems. An IVL Network is required in order to exploit z/VM functions like global virtual switch and shared port groups.

What Is a IVL Domain?

The IVL Domain is a grouping of up to 16 z/VM images connected by an IVL LAN segment. Currently z/VM supports the definition of 8 unique domains per VLAN identified by a letter from A to H. A z/VM system can only belong to one IVL domain.

What Is a IVL Virtual Switch?

The IVL virtual switch provides the communication infrastructure to exchange control information and data necessary to manage Global virtual switches and shared port groups. A single IVL virtual switch may be created per a z/VM system providing access to an IVL Domain.

What Is Virtual Switch Priority Queuing?

Virtual Switch Priority Queuing is the ability of the VSwitch to assign different priorities (low, normal, high) to outbound guest traffic on different virtual NICs. The system reserves the highest priority for its own use. This support uses functions within the OSA adapters to ensure that higher priority virtual NICs don't starve the lower priority NICs.

Systems Network Architecture Terminology

The following SNA terms are briefly described in this section. See the *Systems Network Architecture Transaction Programmer's Reference Manual for LU Type 6.2* for detailed descriptions of SNA terms and concepts.

- SNA network
- Logical unit
- Session
- Transaction program
- Conversation
- Mode name
- Session limit
- Contention
- Session security
- Conversation security
- Negotiation

What Is an SNA Network?

A network is a group of two or more interconnected computing units that lets information be electronically sent from one computing unit to another. The information sent can range in size from a one-line transaction to a book-size online document. A SNA network:

- Enables the transfer of data between end users (typically, terminal operators and application programs), and
- Provides protocols for controlling the resources of any specific network configuration.

What Is a Logical Unit?

The SNA network consists of physical processors, called nodes, which are connected by physical data links. The SNA network also consists of logical processors, called logical units (LUs). An LU lets a user gain access to network resources (such as programs) and communicate with other users. LUs provide protocols that let users communicate with each other.

LU 6.2 is a particular type of SNA logical unit. LU 6.2 provides a connection between its users and network resources (often called transaction programs). The protocol that LU 6.2 provides is called Advanced Program-to-Program Communications (APPC).

What Is a Session?

In SNA, a session is a logical connection between LUs. Sessions can be compared to phone lines through which data flows between the LUs. An LU 6.2 logical unit can concurrently support more than one session with the same partner LU. Such sessions are called parallel sessions. An LU 6.2 logical unit can also support sessions with multiple LUs.

What Is Session Security?

When a session is activated, the local and remote LU may provide session security information to verify their identity to each other. This process is called session level LU-LU verification. To verify their identities, the local and remote LUs define an LU-LU password. An LU-LU password must be defined for each pair of LUs. For more information about session security and LU-LU verification, see Appendix E, “Secure APPC Communications,” on page 335 and the *VTAM Network Implementation Guide*.

What Is a Transaction Program?

A transaction program is an application program that helps users access resources in a network. Transaction programs are written using the APPC/VM or CPI Communications programming interfaces. A transaction program uses the services provided by the LU to communicate with other transaction programs by issuing transaction program verbs (APPC functions). A transaction program depends on program-to-program communications with another transaction program for some or all of its processing.

What Is a Conversation?

While LUs are connected by sessions, transaction programs are connected by conversations. Just as a session is a logical connection between the LUs, a conversation is a logical connection between two transaction programs. As Figure 10 on page 25 shows, the transaction programs TP A and TP B have established a conversation over the session between LU A and LU B.

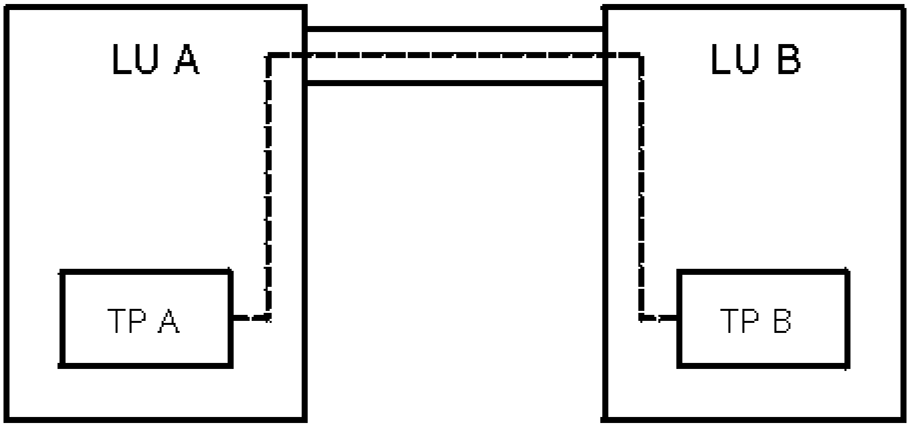


Figure 10. A Session and a Conversation

LU 6.2 treats a session as a reusable connection between two LUs. One session can support only one conversation at a time, but one session can support many conversations in sequence. Because sessions are reused by multiple conversations, a session is a long-lived connection compared to a conversation.

To establish a conversation, a transaction program specifies the following information:

- Name of the remote LU
- Name of the transaction program
- Session mode name
- Security parameters

The LU name specified by the transaction program is a network addressable unit that routes the connection within the SNA network. The session mode name identifies certain session characteristics.

If a session with the characteristics of the specified mode name exists between the local and remote LUs and the session is not being used for another conversation, the LUs assign that session to the new conversation. If a session is not available but the specified mode name is valid, the LUs start a new session using the specified mode name. This new session is then used for the conversation.

After the conversation is initiated, the two transaction programs use LU 6.2 verb functions to send and receive data as necessary to accomplish the transaction. When the transaction is finished, one transaction program ends the conversation. That session is now available for another conversation between transaction programs using the session's LUs as entry points into the SNA network.

What Is a Mode Name?

A mode name defines the characteristics of a session. These characteristics include pacing levels (for example, high speed, batch, and interactive) and class of service (for example, secure, ASCII data, satellite communication). The system administrator assigns the mode name for each session and LU 6.2 associates each session with a mode name.

When transaction programs request a conversation, they cannot specify which session to use for the conversation. However, by specifying a mode name, the transaction programs can specify the characteristics of the session.

A pair of LUs can support multiple sessions that have the same mode name and multiple mode names can be defined between LUs. These sessions form a session group, which is treated as a pool of sessions sharing the mode name characteristics. The system administrator can control the size of the session group, but the LU is responsible for the individual sessions within the group.

An LU can define several session groups for sessions with another LU. For example, LU A may interact with LU B to process requests by file-server programs and for database queries. LU A may define the mode name FILESERV for sessions used by the file-server programs; it can also define the mode name INTERACT for sessions used for database queries. In this example, FILESERV could denote sessions with a large request size, which would aid bulk transmission of data.

What Is a Session Limit?

LU 6.2 imposes limits on the number of total sessions that may be established between a pair of parallel-session capable LUs. These limits are called session limits.

Session limits can be specified for each mode name; this enables LU pairs to have multiple session limits. For example, LU A may be limited to five sessions with LU B when the mode name FILESERV is specified. However, LU A may be able to establish 10 sessions to LU B when the mode name INTERACT is specified on a connection request.

Session limits can be dynamically defined while the programs are executing. LU 6.2 defines Change Number of Session (CNOS) verbs that can change session limits. The session limit defined by the CNOS verbs limits the number of conversations that can be active. If there are 10 sessions, only 10 concurrent conversations can be active.

What Is Contention?

Two LUs may attempt to allocate a conversation over the same session at the same time. This situation is called contention. It is resolved by designating one of the LUs the contention winner for the use of the session. The other LU is the contention loser.

A contention winner can allocate the session without informing the contention loser. A contention loser can request use of the session from the contention winner. For LU 6.2 programs, the LU is responsible for requesting use of sessions on the contention loser side and granting or denying it on the contention winner side. Programs need not be concerned with this process.

Generally, parallel session-capable LUs can divide the contention winner role between them for their sessions. The number of winners is divided based on which LU will typically start a conversation. For example, LU A and LU B can support 10 parallel sessions. If each LU expects to start the same number of conversations, LU A may be designated the contention winner for five sessions and LU B may be the contention winner for five sessions.

The number of contention winners can be defined at system definition time or dynamically. A CNOS verb sets the contention winner and contention loser values. However, in some case, you may not want to set the contention loser and winner values equal.

For example, a workstation is attached to a z/VM system. Because workstation programs would frequently start conversations with the z/VM system, the workstation would define many contention winners and few contention losers. Programs on the z/VM system would be less likely to start a connection to programs on the workstation.

What Is Conversation Security?

For each conversation, conversation security information is sent to the target LU and the target transaction program. The target LU receives an access security field that indicates:

- The type of security being used
- Access security user ID, which identifies the user program that requested the connection
- Access security password

The target LU verifies the user ID and password. The LU then uses the verified user ID to determine if the user program can connect to the target transaction program. The target transaction program receives the access security type and the access security user ID.

The extent to which a target transaction program uses the access security user ID can vary. Some transaction programs, like public bulletin boards, do not use the access security user ID. Any transaction program can access the information on the bulletin board. Other transaction programs, like database managers, can use the access security user ID to determine which data can be accessed by the user program.

When a user program makes an allocation request, it must specify the type of access security information it is sending to the target LU and target transaction program. APPC defines the following types of access security levels:

- SECURITY(PGM)
- SECURITY(SAME)
- SECURITY(NONE)

SECURITY(PGM)

When a user program specifies SECURITY(PGM), it must send an access security user ID and password in its connection request. The target LU verifies this information and determines if the user program is authorized to connect to the target transaction program. The target transaction program also receives the access security user ID, which it uses to determine if the user program can access its resources.

SECURITY(SAME)

When a user transaction program specifies SECURITY(SAME), it does not send an access security user ID or password in its connection request. The logon user ID that was used to start the user transaction program is sent to the target LU and target transaction program; the target LU is also notified that the user ID has already been verified. If the target LU does not support SECURITY(SAME), it acts as if SECURITY(NONE) had been specified on the allocation request.

SECURITY(SAME) should be used when a transaction program acts as an intermediate server to request services for another transaction program. For example, Program A specifies SECURITY(PGM) to allocate a connection request to Program B. To complete Program A's request, however, Program B must call another transaction program, Program C. Program B, which is an intermediate server, specifies SECURITY(SAME) and provides Program A's user ID when it requests access to Program C. Program B also notifies the target LU that this user ID has already been verified. Program C does not receive Program B's access security user ID. There are special session security concerns when using SECURITY(SAME). See Appendix E, “Secure APPC Communications,” on page 335 for details of these concerns.

SECURITY(NONE)

When a transaction program specifies SECURITY(NONE), it does not send an access security user ID or password in its allocation request. The target LU cannot determine if the source transaction program is authorized to connect to the target transaction program. The target transaction program cannot determine which transaction program is requesting a connection. However, in this case, the target transaction program may be written to accept connection requests from any other transaction program.

For example, a public bulletin board, which can be accessed by any user, does not need to know which users access it. A workstation program can specify SECURITY(NONE) to access the bulletin board.

What Is Negotiation?

To coordinate activities in an SNA network, the LU 6.2 protocol allows partner LUs to negotiate session values specified on CNOS verbs. The partner LUs can also negotiate the contention winner and contention loser values and the access security levels. The transaction programs that allocate conversations over the sessions cannot directly negotiate these values.

For example, LU A issues a CNOS verb that indicates it wants a session limit of 100 sessions between itself and LU B. If LU B only wants to have 50 sessions with LU A, it can negotiate with LU A to decrease the number of sessions.

z/VM Terminology

This section introduces the following z/VM connectivity terms:

- *IDENT (Identify System Service)
- TSAF collection

- CS collection
- Domain and domain controller
- VM resources (local, global, system, and private)
- Communications partners (resource managers and user programs)
- AVS gateways (global and private gateway)
- System gateway.

What Is *IDENT?

The Identify System Service (*IDENT) is a CP system service that lets authorized virtual machines identify themselves as resource or gateway managers. *IDENT also lets authorized virtual machines revoke ownership of resources and gateways.

A TSAF or CS collection can be thought of as an LU that has the LU name *IDENT. Transaction programs identify themselves as resources, which allows user programs to allocate conversations to them.

What Is a TSAF Collection?

A TSAF collection is a group of up to eight interconnected z/VM systems each of which has a TSAF virtual machine installed and running. A TSAF collection has the following properties:

- Automatic formation

  Systems can dynamically join and leave the collection.
- Automatic resource identification

  VM resources (transaction programs) can dynamically identify themselves within the TSAF collection without manual intervention.
- Transparent access to VM resources

  VM resources can be accessed by APPC/VM programs within the collection without regard to the resource's location.
- Single name space for global resources and user IDs

  Global resources are known throughout the collection and their names are unique within a collection. User IDs uniquely identify a particular user.

For programs in a TSAF collection to communicate, a logical connection must be established between the programs. Within a single VM system, CP provides an APPC/VM path that logically connects two programs. Within a TSAF collection, CP provides an APPC/VM path that connects each program with the TSAF virtual machine on its system. The TSAF virtual machines provide a logical APPC/VM path (a communications link) between the two systems, which lets the programs communicate.

What Is a CS Collection?

A Communication Services (CS) collection is a group of interconnected domains consisting of VM systems that use ISFC to communicate with other z/VM systems. A CS collection has the following characteristics:

- Automatic formation

  Systems can dynamically join and leave the collection.
- Automatic resource identification

  Resources can dynamically identify themselves within the CS collection without manual intervention.
- Transparent access to resources

  On VM systems, the transaction programs can be written using CPI Communications routines or APPC/VM.
- Single name space for global resources and user IDs

Global resources are known throughout the CS collection and their names are unique within the collection. A user ID uniquely identifies each user.

What Is a Domain?

A domain consists of a domain controller and users. A group of interconnected domains form a CS collection. In a z/VM system running ISFC, CP acts as the domain controller for all of the users that are defined in the directory of that system and authorized to use APPC/VM communications. See Part 3, “APPC Server and Requester Virtual Machines,” on page 163 for details on authorizing users to be requesters and servers.

Transaction programs can use ISFC to access and manage resources. Transaction programs can reside in virtual machines on a VM system. Users of these transaction programs must sign-on to a domain to be able to access or manage resources. This sign-on ensures a user is authorized to use resources in the CS collection.

What Is a VM Resource?

A VM resource is a program, a data file, a specific set of files, a device, or any other entity or set of entities that you might want to identify for use in application program processing. A VM resource is identified by a VM resource name. A VM resource maps to a transaction program, and a resource name maps to a transaction program name.

One program can be represented by one or more resource names. For example, a database program that manages two databases, DataB1 and DataB2, could be known by the resource name DataB1 for requests to access database DataB1. However, the same program could be known by the resource name DataB2 for requests to database DataB2. Resources are managed by resource managers that run in server virtual machines (see “What Are Communications Partners?” on page 31).

A resource can be located on the local system or on any other system within the TSAF or CS collection. You can define four types of resources in a TSAF or CS collection:

- Local
- Global
- System
- Private

The following sections briefly describe the features of the four different types of resources. See “Resource Managers” on page 166 for a detailed description of the different resource types.

What Are Local and Global Resources?

A local resource is known only to the system on which it resides. Only authorized users on the local system can access local resources. The name of a local resource must be unique only within the local system. Resources (for example, a printer) that should be limited to the users of one system should be defined as a local resource to that system.

For a local resource, the VM system on which the resource resides can be thought of as the LU for an allocation request. The LU name is *IDENT.

A global resource is known to all systems in the TSAF or CS collection. Authorized users in the TSAF or CS collection or the SNA network can access global resources. Each global resource name must be unique within the TSAF or CS collection in which it resides. Resources (for example, databases) that contain dynamic information needed by users in the TSAF or CS collection should be defined as global.

Global and local server virtual machines are explicitly logged on and the resource managers are explicitly called. Therefore, the resource managers are always ready for requests.

If a local and global resource are defined with the same name, the resources are accessed as follows:

- When a user on the local z/VM system requests to communicate with the resource, CP routes the user to the local resource. If the resource is unavailable or is not known on the local system, TSAF or ISFC routes the request to the global resource.
- When a remote user on another z/VM system in the TSAF or CS collection requests to communicate with the resource, TSAF or ISFC routes the user to the global resource, even if a local resource with the same name also exists on the target system.

For example, a TSAF collection contains two printers. Printer A is a local resource on System A; Printer B is a global resource on another system in the TSAF collection. Each printer is identified by the resource name PRINTER. When a user on System A requests to communicate with the resource PRINTER, the request is routed to the local printer, Printer A. However, if Printer A is not available, the local user will be routed to the global resource PRINTER, Printer B.

For global resources, the TSAF and CS collection in which the resource resides can be thought of as the LU. In this case, the LU name is *IDENT.

What Are System Resources?

A system resource is known only to the z/VM system where it is located but it is remotely accessible from other systems. A system resource name only needs to be unique to that system. Any authorized user in the TSAF or CS collection can access the system resource. Authorized users from the SNA network can also access system resources. In this case, the system resource is accessible using a global gateway that is defined on the VM system on which the system resource resides; AVS must also be running on that VM system.

From a requester on the same VM system, system resources are accessed the same as a local or global resource on that system. From a TSAF or CS collection, system resources are accessible using the system gateway of the system on which the system resource resides. In this case, the target LU is the system gateway name of the system on which the system resource resides. The transaction program name is the name of the system resource. See “What Is a System Gateway?” on page 33 for more information. See Table 8 on page 170 for identifying the target of a connection request.

The server virtual machine must be authorized to manage a system resource. This authorization is the same that is required for a global resource manager.

Like local and global resource server virtual machines, the system resource server virtual machine needs to be logged on. The resource manager needs to be started before requests for a connection to that resource can be completed successfully.

For system resources, the VM system on which the resource resides is the LU. Within a TSAF or CS collection, the target LU name for the connection request is the system gateway name of that system.

What Are Private Resources?

A private resource is known only to the virtual machine in which it is located. It is not identified to CP and it is not explicitly known throughout the TSAF or CS collection. Each private resource name within the TSAF or CS collection only needs to be unique within the virtual machine in which it resides. Any authorized users in the TSAF or CS collection or the SNA network can access private resources. The private resource server virtual machine uses a special NAMES file, which lists private resources and the authorized users for each resource.

Resources (for example, a plotter) that are not frequently used should be defined as private. Resources that should be limited to a single user or a specific group of users (for example, a department) should also be defined as private. For example, a user working on a workstation uses a program to access files in their virtual machine. These files would be defined as private resources and the workstation user would be the only authorized user of the resources.

If a system administrator wants to control access to a resource rather than having the resource manager program control access, the resource should be defined as private. The system administrator can create a protected batch-like environment using private resources. The administrator can authorize the server virtual machine to issue privileged instructions. The administrator can also identify which programs run in the server virtual machine and limit which users can run certain programs.

The private resource server virtual machine does not need to be logged on and the resource manager does not need to be started when a program requests a connection. If the private resource server virtual machine is not logged on and its CP directory entry contains an IPL statement, CP will automatically log it on and start the private resource manager. The virtual machine in which the private resource resides is the LU for that resource.

In a TSAF or CS collection, private resource support has been enhanced with the system gateway. Using the system gateway in these environments, the same private server may be defined on all systems in a TSAF or CS collection. The system gateway may be used to access the desired private server. In this case, the system gateway name is the LU name qualifier and the user ID of the private server is the target LU name.

Resource Naming Considerations

When you define resource names, you should consider the following information:

- A VM system can have local and global resources with the same name; it can also have local and system resources with the same name. When a user on the local system specifies this resource name, the connection will be routed to the local resource. When a remote user specifies the resource name, however, the connection request is routed to the global or system resource.

  For example, a local and global resource are defined on System A; each resource has the name RESOURC1. When a user transaction program on System A requests a connection to RESOURC1, the request is routed to the local resource. If a user transaction program from a remote system requests access to RESOURC1, the request is routed to the global resource on System A.
- Global resource names must be unique within a TSAF or CS collection.
- A system resource name must be unique in the system on which it resides. Also, a system resource cannot have the same name as a global resource that resides on that system. However, the same system resource name can be defined on other systems in the TSAF or CS collection.

  For example, the system resource SYSRES1 can be defined on System A and System B in the TSAF collection. However, System A cannot define a global resource or another system resource with the name SYSRES1.

  The system resource can be accessed by specifying the system gateway of the system in which the resource resides. See "What Is a System Gateway?" on page 33 for more information.
- In a TSAF collection, a global resource defined on one system may have the same name as system resources defined on other systems. For example, the global resource BBOARD1 is defined on System A; System B and System C may also each define a system resource by the name BBOARD1.

  However, in a CS collection, a system or global resource name can be defined on only one system. For example, if the global resource BBOARD2 is defined on System A, System B cannot define a system resource with the name BBOARD2.
- One system can identify a maximum of 500 global resources and gateways. A maximum of 65,535 local resources can be identified on one system. The total number of global resources and gateways and system resources defined on one system cannot exceed 65,535.

  For System/370 systems, the total number of global resources and gateways, local resources and system resources that can be identified on one system is 200.

What Are Communications Partners?

Communications partners are transaction programs that communicate when one program requests the services of the other transaction program. The transaction program that requests services is called a requester or user program. The requester program generally requests access to a specific resource. The transaction program that provides this service or manages the resource is called the server or resource manager program.

Communication partners, which are written using the APPC/VM or CPI Communications programming interfaces, may be located on the same VM system, on different systems in a TSAF or CS collection, or in the SNA network.

What Is a Resource Manager?

A resource manager is a transaction program that manages access to one or more VM resources. A resource manager receives requests from requester programs to access resources it owns. The following are examples of resource managers:

- Database managers
- File servers that manage sets of files
- A virtual machine that manages a high-function printer

A resource manager program runs in a server virtual machine. The characteristics of this virtual machine can vary, depending on the type of resource (local, global, private, or system) being managed. For example, you may add directory statements to the server virtual machine's CP directory or create a special NAMES file for that virtual machine. See "Managing Local, Global, and System Resources" on page 174 and page "Managing Private Resources" on page 174 for more information about requirements for server virtual machines.

What Is a User Program?

A user program is a program that must communicate with a resource manager for some or all of its processing. A user program starts a conversation with a resource manager to request a connection to a resource.

Like a requester program, a user program also runs in a virtual machine, which is called a requester virtual machine. See "Preparing Virtual Machines to Request Connections to Resources" on page 177 for more information about requester virtual machines.

What Is an AVS Gateway?

An AVS gateway is an LU that has been defined on a z/VM system to represent the TSAF or CS collection to VTAM. (An AVS gateway is also called a gateway LU.) Programs in the SNA network view a TSAF or CS collection as one or more LUs.

An AVS gateway acts as a communication server. When you define an AVS gateway, APPC programs in the SNA network can use the gateway to access resources within the TSAF or CS collection. This type of request is called an inbound request. Programs within the TSAF or CS collection can also use AVS gateways to access resources in the SNA network. This type of request is called an outbound request.

The AVS virtual machine manages AVS gateways. The CP directory entry of the AVS virtual machine contains IUCV statements that enable it to manage specific gateways. Gateways are dynamically added or deleted when you enter the AGW ACTIVATE GATEWAY and AGW DEACTIVE GATEWAY commands. ISFC and the TSAF virtual machine keep lists of all gateways currently in the collection. Each gateway name must be unique within the TSAF or CS collection where the AVS virtual machine resides. When communicating with the SNA network, communication partners specify the name of the AVS gateway to identify the location of the target resource.

On a connection request from the SNA network, the name of the AVS gateway is specified as the target LU name.

You can define two types of AVS gateways in the TSAF or CS collection: global gateways and private gateways. The type of AVS gateway that is used for a connection request identifies the type of resource that is being accessed. Transaction programs use global gateways to access global or system resources; private gateways are used to access private resources.

What Is an AVS Global Gateway?

APPC programs in the SNA network use an AVS global gateway to access a global or system resource that resides in the TSAF or CS collection. The global resource may reside on any system within the TSAF or CS collection. However, the system resource must reside on the same system on which the global gateway has been defined. Programs within the TSAF or CS collection can also use a global gateway to access APPC programs in the SNA network.

An SFS file pool is an example of a resource that could be identified as a global resource in a TSAF or CS collection. Users in a VM system located in the SNA network would access this file pool through a global gateway defined in the TSAF or CS collection.

What Is an AVS Private Gateway?

APPC programs in the SNA network use AVS private gateways to connect to private resources located in the TSAF or CS collection. When a private gateway is activated, it can be defined as either dedicated or non-dedicated.

Private gateways can be dedicated to a single user ID. When a private gateway is dedicated to a user ID, all connection requests routed through that gateway are sent directly that user ID. The connection request is not checked to determine which user ID is the target of the request. A dedicated private gateway should be defined for private resource server virtual machines that receive requests from multiple users.

A non-dedicated private gateway can be used to connect to multiple user IDs; it is not dedicated to a specific user ID. Connection requests to the private server virtual machine are routed through a non-dedicated gateway; these requests are checked to determine the user ID to which the connection request needs to be routed. One non-dedicated private gateway could be used by all users who are requesting access to their own virtual machine.

Private gateways may be associated with a Conversation Management Routine (CMR). A CMR is a service pool manager that routes incoming connections from the SNA network to an available service pool virtual machine. See Appendix C, “Introduction to Service Pool Support,” on page 325 for more information.

What Is a System Gateway?

A system gateway provides access to global, private, or system resources on a specific VM system within a TSAF or CS collection. Unlike global gateways and private gateways, which are defined to AVS, a system gateway is defined by the z/VM system when CP is IPLed.

For example, Figure 11 on page 34 shows two systems in a TSAF collection both having a system resource X identified. A user on system VM1 can access system resource X on system VM2 using the system gateway of the target system (VM2).

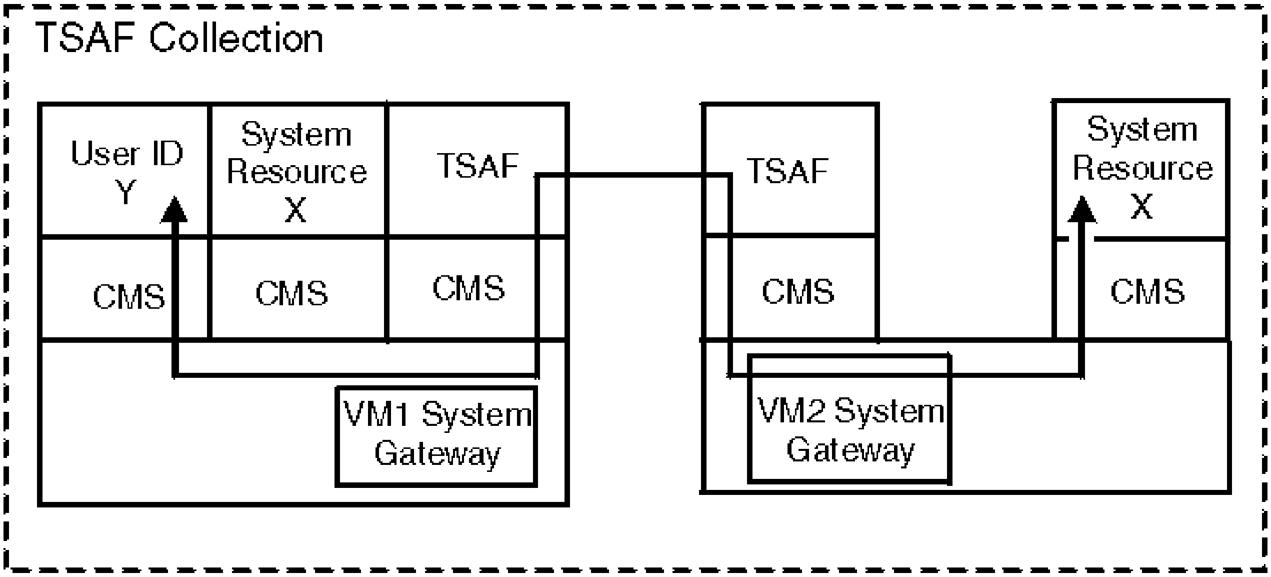


Figure 11. Using a System Gateway to Access System Resources

Similarly, Figure 12 on page 34 shows two systems in a TSAF collection, both with the user ID Z defined, where Z is a private resource manager. User Y on system VM1 can access private resource manager Z on VM2 by specifying the system gateway of the target system (VM2).

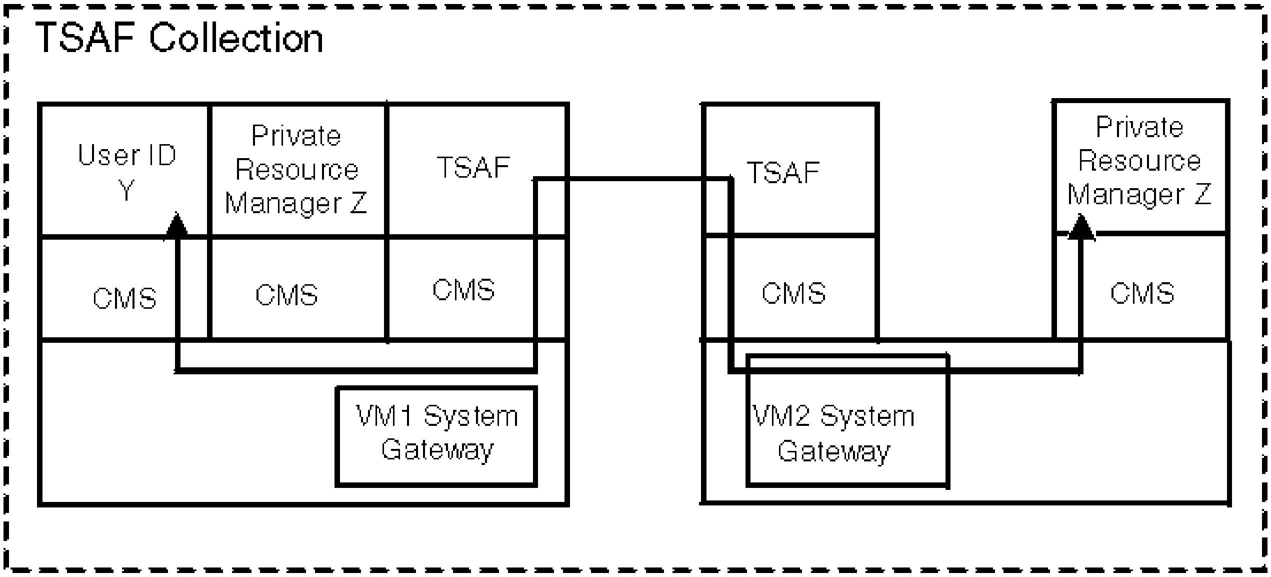


Figure 12. Using a System Gateway to Access a Private Resource Manager

The system gateway also provides access to private and global resources in a CS collection from an adjacent TSAF collection. Similarly, private and global resources in a TSAF collection can be accessed from an adjacent CS collection, as Figure 13 on page 35 shows.

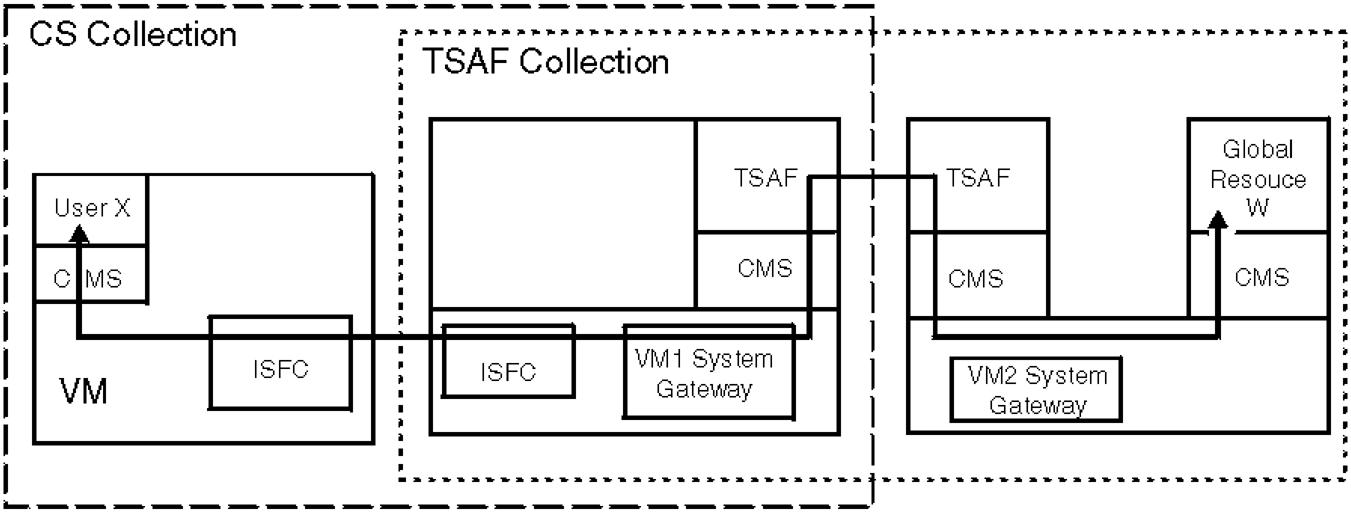


Figure 13. Using a System Gateway to Access Resources in an Adjacent Collection

The system gateway name for a system is identified automatically when CP is initialized on the system. The system gateway name is specified on the SYSTEM_IDENTIFIER Statement or SYSTEM_IDENTIFIER_DEFAULT Statement in the system configuration file. See *z/VM: CP Planning and Administration* for more information about specifying the system gateway name.

Part 2. Planning Virtual Networks

This part provides a comprehensive overview of networking options and virtual network management within the z/VM environment. It details real hardware connectivity through OSA-Express and Network Express adapters, as well as virtual networking via guest LANs and virtual switches, including their configuration, advantages, and operational considerations. The text covers advanced topics such as link aggregation (LAG) with IEEE 802.3ad support, global virtual switches that span multiple z/VM systems by using Inter-VSwitch Links (IVL), VLAN management, and high availability configurations. It also addresses troubleshooting methods like promiscuous mode and packet tracing, security controls for virtual networks, and specific features related to HiperSockets bridging and live guest relocation. Throughout, distinctions between QDIO and EQDIO protocols, adapter types, and their capabilities are clarified. Detailed guidance for planning, deploying, and managing efficient, scalable, and secure virtual networks in z/VM environments is provided.

Related Information: See the following sources for more information on the tasks in this section:

Task	Source
Planning z/VM with guest LANs and virtual switches	*z/VM: CP Planning and Administration*
Using CP commands with guest LANs and virtual switches	*z/VM: CP Commands and Utilities Reference*
Configuring TCP/IP and a virtual switch Controller	*z/VM: TCP/IP Planning and Customization*
Using TCP/IP	*z/VM: TCP/IP User's Guide*
OSA-Express	Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf)
HiperSockets	System z Input/Output Configuration Program User's Guide for ICP IOCP (publibz.boulder.ibm.com/epubs/pdf/b107037b.pdf)

Chapter 3. Networking Options in z/VM

The topics in this chapter detail the networking options that are available in z/VM environments. The information explains two main types of network connectivity for guests: real hardware interfaces (such as OSA-Express, Network Express, HiperSockets, and Channel-to-Channel adapters) and virtual networking options (including guest LANs and virtual switches).

Real hardware options provide high bandwidth and direct external LAN access, with distinctions between IP (Layer 3) and Ethernet (Layer 2) modes. Virtual networks offer flexible, centralized management without physical connections that support both TCP/IP and non-TCP/IP protocols.

The information also covers features like QDIO/EQDIO architectures, VLAN support differences between guest LANs and virtual switches, link aggregation, port isolation, and bridging capabilities between virtual and physical networks. It highlights advantages of virtual networks such as reduced hardware costs, simplified management, and enhanced security, and provides guidance on choosing between guest LANs and virtual switches based on deployment needs.

Real Hardware Options

Real hardware network interfaces supported by the z/VM TCP/IP stack include interfaces such as OSA-Express, HiperSockets, and Channel-to-Channel support. See System Requirements for TCP/IP in *z/VM: TCP/IP Planning and Customization* for the complete list of supported network devices. Real network connections for Linux on IBM Z may be obtained through any device supported by Linux, such as OSA and Channel-to-Channel adapters.

Some hardware network interfaces such as Channel-to-Channel are point-to-point type connections. Point-to-point connections require that each end of the connection be identified by an IP address. OSA-Express and HiperSockets do not require point-to-point connections so you only need to define a single IP address for each connection, thus simplifying network configuration and management.

z/VM Network with OSA-Express Adapter

z/VM supports OSA-Express adapters. This option provides direct connectivity to the physical LAN segment for guest virtual machines by using IP or Ethernet mode:

- **IP (Layer 3) Mode**
  - Transmission of data is based on IP addresses.
  - IP addresses are used to identify hosts on the LAN segment.
  - The actual data to be transmitted is encapsulated into IP packets for transport on the LAN.
  - A single Media Access Control (MAC) address is managed by the adapter for all guests sharing the adapter (either dedicated or sharing through a guest LAN).
  - All Address Resolution Protocol (ARP) processing is handled (off-loaded) by the OSA-Express adapter.
  - The OSA-Express connection adapters are:
    - Gigabit Ethernet (GbE)
    - 10 Gigabit Ethernet
    - 25 Gigabit Ethernet
    - 1000Base-T Ethernet
- **Ethernet (Layer 2) Mode**
  - Transmission of data is based on MAC addresses.
  - MAC addresses are used to identify hosts on the LAN segment.

- The actual data to be transmitted is encapsulated into Ethernet frames for transport on the LAN.
- Each host has or is assigned a unique MAC address.
- ARP processing is managed by each host on the LAN segment.
- Supports IEEE 802.3ad Link Aggregation Configurations.
- The OSA-Express connection adapters are:
  - Gigabit Ethernet (GbE)
  - 10 Gigabit Ethernet
  - 25 Gigabit Ethernet
  - 1000Base-T Ethernet

All of these OSA-Express connections can use QDIO architecture.

For information about z/VM environments that use OSA-Express, see Chapter 5, "Planning a z/VM Environment with OSA-Express and HiperSockets," on page 83, and Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf).

z/VM Network with Network Express Adapter

z/VM supports Network Express adapters. This option provides direct connectivity to the physical LAN segment for guest virtual machines by using Ethernet mode:

- **Ethernet (Layer 2) Mode**
  - Transmission of data is based on MAC addresses.
  - MAC addresses are used to identify hosts on the LAN segment.
  - The actual data to be transmitted is encapsulated into Ethernet frames for transport on the LAN.
  - Each host has or is assigned a unique MAC address.
  - ARP processing is managed by each host on the LAN segment.
  - Supports IEEE 802.3ad Link Aggregation Configurations.
  - The Network Express connection adapters are:
    - 10 Gigabit Ethernet
    - 25 Gigabit Ethernet

All of these Network Express connections can use Enhanced Queued Direct I/O (EQDIO) architecture.

z/VM Network with HiperSockets

z/VM supports HiperSockets. This option provides IP transport connectivity between guest systems within or across logical partitions (LPARs) through an internal hardware LAN segment. HiperSockets is a microcode feature that is part of particular IBM Z hardware.

For information on planning a z/VM environment that uses HiperSockets microcode to connect guest systems, see Chapter 5, "Planning a z/VM Environment with OSA-Express and HiperSockets," on page 83.

Advantages of Using Real Network Hardware

Applications that require constant high networking bandwidth and throughput capability should generally use dedicated network hardware attachments. When dedicated networking attachments are deployed for a guest virtual machine, z/VM also dedicates system resources such as storage in support of this dedicated configuration. Use OSA adapters to connect to external hosts, or use the HiperSockets feature to connect to hosts on the same processor CEC.

Virtual Networking Options

z/VM provides a natural infrastructure for virtual network computing that enables you to participate in multisystem environments without the need for physical connections. There are two types of virtual networks:

- LAN connections with HiperSockets and OSA-Express simulation
- Point-to-point connections with virtual CTC and IUCV

Guest LAN

Guest LANs allow you to set up a virtual network without the need for point-to-point physical connections between guest systems that run on a single z/VM image or LPAR. With guest LAN, you can connect guest systems on the same host system by using virtual network adapters rather than dedicated hardware adapters. In a virtual network, you define the virtual network adapter in each z/VM guest machine and then connect each adapter to the guest LAN.

z/VM offers emulation for two kinds of network adapters:

- OSA-Express QDIO adapter
- HiperSockets iQDIO adapter

Specific hardware is not required, so you can implement guest LAN on any supported processor. However, you cannot define virtual connections between LPARs. LPAR to LPAR communication in a LAN must be configured by deploying HiperSockets or OSA-Express hardware connections.

By using z/VM's simulation of OSA-Express and HiperSockets devices, you can create a network of guest systems on a single host system without the need to define physical connections. Both OSA-Express and HiperSockets allow you to define virtual adapters and control devices for the connections to z/VM guest systems or LPARs on the processor. Guest LANs also offer a single point of management for authorization and authentication.

Guest LAN simulation of either OSA-Express or HiperSockets offers similar networking technologies — that is, unicast, broadcast, and multicast. Your decision to use one or the other may be based on real hardware availability on the processor and whether you are using guest LAN as a production networking environment or as a test vehicle for a real hardware setup.

Virtual Switch

The virtual switch is a guest LAN technology that bridges real hardware and virtual networking LANs, which offers external LAN connectivity to the guest LAN environment. The virtual switch operates with virtual QDIO adapters (OSA-Express).

LAN (Uplink port) connectivity is available through OSA adapters in QDIO mode (OSA-Express; CHPID type OSD) and EQDIO mode (Network Express; CHPID type OSH). The virtual switch supports the transport of either IP packets or Ethernet frames in QDIO mode but only Ethernet frames in EQDIO mode

By default, the virtual switch operates in IP mode. Each guest is identified by one or more IP addresses for the delivery of IP packets. Data is transported within IP packets, and therefore the virtual switch in IP mode supports only IP based application communications. All traffic destined for the physical portion of the LAN segment is encapsulated into an Ethernet frame with the OSA adapter's MAC as the source MAC address. On inbound, the OSA adapter strips the Ethernet frame and forwards the IP packet to the virtual switch for delivery to the guest by the destination IP address within the IP packet.

When operating in Ethernet mode, the virtual switch uses each guest's unique MAC address to forward frames. Data is transported and delivered within Ethernet frames, providing the ability to transport both IP and non-IP base application data through the fabric that the virtual switch supports. Through the ARP processing of each guest, the guest's MAC address becomes "known" (cached) by hosts residing on the physical side of the LAN segment. The generation and assignment of the locally defined MAC address is performed by z/VM under the direct management control of the LAN administrator. Each outbound or inbound frame through the OSA adapter switch trunk connection is an Ethernet frame with the guest's MAC address as the source or destination MAC address.

The virtual switch that is configured in Ethernet mode supports the aggregation of multiple OSA adapters for external LAN connectivity. By supporting the IEEE 802.3ad Link Aggregation protocols and mechanisms, the aggregation of individual physical links (adapters) makes this collection or group appear as one large link. The deployment of this type of configuration increases the virtual switch bandwidth and provides near seamless failover in the event that a port becomes unavailable. This support provides the ability to aggregate physical OSA adapters. The ability also exists to configure multiple virtual switches to the same LAG by sharing the OSA adapters that comprise the Link Aggregation port group. The aggregation of simulated guest NIC ports is not supported (simulated NICs are those defined with the DEFINE NIC command). For more information of z/VM VSwitch support of Link Aggregation see Chapter 8, “Virtual Switch Link Aggregation,” on page 113.

A system administrator has the option to manage a VSwitch by a user strategy, by a port strategy or by using a combination of the two methods. For user management strategy virtual switch, authorization and configuration will be on a user ID basis via the SET VSWITCH GRANT and REVOKE commands. All connections for a particular user have the same attributes (port type, promiscuous, VLAN id, etc). For port management strategy, authorization and configuration is on a port basis. Each port must be defined and configured with the SET VSWITCH PORTNUMBER command or NICDEF directory statement. Connectivity to a specific port number can be specified on the COUPLE command. A guest can have multiple unique ports connected to the same virtual switch. Each port has it own attributes.

The virtual switch coupled with the OSA adapter provides a very powerful, flexible, and robust virtualization model. Data is transferred between guest ports of the virtual switch and between sharing partitions of the same OSA adapter without having to leave "the box". For installations that have security policies that require that access to the guest ports of the virtual switch be controlled, the deployment of the virtual switch port isolation facility is required. This facility actually *isolates* all guest port communications and also isolates the virtual switch connection from all other sharing hosts/LPARs on the OSA adapter or port. An external router configured as a firewall can be deployed to control access between the virtual switch guest ports themselves and between a guest port and any hosts (LPARs) that share the same OSA port.

The virtual switch HiperSockets Bridge Port supports QDIO (OSD) type simulated LANs. Through the configuration of a HiperSockets Bridge Port on the virtual switch, this bridging is extended to the HiperSockets channel LAN (CHPID) as well. A HiperSockets Bridge Port provides a layer 2 Bridge for bridge capable ports connected to a HiperSockets LAN through the virtual switch to an external network LAN over its OSA Uplink port. This configuration places all LAN endpoints on the same flat layer 2 broadcast domain. This bridging capability allows a virtual machine with a single HiperSockets connection, connectivity to destinations that reside on the HiperSockets network, as well as simulated NIC devices coupled to the virtual switch and more importantly external destination located on the physical network.

Advantages of Using Virtual Networks

Virtual networks provide the same resource sharing capabilities that are inherent in other resources that are managed by z/VM (such as devices, storage, and so on). When high bandwidth networking throughput is not required by a guest, then it is more efficient to share networking resources than to dedicate those resources to a guest that is not fully utilizing them. Unlike physical resources, the building, populating, and management of a virtual network can be orchestrated from a single point of reference — the z/VM operating system. This includes authorizations for access to guest LANs and virtual switches and participation in IEEE 802.1Q Virtual LAN segments (VLANs).

Common techniques for networking in the z/VM environment include Inter-User Communication Vehicle (IUCV) and Channel-to-Channel Adapter (CTCA) connections. These methods require a virtual point-to-point connection between systems and devices and two IP addresses for each connection. However, a virtual network that uses z/VM's HiperSockets emulation technology and OSA adapters does not require point-to-point hardware connections and you need to define only one IP address for each connection.

Consider the following advantages when you create a virtual network by using z/VM and QDIO or Enhanced QDIO (EQDIO) technology:

- Virtual networks are easy to define. They reduce the networking hardware investment (fewer cables, hubs) and eliminate dependencies on hardware. They also simplify management and access with centralized access control.
- You can consolidate hardware by connecting guest systems that run in virtual machines in a single processor. You can eliminate separate hardware that runs these systems along with the cost, complexity, and maintenance of the networking components that are needed to connect them.
- Consolidating servers in a virtual network allows you to reduce or eliminate the overhead associated with traditional networking components.
- By defining a virtual network within a single processor, you do not need to consider network traffic outside the processor. As a result, you can achieve a high degree of network availability, security, and performance. For example, security features that are available to TCP/IP network interfaces are available to HiperSockets. Also, HiperSockets, like any other TCP/IP interface, is transparent to applications and operating systems.
- By deploying Ethernet mode guest LANs and virtual switches, you extend your virtual networking capability to include non-TCP/IP based applications. Some examples of these include SNA, IPX, and NetBIOS.
- You can configure virtual switches to use SNMP to provide Bridge MIBs in response to Network Management System (NMS) requests. You can also use SNMP traps to report events to a NMS. Those events include changes to the status of a guest's connection to a virtual switch and changes to the status of the virtual switch's connection to the external network.

Note: Device definitions in the guest networking drivers (for example, TCP/IP stack) are the same for both real and virtual devices. This allows the testing of an environment with virtual devices with easy crossover to real hardware devices.

Guest LAN versus Virtual Switch

Depending on your requirements, you may define guest LANs, virtual switches, or a combination for your system. A guest LAN is a closed LAN where each virtual machine can communicate only with the other virtual machines in the same guest LAN. To connect a guest LAN to an external network, you must do one of the following:

- Use a router virtual machine that has connectivity to both the guest LAN (via a virtual network adapter) and a real hardware network device.
- Deploy a virtual switch.

A virtual switch is a special-purpose guest LAN that can be defined with connections to one or more real OSA adapters as its Uplink port. You can also define a virtual switch without the OSA adapter, in which case the virtual switch acts like a guest LAN. A third choice is to define a virtual switch with the Uplink port directed to a guest port on the virtual switch. With a virtual switch, the requirement for a router virtual machine is eliminated.

VLAN support differs dramatically when deployed on a virtual switch as opposed to on a guest LAN:

- VLAN support for a guest LAN is a host initiated enablement for the NIC device with no authorization applied during the joining of VLAN LAN segments. This adapter endpoint type of support is virtualized by z/VM on a virtual NIC device level.
- VLAN support for a virtual switch device is a representation of a physical networking switch with full authorization on a per port basis for membership in a VLAN LAN segment.

This support represents two different levels of virtualization in z/VM. Either of these may be appropriate based on the intended environment. Even though both levels can be deployed concurrently, this is not recommended due to the added complication in determining topology behavior. For example, if you configure a host to define a particular VLAN ID for a given link, and this host is coupled to a virtual switch that is configured as VLAN-aware, the potential exists that a conflict between the two virtualization layers (NIC and virtual switch) will prevent the guest from establishing connectivity to the desired LAN segment. It is recommended that you do not configure a VLAN ID on the link for a VLAN-aware virtual switch.

QDIO versus iQDIO Guest LAN

As stated earlier, if you build a guest LAN as a test or model environment for deployment using physical hardware components, then the choice of guest LAN type is predicated on your target hardware deployment. Otherwise, here are some things to consider in choosing the guest LAN type:

- QDIO (OSA-Express simulation)
  - IPv4 and IPv6 support
  - If using a virtual switch is a deployment potential
  - Ethernet transport
  - OSA-Express test network
  - If using a virtual switch with a link aggregate group is a possibility
  - Port Isolation for the virtual switch
- iQDIO (HiperSockets simulation)
  - IPv4 and IPv6 support
  - Supports multicast router connections
  - Deploy MTUs larger than 8K
  - HiperSockets test network

Chapter 4. Planning for Guest LANs and Virtual Switches

This chapter provides an in-depth guide to planning, configuring, and managing guest LANs and virtual switches within the z/VM environment. It explains the use of virtual network adapters (NICs) such as HiperSockets, OSA-Express (QDIO), and Network Express (EQDIO) for creating virtual networks between guests on the same host system. The document details the differences between guest LANs and virtual switches, their environments, benefits, limitations, and operational modes including IP and Ethernet transport.

The information covers advanced topics like VLAN support, link aggregation, MAC address management, priority queuing, failover mechanisms, and security features such as MAC filtering and port isolation. Additionally, it addresses configuration strategies, automation during IPL, SNMP management, IPv6 considerations, and troubleshooting techniques. The content also highlights distinctions between OSA-Express and Network Express adapters, especially regarding controller requirements and supported features. The guidance ensures that administrators can effectively deploy and maintain virtual networking within z/VM systems.

See the following sources for more information on guest LANs and virtual switches:

Task	Source
Using z/VM connectivity solutions	*z/VM: General Information*
Planning and customizing guest LANs and virtual switches	*z/VM: CP Commands and Utilities Reference* *z/VM: CP Planning and Administration* *z/VM: Getting Started with Linux on IBM Z*
Command information	*z/VM: CP Commands and Utilities Reference*
Messages information	*z/VM: CP Messages and Codes*

Guest LAN Environment

A guest LAN can connect to a physical LAN through a TCP/IP router. (The TCP/IP router could be a z/VM TCP/IP virtual machine, a Linux guest, or a z/OS® guest.) With a guest LAN, data flows through the router to move between the OSA-Express device to the guest LAN. Figure 14 on page 46 shows an example of a guest LAN named **lan1a**. Coupled to **lan1a** are 4 Linux guests, 2 z/OS guests, and 2 z/VM TCP/IP stacks, and attached to the OSA-Express device are 2 z/VM TCP/IP stacks.

Restriction: QDIO mode supports guest LANs. EQDIO mode does not support guest LANs.

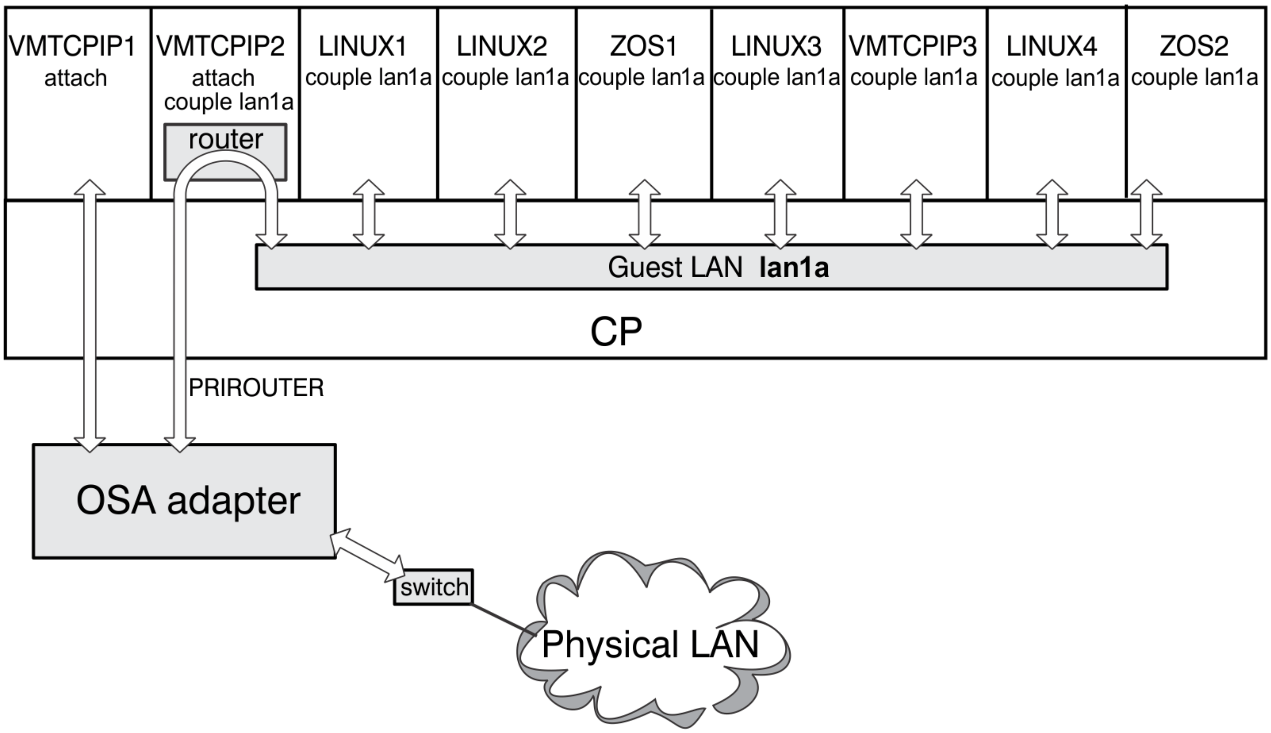


Figure 14. Guest LAN Environment

Use a guest LAN segment to establish a network of virtual adapters. You must define a guest LAN owned by one user or by the CP system. Next, you need to define a compatible virtual network adapter for each user participating in the network. You can do all of this manually, or have it automated during system initialization and guest logon operations. See "Configuring a Guest LAN" on page 78 for more information.

Virtual Switch Environment

A z/VM virtual switch ("virtual switch" or "VSwitch") is a special type of guest LAN. A virtual switch provides a network of virtual adapters to z/VM guests and can be connected directly to a single or multiple OSA adapters. The z/VM control program (CP) provides a virtual networking device that provides switching between an OSA adapter and virtual machines.

The virtual switch connects the virtual machines to the physical (external) LAN segment. The virtual switch manages the OSA adapter control and data connections (read, write, and data devices) and provides a central point of control and management for all data transport between the guest virtual machines and the OSA adapter.

Benefits of z/VM virtual switches

A virtual switch provides the following benefits:

- A virtual switch allows you to gain connectivity to external LAN segments without requiring a router.
- Each virtual switch operates independently, but logically a virtual switch or a portion of a virtual switch can be defined to be part of a Virtual Local Area Network (VLAN). VLANs facilitate easy administration of logical groups of stations that can communicate as if they are on the same Local Area Network (LAN).

Limitations of z/VM virtual switches

A virtual switch has the following limitations and restrictions:

- A virtual switch requires an OSA adapter (OSA-Express or Network Express) to uplink to an external LAN.

- When an OSA-Express adapter is deployed, the virtual switch provides switching for IP or Ethernet transport. When a Network Express adapter is deployed, the virtual switch provides switching for Ethernet transport.
- A virtual switch is more restrictive than a guest LAN in other ways, such as with more access controls (for example, userid with porttype, VLAN and promiscuous attributes).

Additional considerations for z/VM virtual switches in QDIO mode

In QDIO mode (OSA-Express), the control devices (read and write) are automatically attached to the virtual switch controller. The virtual switch controller initializes, configures, and maintains the virtual switch's control connection to the OSA-Express adapter. The virtual switch controls all configuration operations required by the OSA-Express through the QDIO control devices.

The QDIO data device is the virtual switch's trunk connection with the OSA-Express adapter and all data is transported over this device through a set of data queues:

- All traffic for inbound and outbound operations over the OSA-Express flows through the virtual switch over the data device (RDEV). All port-to-port communications between virtual machines are also conducted by the virtual switch.
- The virtual switch controls all configuration operations that are required by the OSA-Express adapter through the control devices.

Additional considerations for z/VM virtual switches in EQDIO mode

The EQDIO device is the virtual switch's trunk connection with the Network Express adapter and all data is transported over this device through a set of data queues:

- All traffic for inbound and outbound operations over the Network Express adapter flows through the virtual switch over the data queues. All port-to-port communications between virtual machines are also conducted by the virtual switch.
- The virtual switch controls all configuration operations that are required by the Network Express adapter through the control queues.

The following special considerations apply to a CHPID that is defined with the LINK_AGGREGATION parameter for Network Express (OSH) devices.

The LINK_AGGREGATION parameter on the DEFINE CHPID command, or the corresponding CHPARM in the IOCP (x02), indicates that the specified CHPID is to be used only for link aggregation. When the CHPID is defined in link aggregation mode, the device can be used only as part of a port group for z/VM virtual switch link aggregation. When the CHPID is not defined in link aggregation mode, attempts to bring up a z/VM virtual switch that is configured to use the device as part of a link aggregation port group will fail if either of the following conditions are true:

- A NETH device is also configured on the port.
- The system is managed by IBM Dynamic Partition Manager.

It is recommended that all CHPIDs that are used as link-aggregation port group members are specifically designated for link aggregation mode in the IOCP. For migration purposes, z/VM supports a device that is part of a link aggregation port group on a CHPID that is not defined in link aggregation mode.

Additional considerations for z/VM virtual switches in QDIO or EQDIO mode

IEEE 802.1Q (VLAN) virtual LAN segments are controlled by the virtual switch, not by the guest virtual machines.

- For a virtual switch managed by user style, the guest's VLAN ID(s) are specified through the GRANT option of the SET VSWITCH command. For a virtual switch that is managed by port style, VLAN ID(s) are specified though the PORTNUMBER option of the SET VSWITCH command or NICDEF directory statement. Optionally, an external security manager (ESM), such as the Resource Access Control Facility (RACF®), can control access to the virtual switch and VLANs.

- An ESM can only manage VLAN ID(s) on a user basis. If a user has more than one NIC connected to the same VSwitch in its virtual configuration, all connections will be authorized to use the same VLAN ID(s). A system administrator can limit a specific port to a subset of the ESM authorized VLAN IDs by using the SET VSWITCH PORTNUMBER or NICDEF directory statement.

Topology example

Figure 15 on page 48 shows an example of a virtual switch named **vsw1a**. Coupled to **vsw1a** are four Linux guests, two z/OS guests, and two TCP/IP stacks. Attached to the OSA device is one z/VM TCP/IP stack (VMTCPIP1).

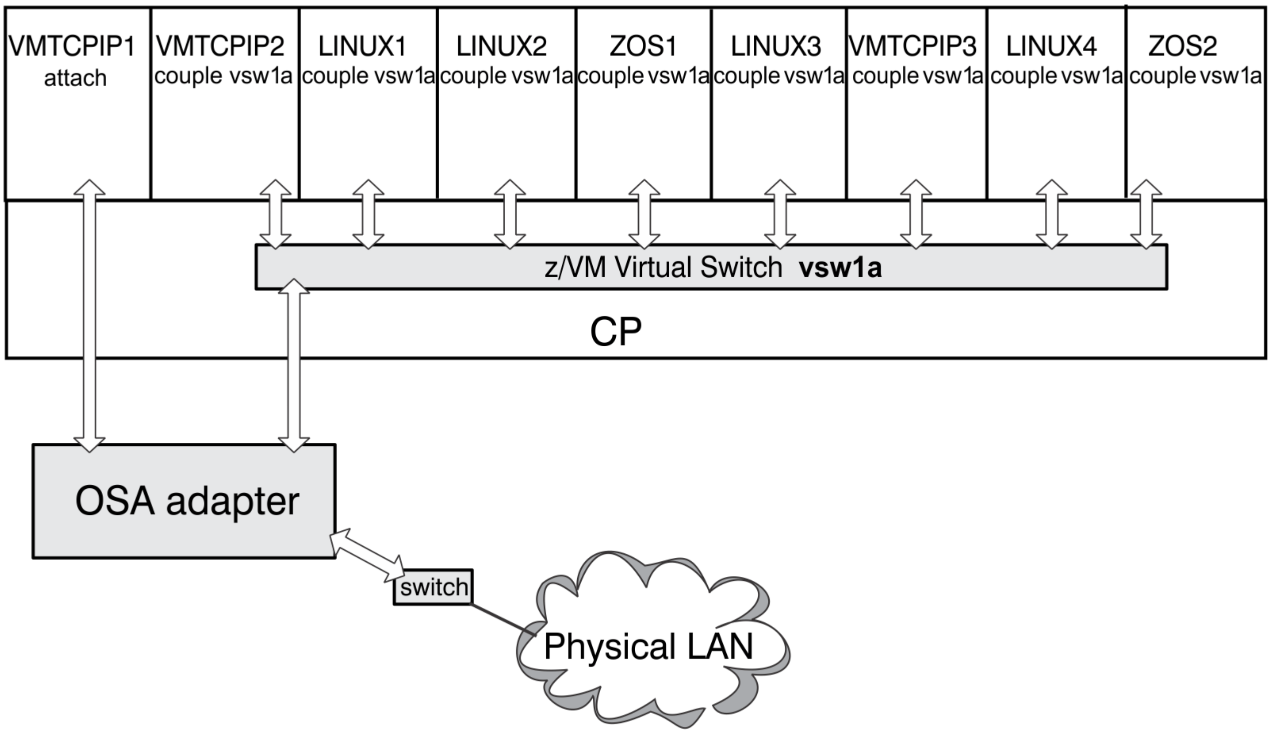


Figure 15. Virtual Switch Environment

UPLINK NIC

Another option for a virtual switch is to configure the Uplink port as a single virtual switch guest port. This can be accomplished with the UPLINK NIC option on the SET VSWITCH command. With this setting, the virtual switch is not connected to a physical switch. The virtual switch routes all broadcasts and destination MAC or IP address unicast frames, that cannot be resolved within the simulated LAN segment, to the specified guest port.

BRIDGEPORT

The BRIDGEPORT operand is used to create a special purpose port on the virtual switch. This Bridge Port provides additional functionality that establishes a binding between HiperSockets and the virtual switch. It is an integral part of the infrastructure converting the different protocols between HiperSockets and the virtual switch network adapter.

For more information on setting up HiperSockets Bridge Port on a virtual switch, see Chapter 7, “Bridging a HiperSockets LAN with a z/VM Virtual Switch,” on page 95.

IPv6 Considerations

When planning to deploy an IPv6 virtual switch configuration that will have external connectivity, the virtual switch must operate in Ethernet mode. Ethernet mode is supported by TCP/IP for z/VM, Linux on z Systems® and DIAG X'2A8'. (For more information on DIAGNOSE Code X'2A8' see *z/VM:*

CP Programming Services). Ethernet mode is not supported by z/OS which precludes it from this configuration. Configurations requiring z/OS and IPv6 can deploy an IP mode guest LAN with a router virtual machine providing external connectivity through the Ethernet mode virtual switch or directly through an OSA-Express adapter (dedicated device).

Link Aggregation

The virtual switch can be configured to operate with aggregated links in concert with a switch that also supports the IEEE 802.3ad specification. Aggregating links with another switch box allows up to eight OSA adapters to be deployed in transporting data between the switches. The adapters must be the same type (OSA-Express or Network Express but not both). This configuration provides the benefits of increased bandwidth and near seamless failover of a failed link within the aggregated group.

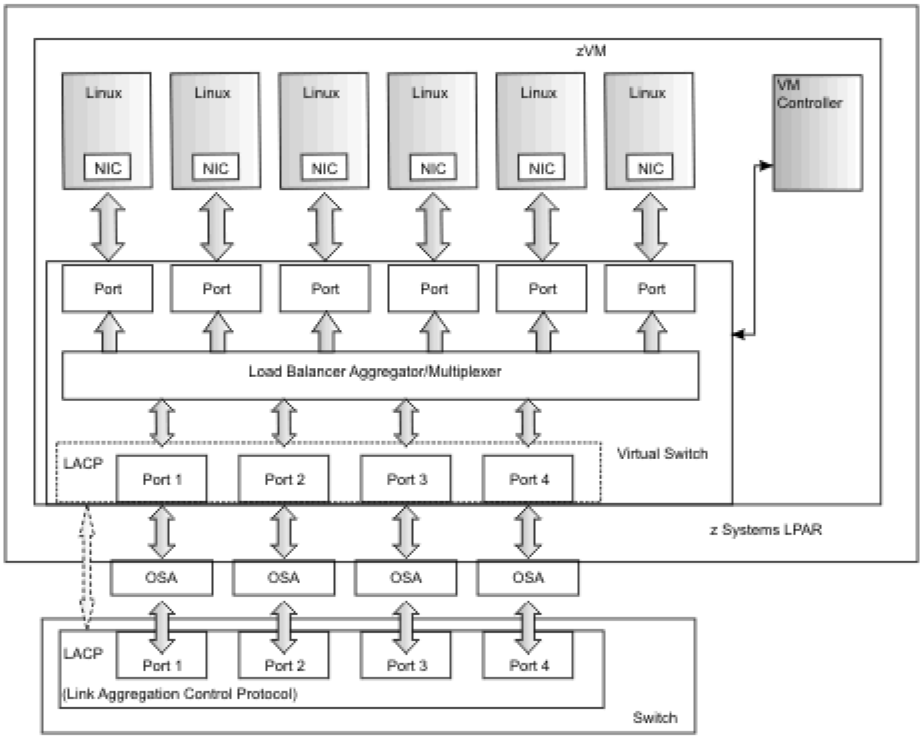


Figure 16. Virtual Switch in a Link Aggregation Environment

All previously stated attributes of the virtual switch are also applicable to the Link Aggregation configuration with the following additions:

- LACP (Link Aggregation Control Protocol) synchronizes and manages the link status between the switches with aggregated links. The interoperability provided by this protocol allows the virtual switch to aggregate links (channels) with switch vendors like Cisco.
- The virtual switch can only be configured with a single port group definition. A port group is configured through the specification of the groupname for the GROUP operand on the SET VSWITCH command. The groupname will be used to identify the target group of devices to be aggregated.
- A group is created through the SET PORT GROUP command for a given groupname by identifying the OSA device addresses that will be members of the group. The virtual switch ports will automatically be created and numbered as a result of the SET PORT GROUP command.

Considerations for Virtual Switch or Guest LAN Configuration

Transport Mode — IP or Ethernet

z/VM LANs support two modes of operation for data transport in support of both TCP/IP and non-IP based applications. In deciding which mode to deploy for your network, some things to consider about deploying an Ethernet virtualized LAN segment are:

- Do your servers or applications need to have their own unique MAC addresses (load balancers)?
- Do you plan to deploy non-IP based applications on your network (SNA or NetBIOS, for example)?
- Do you want to build a virtual LAN segment that operates closely to its physical counterpart?

The key attributes of each transport mode with their operational characteristics are as follows:

- **IP (Layer 3)**
  - Supports IP for TCP/IP applications only.
  - IP packets are transported on the LAN segment.
  - All destinations are identified by IP addresses.
  - IP address assignments are set by the host running in the guest virtual machine.
  - Each host may have more than one IP address (multi-homed).
  - This is Link Layer independent (that is, no MAC addresses), and ARP processing is offloaded onto the OSA-Express adapter.
  - VLAN tagging resides in internal QDIO headers.
  - All hosts share the OSA-Express MAC address.
  - IPv4 networks only.
- **Ethernet (Layer 2)**
  - Supports all applications that deploy Ethernet (IEEE 802).
  - Ethernet frames are transported on the LAN segment.
  - All destinations are identified by MAC address.
  - MAC addresses are locally administered by the LAN administrator through z/VM CP commands or configuration statements.
  - Each host connection is identified by a single MAC address.
  - This is a Link Layer transport, in which all hosts maintain their respective ARP caches.
  - VLAN tagging resides within the Ethernet frames per IEEE 802.1Q specifications.
  - IPv4 and IPv6 networks.
  - Required for deployment of Link Aggregation.
  - Required for deployment of a HiperSockets Bridge Port.

The z/VM LAN operates in only one mode for a given instance. For example, if configured as IP, then all communications on the LAN segment must be IP based. The same is also true when configured in Ethernet mode. These transport modes affect the method of data transfer. For the virtual switch, the other operations, such as guest authorization, failover, controller configuration, and so on, function the same for both modes.

Note: IP layer 3 mode is supported on OSA-Express network adapters but not on Network Express network adapters. Ethernet layer 2 mode is supported on OSA-Express network adapters and on Network Express network adapters.

Media Access Control (MAC) address

z/VM LANs (virtual switch and the guest LAN) and real OSA adapters that are used by the z/VM TCP/IP stack and operate in Ethernet (layer 2) mode support data link layer forwarding of Ethernet frames. In this protocol layer, the MAC address provides the means of identification that is used to forward frames within the LAN segment. The MAC addresses deployed for these LANs on z/VM are classified as Locally Administered. This distinction identifies the MAC address as a locally defined address as opposed to a manufacturer's burned-in MAC address on the physical NIC card. These addresses have the distinctive leading X' 02' value in the first byte of the address (for example: 02-00-41-00-00-07). By default, z/VM generates unique MAC addresses for a single z/VM image (LPAR). However, virtual and real networking devices and LANs deployed across multiple LPARs or even CPCs that reside in the same LAN segment require configuration changes by the LAN administrator to ensure uniqueness of the MAC addresses of all the networking devices that exist across these systems.

Configuring MAC Prefixes

z/VM allows for configuration of unique MAC addresses in your network through a combination of prefixes and suffixes. The prefixes are defined through the MACPREFIX, USERPREFIX, or both of these options on the VMLAN statement in the system configuration file (see The System Configuration File in *z/VM: CP Planning and Administration* for more information). The prefixes define the first three bytes to be used when z/VM generates locally administered MAC addresses on the system. By default these bytes are set by z/VM to be X'02-00-00'. The X'02' is fixed but the remaining two bytes may be changed to create a unique MAC address domain for your z/VM image.

The following is a MACPREFIX example:

```
VMLAN MACPREFIX 020041
```

When z/VM has to generate a MAC address for a networking device on this system, it uses the prefix X'02-00-41'. The remaining three bytes (suffix) of the MAC address (referred to as the MACID) either will be assigned automatically (that is, system-defined) by z/VM or can be user-defined. System defined MAC addresses remain assigned to a device until the guest logs off. If the guest logs on again and creates a NIC, a different suffix may be assigned. If the system administrator wishes to assign persistent, predictable MAC addresses, z/VM provides this capability with user-defined MAC addresses. When the MAC address is user-defined, the suffix can be set by the MACID option on the NICDEF directory statement (see Creating and Updating a User Directory in *z/VM: CP Planning and Administration*) or the MACID operand of the SET NIC command (in *z/VM: CP Commands and Utilities Reference*). When this new MACID is appended to the MACPREFIX, a system-wide unique MAC address is created that identifies the target networking device. Using the NICDEF option in the directory persists across logoff and system IPLs.

z/VM provides two choices for creating user-defined MAC addresses. The preferred method is to use the additional configuration option, USERPREFIX, that defines a three-byte prefix different from the MACPREFIX, to be used exclusively when generating user-defined MAC addresses. That is, the USERPREFIX will be used when a MACID is explicitly specified for the network device via NICDEF or SET NIC. The USERPREFIX method allows clear distinction of MAC addresses assigned by the system administrator versus those generated by the z/VM Hypervisor for SYSTEM MAC addresses.

Another method to attain MAC address uniqueness across multiple z/VM images without altering the MAC prefix is to define a specific unique MAC suffix range to each z/VM image. This is accomplished by defining a different range of MACIDs for a given z/VM image through the MACIDRange SYSTEM operand of the VMLAN statement. The range bounds the MACID (last 3 bytes of the MAC address) when z/VM creates MAC addresses for its networking devices. This allows all the z/VM images to share the same MACPREFIX (possibly to identify a clustered system outside of an SSI configuration) and insure uniqueness through the MACIDs.

As described earlier, z/VM automatically generates (by default) MACIDs for guest virtual NICs as they are defined. To ensure the z/VM system defined MAC addresses do not conflict with your user defined MAC addresses for guest virtual NICs, configure the USER operand of the MACIDRange on the VMLAN statement. The USER operand defines a sub range within the MACIDRange that is reserved for MACIDs specified on NICDEF or the SET NIC command. When creating and assigning MAC addresses those MACIDs that fall within the USER range are not used by z/VM in its automatic (system defined) MAC address creation process.

Note: A MACIDRange cannot be configured if the USERPREFIX has been set to a value different from the MACPREFIX.

Example

The following example statements illustrates how MAC addresses will be set by z/VM with these VMLAN statements in the system configuration file and a NICDEF statement in a user directory:

```
VMLAN MACPREFIX 020041
VMLAN MACIDRANGE SYSTEM 000001-0FFFFF USER 0F0001-0FFFFF

NICDEF FD20 TYPE QDIO MACID 0F0002
```

When z/VM automatically generates a MAC address, the MACID (suffix) will come from the SYSTEM range (000001-0F0000). f the MACID is explicit (from the NICDEF directory statement or the SET NIC command), it must come from the USER range (0F0001-0FFFFF).

Given the previous VMLAN and NICDEF statements, you could expect the following:

```
02-00-41-00-00-01    Set automatically by z/VM for a DEFINE NIC
02-00-41-0F-00-02    Set via a NICDEF statement
02-00-41-00-00-02    Set automatically by z/VM for a Link Aggregation port (OSA)
02-00-41-00-00-03    Set automatically by z/VM for a Link Aggregation port (OSA)
02-00-41-00-00-04    Set automatically by z/VM for a DEFINE NIC
02-00-41-12-22-22    NOT POSSIBLE on this system (122222 > 0FFFFF)
```

Several query commands display MAC addresses in use on the z/VM image. Use the Query VMLAN command to display the MACPREFIX, USERPREFIX and MACID range for your z/VM system. Use Query NIC with the MACID option to display MAC addresses assigned or being used by all or selected network devices. The Query VSWITCH or Query LAN command with the DETAILS option displays the MAC address assignments for all the virtual NICs of a virtual switch/guest LAN instance. The MAC address assignments for port group members are also displayed.

MAC Address Usage in a Link Aggregation Configuration

For Link Aggregation Control Protocol (LACP) and Marker Protocol communications, MAC addresses are also created by z/VM from the system MACPREFIX pool for all ports created through the SET PORT GROUP command to be deployed in a link aggregation group. (For more information, see “Link Aggregation Overview” on page 113).

Single System Image MAC Address Considerations

When a z/VM image is to join a SSI cluster, there are several requirements regarding MAC prefix specification on the VMLAN system configuration statement.

MACPREFIX

The VMLAN MACPREFIX must be set to a different value for each member of an SSI cluster. The MACPREFIX will be used for automatic (system generated) MAC assignments.

USERPREFIX

A USERPREFIX value must be set in order to join an SSI cluster. The USERPREFIX value specified must be identical for all members of the cluster and it cannot be equal to any member's VMLAN MACPREFIX value. The common USERPREFIX value will be used as the first three bytes of customer defined MAC assignments (concatenated with the MACID from the NICDEF directory statement or the SET NIC MACID command).

Example (3 system SSI cluster)

```
System 1:
VMLAN MACPREFIX 021111 USERPREFIX 02AAAA

System 2:
VMLAN MACPREFIX 022222 USERPREFIX 02AAAA

System 3:
VMLAN MACPREFIX 023333 USERPREFIX 02AAAA
```

Note that in the absence of these VMLAN statements in an SSI configuration, z/VM will default the MACPREFIX to 02xxxx where 'xxxx' is the SSI slot number assigned the system. USERPREFIX defaults to 020000. Additionally, VMLAN MACIDRange statements are ignored in a SSI cluster.

MAC Address Protection (MAC Filtering)

Whether a MAC address is system-defined or user-defined, each networking device has one assigned MAC address that is to be used for its network connection. This MAC address is exchanged during QDIO connection initialization as well as the source MAC address in outbound Ethernet frames originated by the guest. z/VM provides a mechanism where the system administrator can configure MAC address protection for Ethernet type virtual network devices on the system. Using these mechanisms provides a means to ensure that a guest uses the assigned MAC address and does not attempt to masquerade as another guest. These MAC protection options in z/VM enforce whether a guest can override the setting of the MAC address during QDIO initialization and whether the guest must use the assigned MAC address in outbound Ethernet frames.

There are three levels of inheritance used when determining the MAC address protection level for a MAC address assigned to a network device. The highest is the system level, followed by the virtual switch or guest LAN level. The lowest level is the protection set for a specific network data device. See the MACPRotect operands on the VMLAN configuration statement as well as the SET LAN, SET VSWITCH, and SET NIC commands in *z/VM: CP Commands and Utilities Reference* for more information.

z/VM will only allow a guest to override the MAC address assigned to the device if:

- MAC protection is not in effect for the device
- the MAC address is locally administered (X'02 bit on)
- the MAC prefix must not match any of the z/VM prefixes (MACPREFIX or USERPREFIX).

Path MTU Discovery Protocol (PMTUD)

PMTUD is a standardized technique for determining an acceptable Maximum Transmission Size (MTU) between two IP network connections. The goal of this technique is to discover the largest size datagram that does not require fragmentation anywhere along the path between the source and destination. This discovered datagram size is known as the Path Maximum Transmission Unit (PMTU) or this also known as the "Effective MTU for sending" [1].

The virtual switch can be configured to provide MTU discovery responses for payloads that are determined by the virtual switch to be larger than the supported MTU of the OSA Uplink port or the system administrator configured MTU of the external LAN. The PMTUD process will cause the virtual switch to respond to the sending guest with an ICMP error response which contains the acceptable MTU. The MTU value used to check payloads to be sent over the virtual switch Uplink port is provided by the PATHMTUDISCOVERY operand of the SET VSWITCH command. By default the virtual switch will use the largest allowable MTU supported by the network adapter that is currently deployed as the Uplink port. PMTUD is provided for guests connected directly to the virtual switch (simulated NICs) and the HiperSockets Bridge Capable ports.

Connecting a virtual switch to a HiperSockets channel introduces the complexity of having a local broadcast domain with different MTU sizes, requiring infrastructure to orchestrate an acceptable MTU size between source and destinations that are connected through the bridged networks. The virtual switch PMTUD provides the ability for the HiperSockets guest port to guest port communications to enjoy the performance benefits of using large MTUs (MFS) of the HiperSockets channel, while also supporting external destinations for HiperSockets bridge capable ports with lower MTUs. Bridge capable guests ports or simulated ports with a single network connection supporting PMTUD, can communicate optimally with all destinations over the entire broadcast domain.

[1] As documented in RFC 1191

Additional Considerations for Virtual Switch Configurations

The role of the virtual switch is to provide external network connectivity through one or more OSA adapters (Uplink ports) for a z/VM simulated LAN segment. Deployment of a virtual switch reduces the CPU utilization cost and latency associated with providing external connectivity through a router virtual machine. The switching logic resides in the z/VM Control Program (CP), which owns the OSA connection and performs all data transfers between virtual machines connected to the virtual switch and the OSA adapter. This eliminates the overhead associated with a router running in a virtual machine performing this same function.

The following topics are additional considerations for a virtual switch.

Virtual Switch Controller Configuration Strategy

The virtual switch controller is a z/VM TCP/IP server that provides the configuration and link management that is required by OSA-Express adapters that are configured to a virtual switch.

Note: A virtual switch controller is used when an OSA-Express adapter is configured to a virtual switch. A virtual switch controller is *not* used when a Network Express adapter is configured to a virtual switch.

The controller provides the following configuration and management:

- Initialization tasks.
- Inbound traffic flow, which is managed through the synchronization of CP's IP and multicast group tables with the OSA-Express to ensure inbound connectivity.
- All link recovery operations for a virtual switch in the event of an OSA-Express Uplink port failure or a HiperSockets bridge port failure.

The controller does not require an IP address because it is not involved in data transfers between the virtual switch and the OSA-Express adapter.

When virtual switch definitions are processed, virtual switch controllers are assigned by z/VM from the pool of available controllers. When a virtual switch's controller becomes inoperative, another controller is assigned from the pool. A single controller can support more than one virtual switch. z/VM includes four pre-built virtual switch controllers: DTCVSW1, DTCVSW2, DTCVSW3 and DTCVSW4.

Consider the following recommendations for virtual switch controller configurations:

- For configurations that deploy one or two virtual switches per z/VM system, the four controllers that are provided by z/VM will be sufficient. As configurations are expanded to more than two virtual switches per z/VM image and/or grow with link aggregation configurations, the four controllers might no longer be sufficient. As more load is placed on a given controller, longer recovery times for failover situations will be experienced. More controllers can be created. See Appendix F, “Creating a VSwitch Controller,” on page 337.
- Even though a given controller can service more than one virtual switch, optimum operational performance is achieved when there is a one to one pairing of virtual switch to controller.
- Extra controllers might be required to isolate messages for a specific virtual switch in a test environment or when TRACE OSD or MORETRACE OSD is used to debug a problem.
- When using the OSD trace, virtual switch timeout (stall) checking should be disabled. To disable stall checking, specify the FAILOVER_DISABLED option on the VSWITCH CONTROLLER statement in the controller's TCP/IP configuration file.
- Have your controllers up and ready prior to instantiating any virtual switches. In this way as you instantiate virtual switches, z/VM will evenly distribute the pairings of Uplinks and controllers. z/VM assigns and spreads the load across the entire pool of available controllers.
- In almost all cases:, virtual switches should be defined with the default of "CONTROLLER *"
- More information on the operational aspects of virtual switches and controller failover is available. See “Virtual Switch Failover” on page 67.

Virtual Switch Priority Queuing Function

All OSA-Express adapters (QDIO mode) and all Network Express adapters (EQDIO mode) support priority queuing, which allows a system administrator to influence the order the adapter will transmit data to the physical network. Without priority queuing, all network traffic will be sent to the external network at the same priority. By default, a z/VM virtual switch does not exploit priority queuing.

A system administrator can configure the z/VM virtual switch to exploit the OSA adapter's priority queuing capabilities by setting PRIQueuing ON when defining a VSWITCH with either a DEFINE VSWITCH command or dynamically with a SET VSWITCH command. Setting PRIQueuing ON dynamically will require the uplink port to be disconnected using the SET VSWITCH *name* DISCONnect command.

In order for the z/VM virtual switch to exploit the OSA adapter's priority queuing function, the following device and virtual switch configuration settings are required:

1. For OSA-Express (QDIO) adapters only: I/O Configuration Definition

   By default, priority queuing is enabled on an OSA-Express CHPID (PQ_ON). Disabling priority queuing on the OSA-Express CHPID (type OSD) prevents the virtual switch from exploiting priority queuing, but allows more dedicated OSA-Express devices to be available to z/VM guests.

   Note: For Network Express (EQDIO) devices, priority queuing is always enabled and cannot be disabled.
2. Virtual switch enablement

   Code PRIQUEUING when you define the virtual switch or when you use a SET VSWITCH command before you establish a network connection on the virtual switch's uplink port.
3. Optional System Administrator NIC Priority Assignment

   Assign a High, Normal, or Low priority for each NIC coupled to the virtual switch using the PQUPLINKTX operand on the NICDEF directory statement. Use the SET VSWITCH command with the PQUPLINKTX operand to make dynamic changes.

With PRIQueuing ON, the z/VM virtual switch will establish four queues, one for each priority. The highest priority queue is exclusively used by the z/VM virtual switch to provide switch management communications and the other three priority queues may be configured to a virtual machine's network connection by a system administrator.

The network adapter will process all four queues by using a fair share algorithm, which will insure that all queues are processed.

Each NIC defined with the NICDEF directory statement can be assigned either a HIGH, NORMAL, or LOW priority using the PQUPLINKTX operand. If a priority is not set by a system administrator, the NIC is automatically assigned to NORMAL priority.

IVL Virtual Switch Priority Queuing

An IVL virtual switch will always attempt to exploit the network adapter's priority queuing function on its uplink port. There is no option to turn off this function on an IVL virtual switch. z/VM will use priority queuing for the following adapters:

- An OSA-Express adapter that is enabled for priority queuing.
- A Network Express adapter. (Priority queuing cannot be disabled on a Network Express adapter.)

Activating an IVL virtual switch's uplink port with an OSA-Express adapter that has priority queuing disabled (PQ_OFF) will result in a warning message being displayed to enable the priority queuing function on the OSA-Express adapter. Although the network connection will operate without priority queuing, it is highly recommended the function is enabled on the adapter.

The IVL virtual switch uses priority queuing to isolate data transmissions that manage the IVL Network from production traffic in the event of a device error occurring within a Shared LAG Port Group. With priority queuing, two output queues instead of a single queue will be established to communicate with the network adapter. This will allow IVL management transmissions to flow at a higher priority queue and any error recovery traffic to flow at a lower priority queue. This will prevent a high bandwidth error recovery transmission from disrupting management of the IVL Network.

Relocation Information for Priority Queuing

- Virtual Switch Priority Queuing does NOT need to be enabled/supported on both the source and destination systems.
- Guests with NORMAL priority can always relocate.
- When the source virtual switch has Virtual Switch Priority Queuing enabled and the guest NIC has a HIGH or LOW priority assigned, then Virtual Switch Priority Queuing must be enabled on the target virtual switch.

OSA-Express Multiple Ports Function

Some OSA-Express adapters provide multiple ports on a single Network Interface Card (NIC). The multiple ports provide more physical connections without requiring multiple channel path ids (CHPID) to be defined. For example, with an OSA-Express adapter having two channels, where each channel supports two ports, four physical ports are available for use.

Restriction: The multiple ports function applies to OSA-Express adapters but not to Network Express adapters.

The virtual switch can make use of the additional ports through specification of a port number in the configuration commands that define the OSA-Express adapters for the virtual switch (for example, RDEV operand on DEFINE or SET VSWITCH, SET PORT GROUP). There are no changes required to the system I/O definitions (for example, IOCDS, IOCP, or HCD) to utilize the additional ports. A device can be associated with only one port number at a time.

Each port can be connected to the same physical network, or it can be connected to different networks. From a port sharing perspective, connectivity exists between hosts and or virtual switches that are configured on the same port. There is no bridging between ports on an OSA-Express adapter. Each port is isolated from one another.

Deploying multiple ports of an OSA-Express adapter as members of a Link Aggregation group can reduce the number of adapters required. Each port is a separate autonomous link. So today, a link aggregation configuration that requires four adapters, could be re-configured with two OSA-Express adapters with 2 ports each. Remember it is recommended that a minimum of two adapters be aggregated to providing transparent recovery within a link aggregation group. When an OSA-Express adapter fails, all ports on the adapter may be affected by the outage depending upon the type of error incurred.

Exclusive LAG Configuration Considerations

- A single port, not the entire adapter, is dedicated to a single VSwitch in a single z/VM system
- One port provides connectivity to a link aggregation group (exclusive use mode) while the other port may be configured to a shared port group, an exclusive group, directly to a VSwitch or a guest virtual machine
- Hardware enforced (OSA-Express adapter)

Multi-VSwitch LAG Configuration Considerations

- A single port may be shared by multiple VSwitches in multiple z/VM systems
- One port provides connectivity to the link aggregation group, while the remaining port(s) may be configured to another shared port group, an exclusive group or directly to a VSwitch. Sharing of the remaining port(s) of the OSA-Express adapter with other guest virtual machines or operating systems is NOT supported
- Software enforced (z/VM system)

See also “OSA-Express Multiple Ports Function” on page 83 for information about configuring additional ports for use by the z/VM TCP/IP stack.

Virtual Switch Management (SNMP)

z/VM provides an SNMP subagent for virtual switches providing the capability to manage these switches from a Network Management System (NMS).

The SNMP architecture consists of network management stations (SNMP clients), network elements (hosts and gateways), network management agents and subagents. Network management agents perform information management functions, such as gathering and maintaining network performance information and formatting and passing this data to clients when requested. This information is collectively called the Management Information Base (MIB).

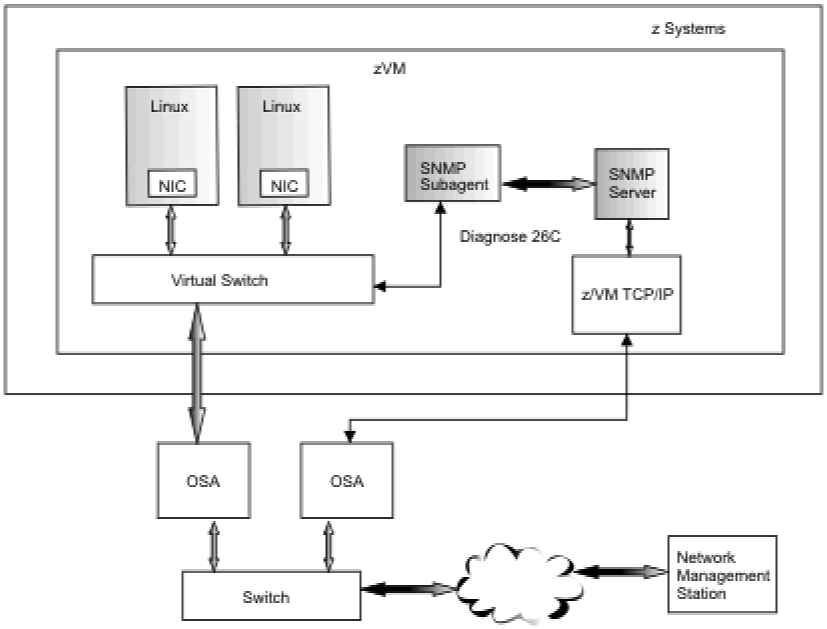


Figure 17. SNMP Connectivity for the Network Management System

An SNMP subagent provides an extension to the functionality provided by the SNMP agent. The virtual switch SNMP subagent provides RFC 1493 BRIDGE_ MIB variables in response to GET or GETNEXT requests from a NMS. When requests for these variables are received by the z/VM SNMP agent, the agent passes the request to the subagent. The subagent acquires the information using the CP diagnose interface, fill out, and return a response to the SNMP agent. The agent then creates an SNMP response packet and sends the response back to the remote network management station (SNMP client) that initiated the request. The existence of this new subagent is transparent to the network management station.

The virtual switch can be configured to deploy SNMP to provide status for one or more virtual switches using SNMP Bridge MIBs and to report virtual networking events such as link up or link down. The network connection for the z/VM TCP/IP stack providing SNMP connectivity for the Network Management System (NMS) should be separate from the managed virtual switches, so that a failure of a virtual switch connection can be reported to the NMS using SNMP traps.

Guest Port Traffic Forwarding Strategies for the Virtual Switch

The virtual switch configuration options provide three distinct modes of operation with respect to guest-to-guest and guest-to-external destination communications.

Default
Guest ports can communicate with each other and with hosts and/or LPARs that share the same OSA port (subject to VLAN authorization).

ISOLATION ON

Guest ports are prohibited from sending traffic to other guests on the same virtual switch or to hosts that share the same OSA port.

VEPA ON

All traffic from a Guest port is sent directly to the external switch regardless of the destination. The external switch is then able to forward this traffic back to the VSwitch for delivery to the intended destination. VEPA provides the means to leverage policy enforcement in the physical switch to monitor, secure, and manage Guest port traffic. VEPA is also supported by the OSA-Express adapter in concert with the z/VM VSwitch. Physical switch support of reflective relay is required.

Restriction: VEPA is currently not supported for Network Express adapters.

Default Mode

In the default operation mode, a virtual switch and an OSA port can be shared by multiple TCP/IP stacks. In such a configuration, when a packet is sent by a guest port on the VSwitch and the next hop IP address is registered by another TCP/IP stack that shares the virtual switch or OSA port, then the packet is directly routed to the sharing stack without traversing the external LAN.

Note: In some OSA documentation this is referred to as port sharing or "image-to-image" communication.

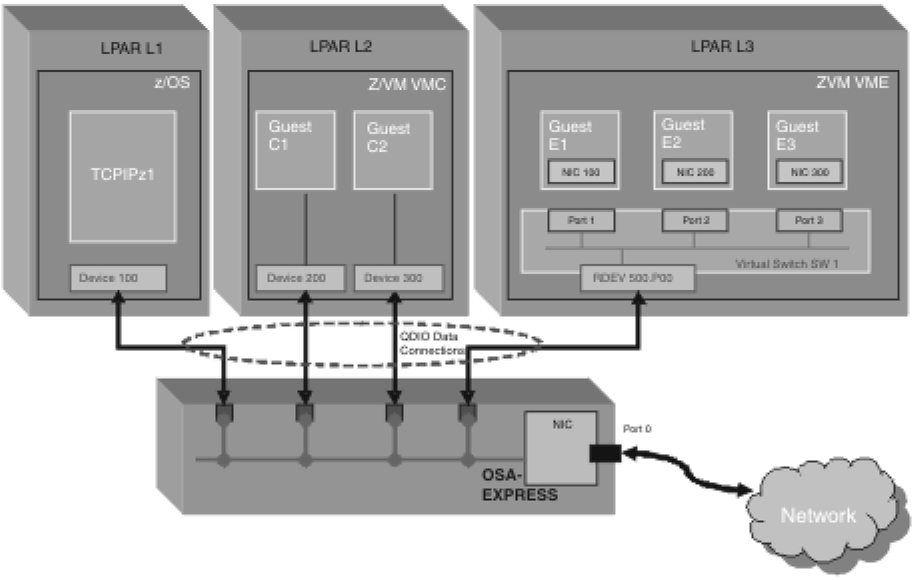


Figure 18. Typical IBM Z CPC Configuration

In Figure 18 on page 58, full connectivity is available to all sharing hosts and LPARs that are connected to the same OSA port (Port 0) and between all guest ports (Port 1, Port 2, Port 3) on virtual switch SW1. The guests (E1, E2 and E3) can communicate with each other or with z/VM guests in other LPARs (C1, C2 and TCPIPz1). This virtualization provides communications that transpire all within the boundaries of the CPC (Central Processor Complex). Note that none of this communication is sent to the external network.

Note: Figure 18 on page 58 shows an OSA-Express adapter and indicates a QDIO data connection. The same configuration applies when a Network Express adapter is deployed, in which case the data connection uses EQDIO mode.

Isolation Mode

Virtual switch port isolation provides the means to prevent packets from being directly sent to another host on the virtual switch itself or the network adapter. When port isolation is in effect, the virtual switch and network adapter will discard any packets when the next hop address was registered by a sharing host. The OSA adapter requires that the virtual switch that shares the port be non-isolated for direct routing to occur. When port isolation is in effect on a virtual switch, the only way a packet can flow to another host that shares the network adapter is to first go through a router on the external LAN and then onto the target host.

The virtual switch administrator has the ability to restrict internal virtual switch traffic and restrict the communications between the virtual switch data connection and other sharing hosts and/or LPARs on the same network adapter. The restriction can be used to establish security zones or meet established security policies. This restriction of communication is achieved by placing the virtual switch into ISOLATION mode through the use of the SET/MODIFY VSWITCH commands.

ISOLATION mode for the virtual switch consists of:

1. Isolating all active guest ports on the virtual switch, which prevents direct (intra-virtual switch) communications.
2. Isolating its OSA data connection (RDEV 500.P00) from other sharing LPARs on the same OSA port.

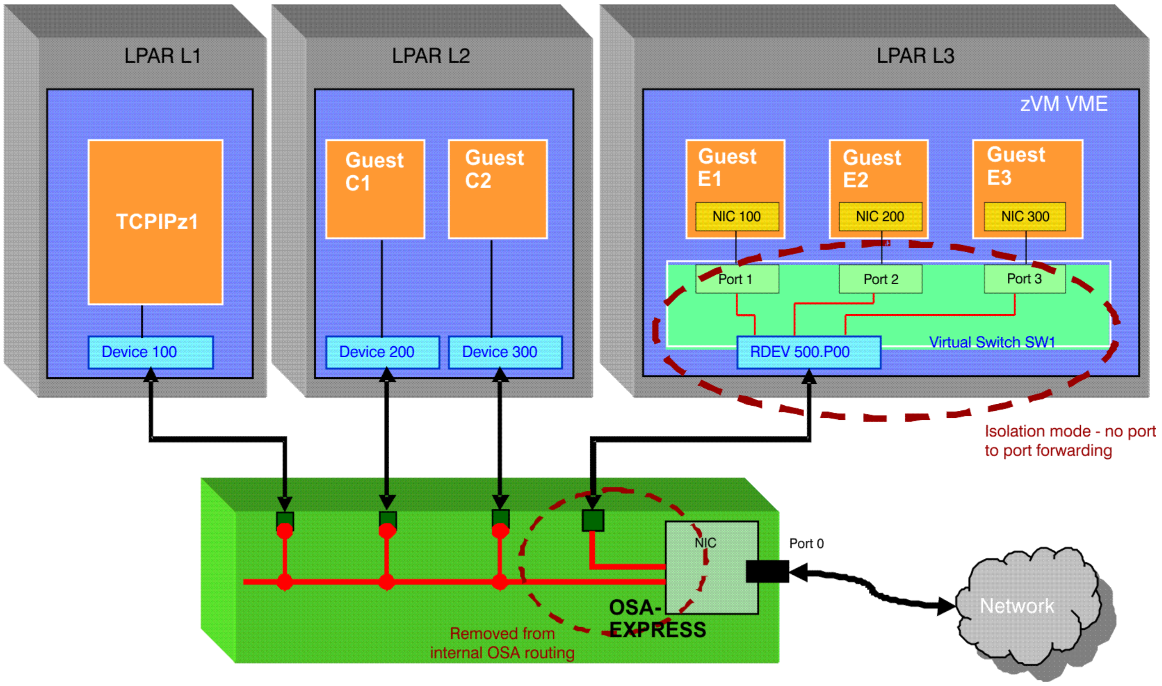


Figure 19. Virtual Switch Port Isolation

The ISOLATION ON option on the SET VSWITCH command can be used to place the virtual switch into isolation mode. Isolation mode affects all internal (guest to guest) communications, unicast, broadcast, and multicast traffic along with the virtual switch's RDEV data connection. Figure 19 on page 59 presents ISOLATION made in both the virtual switch SW1 and its RDEV data connection on the OSA port. Note that the remaining sharing hosts/LPARs that are connected to the OSA port are unaffected by this isolation change and continue to communicate with each other as usual. Any traffic from the sharing hosts/LPARs that is targeted for the isolated virtual switch connection (Guests E1, E2 and E3) is dropped.

The ISOLATION ON option restricts the internal traffic on the virtual switch to only traffic that is destined for the external network. All guest-to-guest traffic on the virtual switch is dropped. When the virtual switch is functioning in IP mode, dropped unicast packets are recorded as discarded packets. When the virtual switch is functioning in ETHERNET mode, ARP broadcasts to guest ports are dropped and not recorded as discarded packets. Any dropped unicast packets are recorded as discards. The ON option only permits guest ports to communicate with external servers or clients on the network providing a secure zone boundary within the CEC.

In practice, an installation that simply desires that the virtual switch guest ports be isolated from all local traffic will just need to configure the ISOLATION ON operand. Installations that desire an external ACL or firewall to control the locally destined traffic will also need to configure routes in their guest's TCP/IP routing tables to route this traffic to a gateway router that has been configured to provide this control.

Virtual switch isolation mode in conjunction with the an external firewall or ACL provides a means for controlling traffic flow between guest ports and sharing LPARs on the same OSA port.

For OSA-Express adapters, the use of the VEPA option can be considered (see "VEPA Mode" on page 60).

Note: Figure 19 on page 59 shows an OSA-Express adapter and the example mentions QDIO data connections. The same configuration applies when a Network Express adapter is deployed, in which case the data connection uses EQDIO mode.

Port Isolation Routing Considerations

Port isolation provides a way to prevent the virtual switch and the OSA-Express adapter from internally routing packets directly to a sharing host. When port isolation is in effect, any unicast packets will be discarded when the next hop address was registered by a sharing host and will also prevent any multicast or broadcast packets from being routed between the hosts.

Dynamic routing will not work through the LAN between two hosts on a LAN when OSA port isolation is in effect between them. However it does not preclude traffic between the stacks over the uplink, but this can only be done by using a static route with a next hop address of a router on the LAN. Doing this can result in excessive ICMP redirect packets from the router to the originating host. Also, if you use such a static route technique, you should turn off receipt of ICMP redirects on the sharing hosts, and if possible you should configure the router to not send ICMP redirects.

This function can be useful when you want to prevent communication between two hosts that share the same virtual switch or OSA port. It also provides extra assurance against a misconfiguration that might otherwise allow such traffic to flow. It can also be useful if you want to ensure that traffic that flows through the uplink does not bypass any security features implemented on the external LAN.

Tip: If you want traffic to flow between two hosts that share a virtual switch or OSA port but ensure that the traffic flows over an external LAN, you should configure each stack on a separate VLAN.

VEPA Mode

Virtual Edge Port Aggregator (VEPA) is part of the IEEE 802.1Qbg standard that is designed to reduce the complexities associated with highly virtualized deployments such as hypervisor virtual switches that bridge many virtual machines. VEPA provides the capability to take all virtual machine traffic that is sent by the server and send it to an adjacent network switch. This mode of operation moves all frame relay switching from the hypervisor virtual switch to the (external) adjacent switch. With the adjacent switch handling the frame relay for VSwitch guest port to guest port communications, imbedded network based appliances in the adjacent switch such as firewalls, Access Control Lists (ACLs), Quality of Service (QoS), and port mirroring can be deployed for this guest port to guest port switching. VEPA eliminates the need to provide and support these network based appliances in the hypervisors/LPARs. The IEEE 802.1Qbg standard contains a means for an adjacent (layer 2) switch to support a VEPA mode with a virtual switch through a Reflective Relay (also known as Hairpin Turn) which enables packets received on the Switch Port to be "reflected" back on the same Switch Port.

The VEPA ON option on the SET VSWITCH command can be used to place the virtual switch into VEPA mode. Similar to ISOLATION ON, VEPA mode affects all internal guest-to-guest communications – unicast, broadcast, and multicast traffic – along with the virtual switch's Uplink RDEV data connection. Figure 20 on page 61 illustrates a VEPA environment.

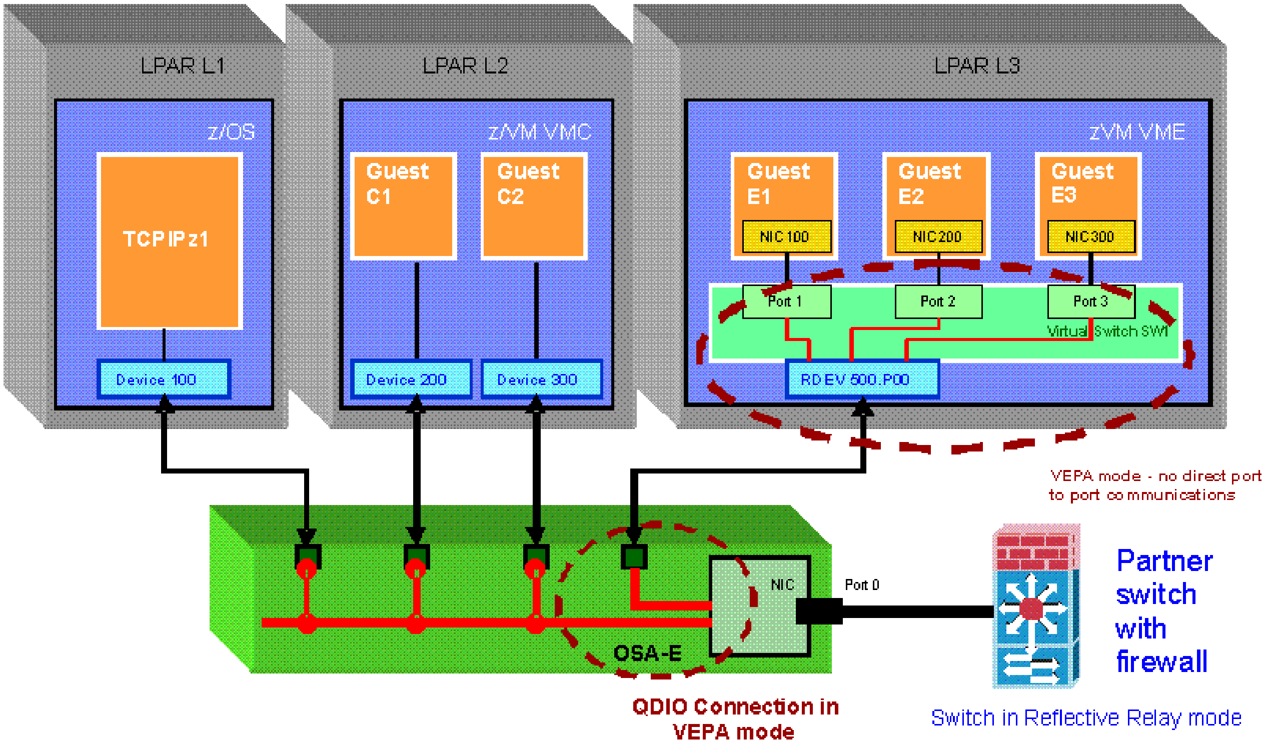


Figure 20. z/VM VSWITCH-initiated VEPA Mode

In VEPA mode, there are no direct communications between guests coupled to the VSwitch (E1, E2, E3). All communication is forwarded to the partner switch. Also, there are no direct communications between sharing connections on the OSA adapter and the VSwitch Uplink RDEV. All direct communications from and to the VSwitch Uplink connection are sent to the partner switch. In each of these cases, the partner switch determines whether the frames will be sent back for delivery to the destination via the Reflective Relay capability.

VEPA ON requires an ETHERNET virtual switch (without a Bridge Port) with OSA uplink(s) that supports VEPA as well as the partner switch must support Reflective Relay.

Restriction: VEPA is not supported for Network Express adapters.

Promiscuous Mode

Promiscuous mode on an individual guest is not affected by the virtual switch isolation or VEPA modes. Promiscuous guests will still receive a copy of the guest to guest internal traffic when the virtual switch is operating in either of the two isolation modes. For more information on promiscuous mode, see Chapter 10, “Troubleshooting a Virtual Switch or Guest LAN,” on page 147.

Link Aggregation

A virtual switch that is operating with a Link Aggregation group already has isolated RDEV QDIO or EQDIO connections. Each RDEV in the port group is a connected to an OSA port that is in exclusive use mode. An OSA port that operates in this mode has no other LPARs or hosts that share the OSA port. Essentially the virtual switch has exclusive use of the OSA port. If guest port isolation is also required, then the SET VSWITCH ISOLATION ON command must be deployed. Note that this command will not attempt to isolate the RDEV connections within the port group but will insure that the backup RDEV connection will be isolated when activated during a failover incident. Similarly, SET VSWITCH VEPA ON must be configured if policy enforcement is to be provided by the physical switch.

Virtual Switch Configuration Strategy with VLAN

The virtual switch supports VLANs as described in IEEE Standard 802.1Q. VLANs increase traffic flow and reduce overhead by allowing the network to be organized by traffic patterns rather than physical locations. The virtual switch provides a single point of management for deployment of VLAN segmentation in the z/VM environment. Through the virtual switch, VLAN segmentation can encompass external networks as well, providing the ability for virtual machines to belong to VLANs that extend beyond z/VM's virtual network. Through its OSA trunk port, the virtual switch ensures adherence to established VLAN LAN segment boundaries in both its inbound and outbound data transfer operations. Within a VLAN set, hosts in the external LAN segments and hosts in a guest LAN segment appear to be in the same LAN segment.

VLAN ID 1 Considerations

Some switch vendors use VLAN ID 1 as the default value when a VLAN ID value is not explicitly configured. It is recommended that you avoid the value of 1 when configuring a default VLAN ID value. Native VLAN recommendations:

- If untagged traffic is desired then set NATIVE 1
- If untagged traffic is not desired then set NATIVE NONE

VLAN-Aware or VLAN-Unaware

Note: The VLAN_counters option is turned on to maintain transmission counters at the VLAN level. In order to maximize the performance when this counting is performed, choose VLAN IDs in which the low-order nibble differs (For example, choose X'100' and X'101' instead of X'100' and X'200'). If possible, the use of consecutive VLAN IDs is recommended.

A virtual switch may be configured as either VLAN-aware or VLAN-unaware. This configuration option defines the behavior of the virtual switch when operating in a LAN segment that may be virtually segmented by VLANs. The virtual switch accepts all packets and frames and passes them to its ingress rules processing whether they are tagged or not. The virtual switch's VLAN support is based on the IEEE 802.1Q specification and governs how the virtual switch processes frames or packets that are VLAN tagged:

VLAN-unaware

The virtual switch ignores VLAN tags in frames or packets. Essentially, the virtual LAN segment that may have been defined by another switch is not recognized. Frames and packets are filtered and delivered by their destination MAC or IP address. Any VLAN tagging that exists is ignored and remains in the frame. The virtual switch does not strip the tag from the frames (Ethernet mode), so if the virtual switch receives a tagged frame, the frame is delivered tagged to the destination MAC. This includes traffic destined for remote hosts through the OSA trunk port. Tagged traffic flows unimpeded through the virtual switch to guest virtual machines and the OSA trunk connection.

It is recommended that hardware switch port that is connected to the OSA port be configured as an access port. This insures that the virtual switch only processes traffic for the VLAN that has been configured on the hardware switch access port.

VLAN-aware

The virtual switch enforces the VLAN defined topology constructed by the LAN administrator. The virtual switch uses the VLAN tagging in conjunction with the IP or Ethernet address for delivery of packets or frames. The port type definition for the destination MAC determines whether the frames are delivered (tagged or untagged). All inbound (ingress) and outbound (egress) frames or packets are processed according to the defined rules. The virtual switch supports one untagged VLAN LAN segment of which the OSA trunk port and those guest trunk ports which have been explicitly granted the designated native VLAN ID are members. VLAN construction and assignments are a manual process for the virtual switch through z/VM CP commands or an ESM. The virtual switch’s native VLAN ID is set with the DEFINE VSWITCH command. The OSA-Express adapter that is associated with a VLAN-aware virtual switch becomes a participant or an end station in a GVRP network. VLAN topology changes must be implemented statically using the SET VSWITCH command, the MODIFY VSWITCH system configuration statement, or an ESM. That configuration is propagated to the hardware switch, eliminating the need for manual configuration. GVRP is not supported on a Network Express device.

The hardware switch port that is connected to the OSA port must be configured as a trunk port. This insures that the virtual switch receives VLAN traffic for all VLANs that have been configured for the virtual guest ports.

Native and Default VLAN ID

The virtual switch supports the designation of each of these VLAN types. The purpose of each is to assist in the management of VLAN assignments and traffic flow through the virtual switch. Each switch manufacturer's has similar constructs for these VLAN designations so it is advised that the manufacture’s documentation be consulted when configuring. For the virtual switch these VLAN designations have the function attributes:

Default VLAN

- Is specified on the DEFINE VSWITCH command through the VLAN operand.
- The VLAN ID specified as the *defvid* is assigned by CP to all ACCESS type guest ports that have not implicitly been assigned a specific VLAN ID. The virtual switch tags all outbound frames from the guest access port with this VLAN ID and delivers all traffic tagged with this VLAN ID destined for this guest access port with the tag removed from the frame.
- When the default also takes on the role of the native VLAN ID, this results in unintended traffic being sent to the guest access ports with the default VLAN ID. For example all untagged broadcasts are forwarded to all access port guests who have been assigned the default VLAN ID.
- Consider setting NATIVE NONE and force all traffic to an appropriate VLAN ID. Untagged packets will be dropped.

Native VLAN

- Is specified on the DEFINE VSWITCH command through the NATive operand.
- The Native VLAN ID should be the same VLAN ID for the untagged set in the other switches connected to this same LAN segment. This provides a discernible means of identifying which ports are communicating with untagged frames.
- The VLAN ID specified as the *natvid* is deployed by CP to forward all untagged traffic received from a trunk port (guest or OSA uplink). For example, if a trunk port has VLANs 2, 3 and 4 assigned and VLAN 2 the Native VLAN, then frames on VLAN 2 that egress (exit) the trunk port are not given a tag. Frames which ingress (enter) this trunk port do not have a tag. Untagged frames are forwarded to and from trunk ports that are assigned VLAN 2.
- A VLAN-aware virtual switch’s OSA uplink port is automatically configured as a trunk port and assigned the Native VLAN ID.
- As a general rule configure the Native VLAN to those guest trunk ports that actually require access.

External Switch and OSA Configuration Recommendations

The OSA adapter that is deployed for the virtual switch's physical LAN connectivity can be connected to switching devices. These devices, like the virtual switch, must be configured properly to ensure desired operational characteristics. When the virtual switch is configured to operate as VLAN-unaware, the port on the external switch for the OSA adapter's connection should be configured as an access port or untagged. Subsequently, when the virtual switch is configured to operate as VLAN-aware, the port on the external switch for the OSA adapter's connection should be configured as a trunk port or tagged.

Through the port sharing capability in the OSA adapter, virtual switches that operate in separate z/VM images or LPARs can communicate directly through the same OSA adapter without sending data out over the physical network. This support is transparent to the virtual switch.

When OSA devices are grouped (configured) into a Link Aggregation group, they are no longer sharable with other operating systems or LPARs. Once an OSA device is removed from the group, it will be sharable once again. For more information on exclusive use of OSA devices see "Exclusive LAG Configuration" on page 119.

Guest VLAN Membership and Port Type

VLAN topology and traffic flow are controlled by the LAN administrator through the z/VM commands for the virtual switch, and/or by the security administrator if an ESM is controlling access to the virtual switch. If Directory Network Authorization (DNA) is enabled, each virtual NIC can be configured in USER DIRECT with NICDEF PORTNUMBER, PORTTYPE, and VLAN. In that case, the SET VSWITCH commands are unnecessary (but can be used to make immediate changes in network configuration). The PORTTYPE operand defines the type of connection that is established when the guest (*userid*) NIC is coupled to the virtual switch. The choice of a port type is based on whether the target guest for this connection is VLAN-aware (the guest can process inbound tagged frames and send tagged frames) or VLAN-unaware (the guest is unable to handle tagged frames):

Access port
This type of connection is reserved for guests that are VLAN-unaware. A guest that is connected to an access port must be configured as a member of only one VLAN. The VLAN ID may be specified using the SET VSWITCH command, the NICDEF directory statement, the ESM interface, or it may default to the virtual switch's default VLAN ID. This VLAN ID becomes the Port VLAN ID (pvid). For this port type, when the virtual switch receives data from the guest (ingress), it tags the packet or frame with this pvid. When the virtual switch delivers data (egress), it strips the tag before delivering the packet or frame to the guest.

Trunk port
This type of connection is designated for those guests that are VLAN-aware. VLAN membership is determined for these connections based on the access list that was granted to the guest or port. A trunk port that is not configured (granted) with any VLAN memberships is a member of the virtual switch's default VLAN ID. This VLAN ID becomes the Port VLAN ID (pvid). For this port type, when the virtual switch receives data from the guest (ingress), it tags the packet or frame with the pvid. When the virtual switch delivers data (egress), it does not strip the tag before delivering the packet or frame to the guest. Trunk ports with explicit VLAN membership are not automatically members of the untagged set (native VLAN ID) and must be explicitly authorized through the SET VSWITCH command, the NICDEF statement, or an ESM to add the switch's native VLAN ID to the guest's access list. The OSA adapter, Global VLAN ID, connection to the virtual switch is a trunk port and is automatically configured to be a member of the untagged set. This provides the virtual switch with the ability to transmit and receive untagged frames.

If you specify VLAN AWARE NATIVE NONE, then it is recommended that you specify the VLAN ID on the NICDEFs. This configuration insulates the guest from any VLAN ID changes if the networks are moved or if there are domain relocation considerations. This configuration also ensures that if you forget to specify a VLAN ID, then you will get an error when the NICDEF tries to couple to the VSWITCH. In a software-defined network scenario, you should not depend on the VSWITCH default settings. You should explicitly specify which VLAN each vNIC is supposed to connect to (there is a 1-to-1 mapping between subnet and VLAN ID).

The QUERY VSWITCH command issued with the ACCESSLIST or PORTNUMBER option shows the authorizations and VLAN assignments known to CP. When the virtual switch is protected by an External Security Manager (ESM) the ESM interfaces must be used to determine the true authorization list. This list represents the VLAN LAN segments that a guest or port has been authorized to use to send and receive packets or frames. It may include inactive ports (pre-configured by a VSWITCH GRANT or PORTNUMBER) and it includes active ports whether they were configured by VSWITCH, NICDEF, or an ESM. A port defined by a NICDEF directory statement or an ESM will only be shown in the access list when the virtual machine is logged on and the NIC is coupled to the VSwitch. A VLAN-aware guest's link initialization of a virtual NIC that is coupled to a virtual switch must be in agreement with the VLAN authorization that has been granted for its connection. For example, if a guest or port has been granted access to only VLAN 12 on the virtual switch, this guest or port is permitted to receive and send datagrams on VLAN 12. The virtual switch ignores any attempt by the guest TCP/IP stack to configure the virtual NIC connection for a VLAN other than what is authorized. A mismatch of VLAN ID assignments between the virtual switch and the guest stack results in a loss of connectivity to the desired virtual LAN segment. Use the QUERY VSWITCH command with the DETAILS option to view VLAN ID conflicts.

Virtual Switch Ingress Rules

These ingress rules describe how the virtual switch processes data received on a port (inbound data):

- **When a guest virtual machine is connected through an access port (does not support VLAN tagging of frames):**
  - If the guest sends an untagged frame, the frame is associated with the pvid for this port. If the pvid is the same as the virtual switch's native VLAN ID, then the frame is transferred untagged. Otherwise, the frame is transferred tagged with the pvid assigned to this port.
  - If the guest sends a tagged frame, the VLAN ID in the tag must match the VLAN ID authorized for this guest:
    - If the guest has been assigned a pvid, and the VLAN ID in the tag matches that pvid, then the frame is sent to the destination. If it does not match, the frame is discarded.
    - If the guest is authorized only for the native VLAN ID, and the VLAN ID in the tag matches the virtual switch's native VLAN ID, then the tag is stripped from the frame and the frame is transferred untagged.
  - Packets or frames that violate these ingress rules are dropped (discarded).
- **When a guest virtual machine is connected through a trunk port (actively participates in VLAN tagging of frames):**
  - All frame tagging is performed by the guest.
  - If the guest has been granted access to the untagged set (the virtual switch's native VLAN ID), then untagged frames received from this guest are associated with the native VLAN and transferred untagged.

  Note: When a guest is granted access to a specific list of VLAN IDs, any conflicts with the access list result in inbound packets from that guest being dropped by the virtual switch. The QUERY VSWITCH command can be used to discover if there are any VLAN conflicts between the guest's VLAN configuration and the virtual switch's port access list.

Virtual Switch Egress Rules

These egress rules describe how the virtual switch delivers data to a port (outbound data):

- **When a guest virtual machine is connected through an access port (does not support VLAN tagging of frames):**
  - If the guest is explicitly granted access to a VLAN ID, and a frame tagged with this default VLAN ID (pvid) is destined for this guest's IP or MAC address, the frame is delivered to the guest untagged. The VLAN tag is stripped prior to delivery of all frames on an access port.
  - If the guest is not explicitly granted access to a VLAN ID, it means the guest is implicitly granted access to the virtual switch's default VLAN ID (pvid).
- **When a guest virtual machine is connected through a trunk port (actively participates in VLAN tagging of frames):**
  - If a tagged frame destined for the guest's IP or MAC address contains a VLAN ID that is in the guest's virtual machine access list, the frame is delivered to the guest with tagging intact.
  - If the guest has been granted access to the untagged set (the native VLAN ID), an untagged frame destined for this guest's IP or MAC address is delivered to the guest untagged.
  - If the guest is not explicitly granted access to a VLAN ID, the trunk port is granted access to the default VLAN ID (pvid)

Using the Same IP Address in Two Different VLANs

In general, you should avoid having two hosts with the same IP Address. There are two options for supporting this configuration with guest LAN or VSWITCH:

1. IP layer networks

   The OSA IP Layer implementation isolates specific network frames by VLAN tag, but does not isolate IP Address registration by VLAN group. Therefore, you must create two guest LAN segments, or two VSWITCH segments using two different OSA adapters, to allow two different hosts to register the same IP Address. Two guest LAN segments would automatically be isolated (unless a virtual router is deployed) and two VSWITCH segments would be isolated if the external switch hardware configures a unique VID set for each physical OSA adapter connection.

2. Link layer networks (layer2)

   The OSA Layer 2 implementation allows the hosts to manage IP addresses and ARP cache, so it is possible to have a single guest LAN segment (or VSWITCH segment) where two different hosts use the same IP Address on different VLAN groups. The network must be defined with the ETHERNET option (instead of the default IP Layer options), and the virtual hosts must be configured to use layer2 (instead of the default). The ARP traffic for a given VLAN group will only be visible to authorized members of that VLAN group. Note, however, that in order to support this configuration, neither host can be a member of the VLAN group assigned to the other. Furthermore, all switches and routers in the broadcast domain must be configured to avoid forwarding network frames from one VLAN to the other.

Virtual Switch Guest Configuration Considerations in IP Mode

OSA-Express and Network Express adapters enable VLAN-unaware TCP/IP stacks to participate in VLAN-segmented networks by tagging outbound frames and stripping VLAN tags from inbound frames. This is achieved through the configuration of a Global VLAN ID at the host level, which allows the adapter to manage VLAN tagging transparently. In z/VM environments, virtual NICs simulate this behavior. VLAN operations are handled within the virtual switch, which simplifies host configuration by avoiding VLAN settings in the TCP/IP stack. Instead, guests are connected to access ports with assigned port VLAN IDs (PVIDs).

For z/OS systems that run multiple TCP/IP stacks under a single virtual OSA TRLE, a trunk port is required to support multiple LANs. Each stack is assigned a unique Global VLAN ID.

Additionally, PriRouter support allows one host per VLAN to be designated as the primary router, based on its VLAN ID configuration. If no VLAN-specific PriRouter is defined, non-VLAN traffic is routed through a default PriRouter. This architecture ensures efficient VLAN management and routing in virtualized mainframe environments.

OSA Adapter Support for Host VLANs

The OSA adapters support TCP/IP stacks that are VLAN-unaware by providing a mechanism to tag outbound frames from the host and strip the VLAN tag from inbound frames before delivery to the host (only IP connections). This support provides the ability for a VLAN-unaware host to be a member of a VLAN LAN segment. Through its link configuration method, a host device driver can register a Global VLAN ID with the network adapter. The OSA adapter uses the Global VLAN ID to tag frames and send out Gratuitous ARP requests (ARP requests to check for duplicate IP addresses) on behalf of the host. Each host can define only one Global VLAN ID per connection. The NIC simulation in z/VM also provides this support, which is separate from the virtual switch support. The Global VLAN ID processing for the virtual NIC is performed prior to any virtual switch port ingress processing and after virtual switch port egress processing. To reduce complexity and host TCP/IP configuration changes, it is recommended that you do not configure link VLAN IDs in your TCP/IP stacks. Instead, connect these guests to access ports on the virtual switch and authorize each guest for the desired VLAN. This will assign a port VLAN ID (pvid) for each port, and all VLAN tag operations will occur within the virtual switch.

z/OS Support

Deploying multiple z/OS TCP/IP stacks under the same virtual OSA TRLE on a virtual switch requires a trunk port connection. All z/OS stacks are connected through the same virtual NIC. Each stack in this group is configured with a Global VLAN ID through its TCP/IP stack configuration. To support multiple VLAN LAN segments across a single NIC, a trunk port connection must be deployed. In this way, each required VLAN ID can be granted for this connection (a VLAN ID associated with each stack in the TRLE group). Even though these stacks are VLAN-unaware, the tagging performed in the IP layer does not cause a problem for the guest.

Using PriRouters and VLAN

The PriRouter support allows for the assignment of a single PriRouter per VLAN LAN segment. The designation of a host as a PriRouter for a given VLAN LAN segment requires that the host be configured with a Global VLAN ID for the VLAN segment that will be providing the routing. This provides the virtual switch with the necessary means to make the PriRouter assignment to this guest. Only one PriRouter per VLAN is supported. You can configure a PriRouter for non-VLAN traffic and concurrently configure a PriRouter for each VLAN LAN segment. In the absence of a VLAN ID PriRouter, all unknown VLAN traffic is routed to the host with the non-VLAN PriRouter assignment.

Servicing Virtual Switch OSA adapters

If you need to take an OSA adapter out of a virtual switch configuration in order to apply service, the following steps are recommended.

- For this example, assume that virtual switch LINXNET1 is defined to use port group LINXGRP1. LINXGRP1 is defined to use devices A000 and B000. B000 is in need of service.
  1. Issue "SET PORT GROUP LINXGRP1 LEAVE B000" to remove device B000 from the port group configuration.
  2. Apply service to B000.
  3. Issue "SET PORT GROUP LINXGRP1 JOIN B000" to return device B000 to the port group configuration.
- For this example, assume that virtual switch LINXNET2 is defined without a port group. Instead, device D000 is the active connection and E000 is a backup device.
  1. Issue "SET VSWITCH LINXNET2 UPLINK SWITCHOVER RDEV E000" to shut down the network connection on D000 and activate a new network connection on device E000.
  2. Apply service to D000.
  3. Optionally issue "SET VSWITCH LINXNET2 UPLINK SWITCHOVER RDEV D000" to return device E000 to the virtual switch configuration as a backup device and restore the network connection back on D000.

Virtual Switch Failover

Failover support for virtual switches provides recovery for virtual switch controller failures and for OSA adapter (OSA-Express and Network Express) failures.

Failover of VSwitch with OSA-Express devices

For OSA-Express devices, which require a virtual switch controller, z/VM pairs virtual switch controllers and initializes the backup device as much as possible. To reduce the time necessary to recover from the failure of a virtual switch's network connection (including the loss of the entire port group), z/VM pre-initializes the OSA-Express device and the virtual switch controller pairs for backup support. When multiple controllers and backup devices are available, a failure of either component allows quick failover to the previously-initialized backups. When a failure happens for a virtual switch that has pre-initialized backup RDEVs, the virtual switch is put into a Recovering state, and it stays in that state until initialization of the backup completes. See Figure 21 on page 68 for an example of this pairing.

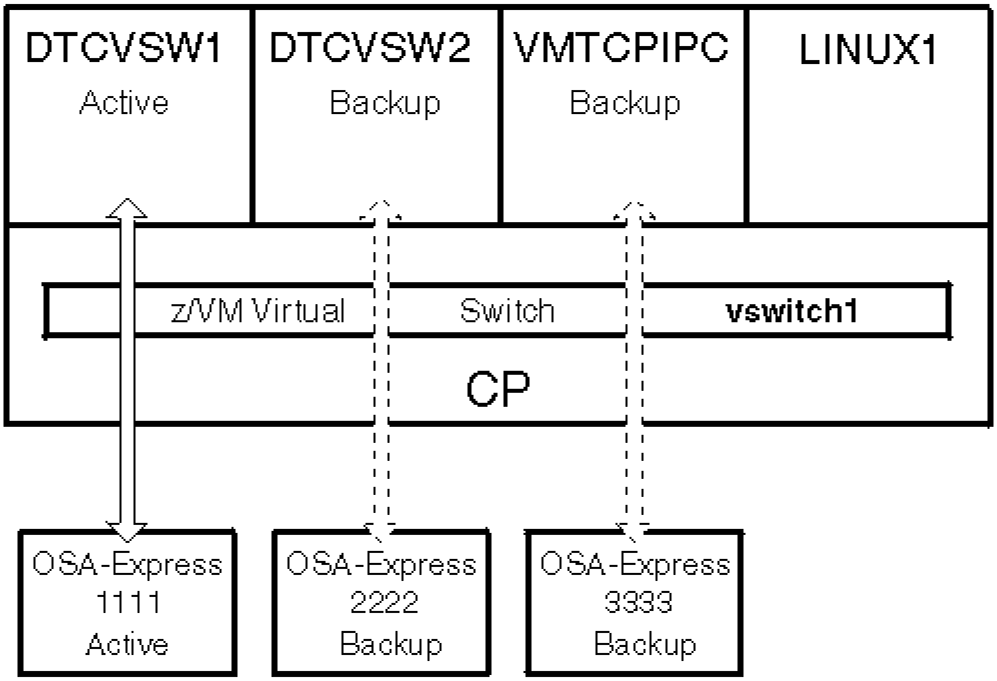


Figure 21. Failover of a Virtual Switch that Uses OSA-Express Devices

Figure 21 on page 68 shows virtual switch vswitch1 defined with three OSA-Express devices, using RDEV 1111 2222 3333, on the DEFINE VSWITCH command. The virtual switch is also defined with the CONTROLLER * option. This allows the virtual switch controller functions to be spread across multiple z/VM virtual machines. Because there are three virtual switch controller machines available, initialization for vswitch1 spreads the RDEVs across the virtual switch controllers – OSA-Express device 1111 is attached to DTCVSW1 and fully initialized, OSA-Express device 2222 is attached to DTCVSW2 and initialized as a backup for vswitch1, and OSA-Express device 3333 is attached to VMTCPIPC and initialized as a backup for vswitch1.

If a failure occurs for the active OSA-Express device or for the virtual switch controller, then data transfer to the network is suspended during recovery from the failure. Some transmissions that are bound for guests that are coupled to the virtual switch might be lost while their IP addresses are unknown on the real LAN segment. After the recovery is complete, the remaining initialization is done for OSA-Express device 2222, and normal virtual switch operation continues. Data transfer continues without interruption among guests in virtual machines that are coupled to vswitch1.

When a virtual switch is defined with the CONTROLLER * option or a list of controllers, z/VM allows the controller functions to be spread across multiple z/VM virtual machines. If there are fewer virtual switch controllers than devices, ownership of the devices is spread across the available virtual machines, with some controllers handling more than one device. The CONTROLLER option is ignored when the virtual switch manages EQDIO devices to connect the real hardware LAN segment.

Issue the QUERY VSWITCH command to see the OSA-Express devices and virtual switch controllers. Issue the QUERY CONTROLLER command to display information about the z/VM TCP/IP virtual machines that are used to manage OSA-Express devices that are associated with the virtual switch. The output shows if a virtual switch controller is active or if it is a backup for a virtual switch.

You can reassign backup devices. Use the SET VSWITCH RDEV command while a virtual switch is connected to allow the addition or removal of backup RDEVs from a virtual switch.

Note: Use the SET VSWITCH UPLINK SWITCHOVER option to manually initiate a controlled port change or failover from the currently active OSA-Express port to another configured backup port with minimal network disruption.

Failover of VSwitch with Network Express devices

The process of failover of a virtual switch that uses a Network Express device is similar to failover of a virtual switch that uses an OSA-Express device. The significant difference is that a virtual switch that uses a Network Express device does not use a virtual switch controller machine. For Network Express (EQDIO) devices, controller functions are managed within the virtual switch. z/VM pre-initializes Network Express devices to allow quick failover to the initialized backup devices. See Figure 22 on page 69 for an example of failover with Network Express devices. In this example, Network Express devices 2222 and 3333 are connected to virtual switch vswitch1 and are backups for device 1111.

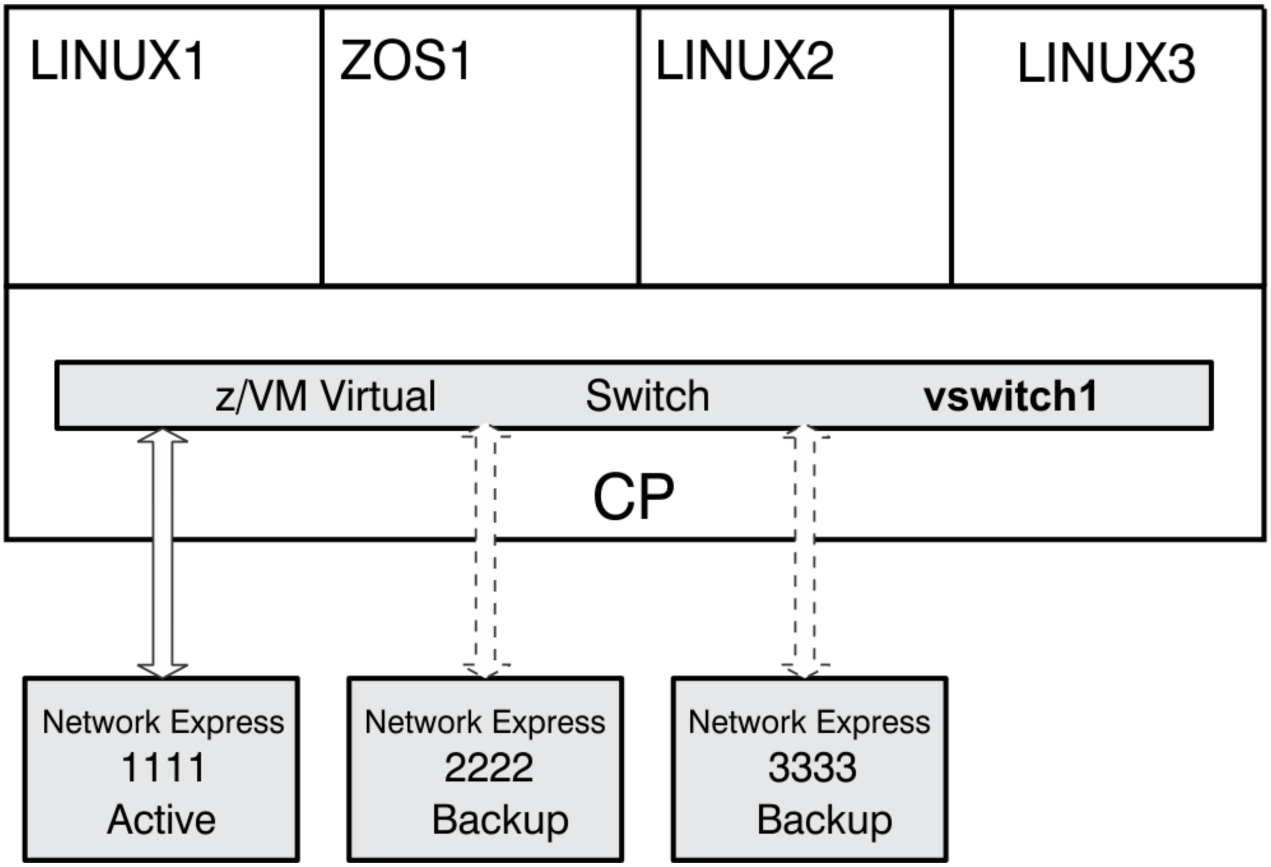

LINUX1
ZOS1
LINUX2
LINUX3
z/VM Virtual Switch vswitch1
CP
Network Express 1111 Active
Network Express 2222 Backup
Network Express 3333 Backup

Figure 22. Failover of a Virtual Switch that Uses Network Express Devices

For more information, see the following CP command and configuration topics:

- DEFINE VSWITCH Statement in *z/VM: CP Planning and Administration*
- QUERY CONTROLLER in *z/VM: CP Commands and Utilities Reference*
- QUERY VSWITCH in *z/VM: CP Commands and Utilities Reference*
- SET VSWITCH in *z/VM: CP Commands and Utilities Reference*

OSA Adapter Hang Detection

z/VM uses a timestamp and a full buffer condition to recognize when a real device that is associated with a virtual switch is stalled. The associated device can be an OSA-Express adapter or a Network Express adapter. When CP determines that the buffers that are associated with the device are full and a timeout value passes with no relief, an error message is issued and an attempt is made to fail the virtual switch over to a backup device. This causes the VSwitch controller to start cleanup of the device, and error message HCP2832E is issued.

z/VM also determines when a simulated device that is associated with a guest LAN is stalled, but the Control Program's action is slightly different. A guest LAN uses a virtual NIC that is defined as a simulated OSA-Express adapter. When CP determines that the buffers that are associated with the simulated

OSA-Express adapter are full and a timeout value passes with no relief, CP assumes that the guest LAN is hung. CP disables the device and issues an error message.

Conventional I/O that is issued to an OSA-Express or HiperSockets read, write and data device can also be protected by using z/VM's Missing Interruption Handler (MIH). This protection is automatically enabled or disabled appropriately. MIH is used to detect and recover (failover) network connectivity that is caused by unresponsive or broken hardware.

Note: MIH does not apply to Network Express adapters.

VSwitch Queue Hang Detection

The z/VM Control Program (CP) detects hung virtual switches and can start error recovery.

QDIO VSwitch Controller Hang Detection

A virtual switch that uses an OSA-Express device requires a virtual switch controller machine. The z/VM Control Program (CP) controller component uses time-out checks to check the VSwitch controllers. If a response to a scheduled task is not received within a certain time limit, error recovery is kicked off. There is one timestamp for each VSwitch controller that is managing the OSA-Express device associated with one or more virtual switches. When this condition is detected, error recovery is started and an error message is issued.

VSwitch controllers that are backups for virtual switches are also checked. However, a detected hang on one of these backup VSwitch controllers results in only a warning message.

EQDIO VSwitch Queue Hang Detection

A virtual switch that uses a Network Express device does not use a virtual switch controller machine. For Network Express (EQDIO) devices, controller functions are managed within the virtual switch. The CP controller component uses time-out checks to detect a hung queue. If a response to a scheduled task is not received within a certain time limit, error recovery is kicked off. There is one timestamp for each EQDIO queue that is managed by the virtual switch for the Network Express device. When a hung queue is detected, error recovery is started and an error message is issued.

Virtual Switch Topology Considerations

The virtual switch is optimized to provide an efficient and low cost (CPU utilization) method of connecting a guest LAN to an external network. This optimization is based on a flat network topology. A flat topology from a guest LAN perspective is one that does not require a router. Every virtual machine port that is serviced by the virtual switch has an adapter coupled directly to the virtual switch's LAN. Router virtual machine ports provide connectivity for remote guest LANs through IP routing/forwarding which increases the number of times a datagram must be copied and adds additional routing decisions.

If multiple LAN segments are required ”inside” the virtual switch with connectivity to the external network, merge the nodes from the remote LAN segments into the virtual switch LAN segment and use VLAN groups to segment a single LAN (guest LAN) into smaller LAN segments. This requires support of VLANs in the physical LAN network topology. See Figure 23 on page 71 for an example.

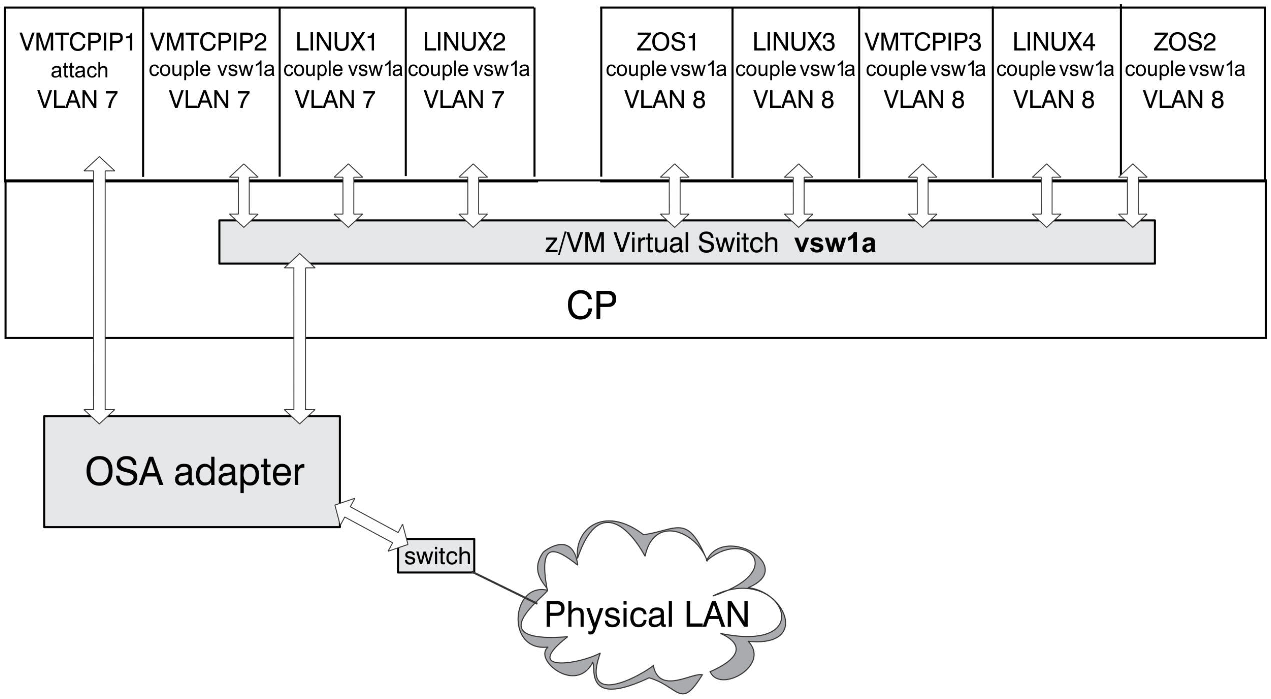


Figure 23. Multiple VLAN Environment GIF

Virtual Machine Routing for QDIO

In the case where a router virtual machine port must be deployed on a virtual switch, the following configuration considerations must be taken into account:

- The following considerations are required to insure that inbound datagrams are delivered to hosts that reside on the remote guest LANs (see Figure 24 on page 72):
  - The router virtual machine port must be configured as a PRIROUTER.
  - The virtual switch that services the guest LAN with a router virtual machine port must also have its OSA-Express connection configured as PRIROUTER.
- This topology will cause additional flooding of IP traffic within the virtual switch, which can result in higher CPU utilization costs.
- OSA-Express supports one PRIROUTER per shared adapter, which limits this topology to a single virtual switch on each adapter. Multiple OSA-Express adapters are required if multiple instances of this configuration are required.

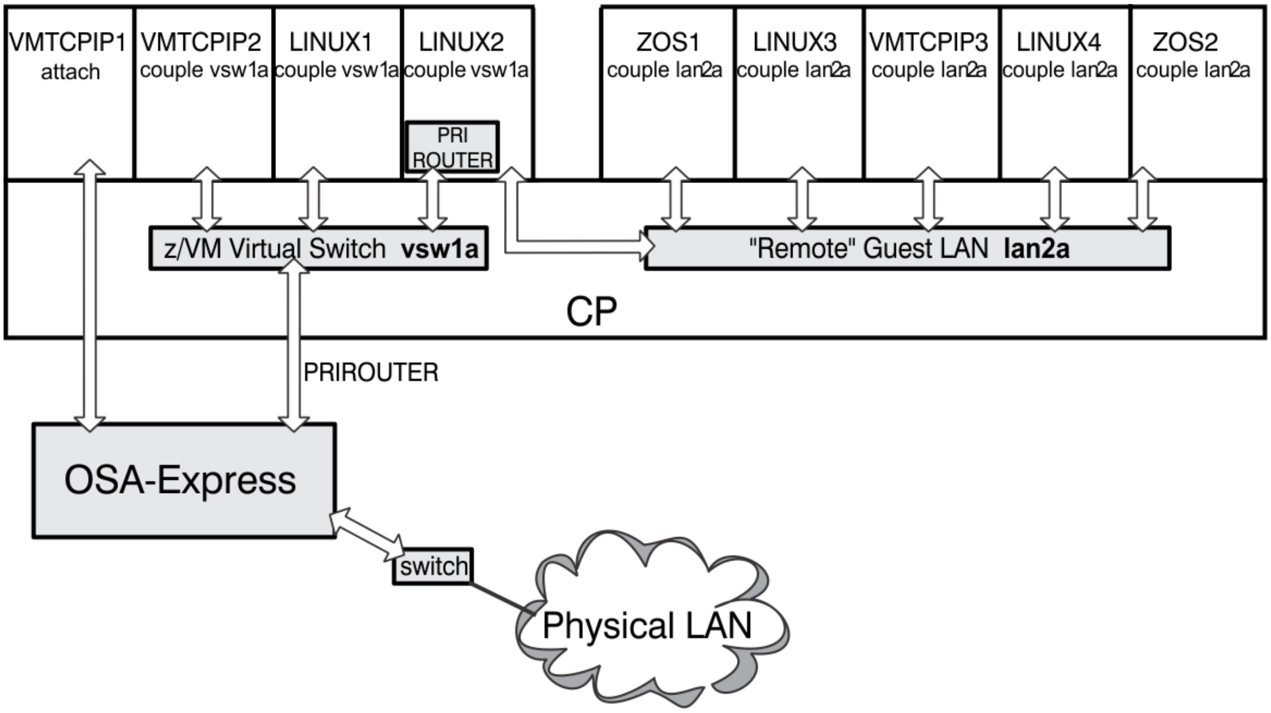


Figure 24. Virtual Switch with Remote Guest LAN

Restriction: A Network Express adapter does not support layer 3 transport or PRIROUTER and operates in EQDIO mode. This virtual machine routing information for QDIO applies only when an OSA-Express adapter is used.

Spanning Tree Protocol and the Virtual Switch

This section provides an overview of the Spanning Tree Protocol (STP) overview and the virtual switch.

- **Spanning Tree Protocol (STP) Overview**

  The Spanning Tree Protocol (STP) prevent loops while still allowing for redundancy in a bridged network. Details for STP can be found in the IEEE 802.1d, 802.1q, and 802.1s specifications. Two potential problems with loops in a bridged network are broadcast loops and bridge table corruption. Spanning Tree Protocol dictates which path to take through a looped network in order to avoid these problems. STP enabled bridges communicate with each other to discover redundant paths in the network. This discovery results in a spanning-tree which defines a single path to a given destination and alternate paths in the event of a network outage or configuration change.

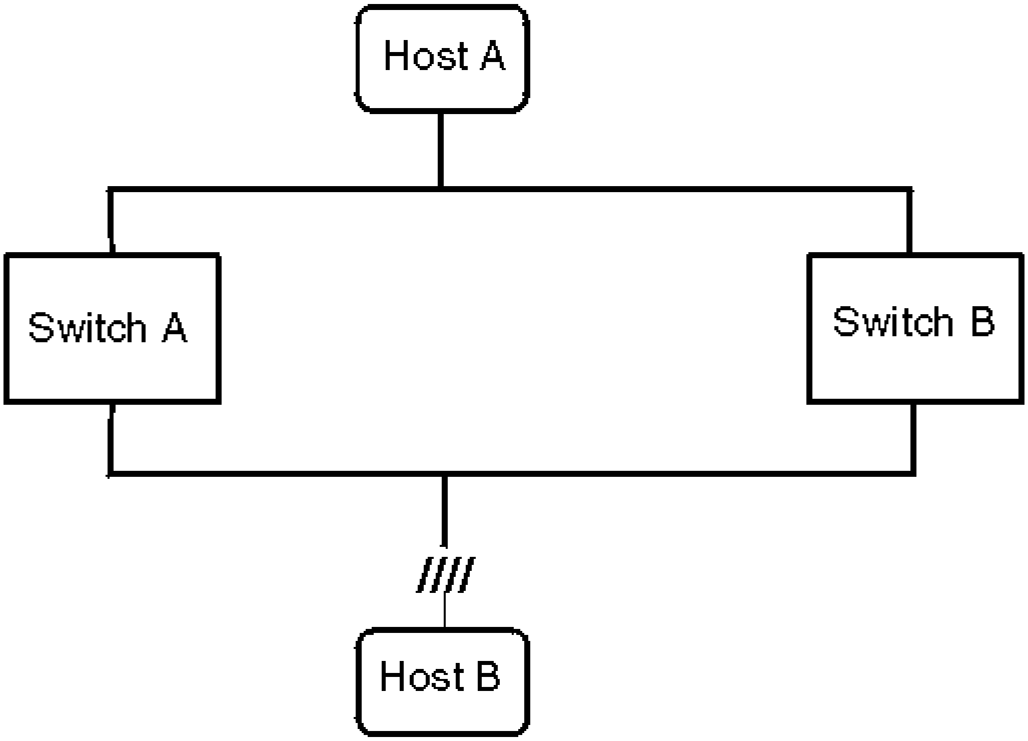


Figure 25. Redundant network example

Broadcast loops or **Broadcast storms** cause broadcast packets to be forwarded on repeatedly. Refer to Figure 25 on page 73, switch A is in a loop connected to switch B. A broadcast comes from a host A. Switch A forwards the broadcast to switch B. In turn, Switch B forwards the broadcast out a different port back to Switch A. Switch A repeats its initial action and a broadcast loop is created.

Bridge table corruption can occur on a layer 2 network when ARP frames get circulated in a loop similar to the broadcast loop causing the ARP table entries to become corrupted. Refer to Figure 25 on page 73, assume host A has previously sent traffic across switch A. Therefore, switch A has an entry for host A's MAC address. Host A attempts to find host B but host B is unavailable. Host A has access to switch A and switch B. An ARP packet is sent to both switches for host B. Switch B doesn't know where host B is and forwards the packet on (similarly switch A does not know where host B is and forwards the packet creating a feedback loop). Switch A receives the ARP from switch B on a different port (assume port 2) then it received the original ARP on (assume port 1). Switch A still does not know where host B is but it does know that ARPs from host A come in on port 2. Switch A updates its bridge table to change the entry for Host A from port 1 to port 2. Now there is a feedback loop in the network and the bridge table has been corrupted in the switch.

- **How does this affect the Virtual Switch?**

  The virtual switch (VSWITCH) does not participate in the Spanning Tree Protocol. The VSWITCH can be thought of as an edge switch. The VSWITCH connects the z/VM host LAN segment to the external network. This configuration typically does not have redundant paths from each host. Due to the nature of z/VM virtual networks, redundant paths (loops) can be easily remedied through configuration changes. Avoiding dual path situations caused by routing hosts forwarding broadcasts in a virtual switched network is the key to preventing broadcast storms. Redundancy is rarely required because recovery of network outages on OSA adapter ports is transparently provided by the VSWITCH. Some suggestions:

  - Problems can arise if a guest on the VSWITCH has a separate connection to the external network on the same subnet. To avoid this, guest's connections to the external network should be through the VSWITCH only. If a need arises to keep some traffic isolated from other guests on the virtual switch, VLANs should be used.
  - Configuration problems with a link aggregation group on the physical switch could also lead to problems if spanning tree protocol is not in place on the physical switch. Symptoms would include

forwarding multiple copies of packets into the VSWITCH network. However, the virtual system implementation avoids aggravating the problem eliminating loop back from the link aggregation port. Packets that have been received on the link aggregation port are not forwarded back onto the physical network.

– When a z/VM host is acting as a router over to separate VSWITCHs (two separate LAN segments), you should ensure that the router is not configured to forward broadcasts from one LAN segment to the other.

Configuring a Virtual Switch

To use a virtual switch, do the following tasks:

1. Create a virtual switch to act as the network between virtual machines.
2. Configure each virtual machine to access the virtual switch.
3. Optionally configure an SNMP Subagent for a virtual switch
4. Create a simulated network interface card (NIC) on each virtual machine to be connected to the virtual switch.
5. Couple each virtual machine's simulated NIC to the virtual switch.
6. Configure the network interface for the virtual switch.

These tasks are described in the following sections.

Creating a Virtual Switch

Use the DEFINE VSWITCH command to define a virtual switch. The command syntax is:

```
DEFINE VSWITCH switchname [ operands ]
```

where:

switchname
is the name of the new virtual switch. The *switchname* is a single token (1–8 alphanumeric characters) that identifies this virtual switch for subsequent commands.

operands
define the characteristics of the virtual switch.

When creating your VSWITCH, you need to consider:

- Mode of operation

  Note:

  – A virtual switch with an OSA-Express (QDIO) adapter supports IP or Ethernet.

  – A virtual switch with a Network Express (EQDIO) adapter supports only Ethernet.
- VLAN connectivity
- Link Aggregation
- User or Port based live guest relocation strategy.

When an Ethernet (Layer2) VSWITCH is to be deployed, you need to determine if the default MAC address assignment performed by z/VM is sufficient for your network. If your VSWITCH is to be connected to a network (switch) that is deploying VLANs, then you need to decide whether the VSWITCH will operate as VLAN-aware or VLAN-unaware. If your VSWITCH will be VLAN-aware, you also need to determine the native VLAN ID and the PORTTYPE and VLAN assignments for each guest that will be connected to the VSWITCH. For more information on these topics, see “Additional Considerations for Virtual Switch Configurations” on page 54.

For more information on DEFINE VSWITCH operands, see DEFINE VSWITCH in *z/VM: CP Commands and Utilities Reference*.

Configuring Access to a Virtual Switch

All virtual switches are restricted in their access. That is, users must be authorized in order to couple to the virtual switch. With Directory Network Authorization (DNA) enabled on the system, authorization can all be done centrally within the user directory. Within the system directory, the NICDEF LAN statement allows this device to connect to the virtual switch:

```
NICDEF vdevno ... LAN SYSTEM switchname
```

NIC characteristics may also be included on the NICDEF statement such as PORTTYPE and VLAN operands.

Alternatively, for user based management, use the SET VSWITCH command with the GRANT operand to authorize z/VM users to couple to the switch. The command syntax is:

```
SET VSWITCH switchname GRANT userid
```

As an alternative, access can be managed by a specific port using the SET VSWITCH command with the PORTNUMBER operand to define ports. The PORTNUMber operand defines a port for a virtual switch and associates the specified userid with a virtual switch port portnum. The userid will be added to the access list for this virtual switch. This allows the userid to connect an adapter to the port defined on the virtual switch. The command syntax is:

```
SET VSWITCH switchname PORTNUMBER portnum USERID userid
```

In addition, if the default port type or native VLAN ID of the VLAN-aware virtual switch is not sufficient, specify the PORTTYPE and VLAN options with the NICDEF, GRANT or PORTNUMBER operands. An ESM can also be used to control access and VLAN authorization for a virtual switch.

Configuring an SNMP Subagent for a Virtual Switch

Add the new VSWITCH *switchname* keyword to a **HOME** statement in a z/VM TCP/IP stack to define an interface which can be used by SNMP for communicating network management information for the virtual switch.

- Select a z/VM TCP/IP stack that has the SNMP service and SNMP subagent. It is recommended that the network connection for the stack providing connectivity to the SNMP service from the Network Management System (NMS) is separate from the managed virtual switches, so that a failure of a virtual switch connection can be reported to the NMS using SNMP traps.
- Update the TCP/IP configuration file for that stack, defining DEVICE, LINK, START, and HOME statements to provide an interface that can be reached by the NMS. Use the **VSWITCH** *switchname* option on the HOME statement to identify the virtual switch name.
- Update the TRAP destination file (SNMPTRAP DEST) to define a list of managers to which LINK_UP and LINK_DOWN traps are to be sent.
- Ensure that an SNMP subagent has been configured to provide VSWITCH MIB data. For information on setting up an SNMP subagent, refer to Configuring the SNMP Servers in the *z/VM: TCP/IP Planning and Customization*.

Creating a Simulated NIC

Use the DEFINE NIC command to create a simulated NIC. The command syntax is:

```
DEFine NIC vdev [ operands ]
```

where:

vdev
specifies the base virtual device address for the NIC.

operands
define the characteristics of the simulated NIC.

For more information on DEFINE NIC operands, see DEFINE NIC in *z/VM: CP Commands and Utilities Reference*.

Restriction: Only QDIO simulated NICs can be defined to couple to a virtual switch. EQDIO simulated NICs are not supported.

Coupling the Simulated NIC to the Virtual Switch

Use the COUPLE command to couple a simulated NIC to a virtual switch. The command syntax is:

```
COUPLE vdev TO SYSTEM switchname [PORTNUMBER portnum]
```

where:

vdev
is the base virtual device address for the NIC.

SYSTEM *switchname*
identifies a virtual switch as the target of the connection.

PORTNUMBER *portnum*
identifies the port number in the user-defined range of 1 to 2048 for the connection. This option is only allowed for a user-defined port. COUPLE by PORTNUMBER is not permitted for a port which was automatically assigned by z/VM (outside of the user-defined port range).

Restriction: Only QDIO simulated NICs can be defined to couple to a virtual switch. EQDIO simulated NICs are not supported.

Configuring the Network Interface

Once the virtual switch is defined and a simulated NIC is coupled to that virtual switch in the virtual machine, the network interface can then be configured and enabled. The steps to do this are different depending on whether you are running z/VM TCP/IP in the guest or another operating system such as Linux or z/OS. See Configuring the TCP/IP Servers in *z/VM: TCP/IP Planning and Customization* for information about configuring TCP/IP.

Example — Dynamically Creating and Destroying a Virtual Switch

To create and then explicitly destroy a port based virtual switch, do the following. Note: This example was done from the LINUX1 userid with Class B privilege.

1. Define a port based virtual switch named PORTVSW.

   ```
   DEFINE VSWITCH PORTVSW PORTBASED
   ```

2. Define a port for LINUX1 userid. (Authorizes LINUX1 to connect to port 10).

   ```
   SET VSWITCH PORTVSW PORTNUMBER 10 USERID LINUX1
   ```

3. Define a simulated NIC of type QDIO. The virtual device number is 0800.

   ```
   DEFINE NIC 0800 QDIO
   ```

4. Couple the simulated NIC to the virtual switch on the defined port.

   ```
   COUPLE 0800 TO SYSTEM PORTVSW PORTNUMBER 10
   ```

5. Explicitly destroy the virtual switch.

   ```
   DETACH VSWITCH PORTVSW
   ```

At the end of this process, the simulated NIC at devices 0800-0802 still exists, and can be coupled to another virtual switch. The command DETACH NIC is used to destroy the simulated NIC. To do this, enter:

```
DETACH NIC 0800
```

The command DETACH 0800 cannot be used to detach a simulated NIC. You must issue DETACH NIC when you want to remove a simulated NIC.

Virtual Switch Attributes

Viewing Virtual Switch Attributes

You can view the attributes of a virtual switch using the QUERY VSWITCH command. The syntax to get a detailed view of a specific virtual switch is:

```
QUERY VSWITCH switchname DETAILS
```

To get a list of the ports that are authorized (by z/VM) to connect to the virtual switch (including all active ports regardless of how they were authorized), use the ACCESSLIST or PORTNUMBER operand:

```
QUERY VSWITCH switchname ACCESSLIST
```

or:

```
QUERY VSWITCH switchname PORTNUMBER
```

Note: A port defined by a NICDEF directory statement or an ESM will only be shown in the access list when the virtual machine is logged on and the NIC is coupled to the VSwitch.

You can get a detailed report on all virtual switches. To do this, enter:

```
QUERY VSWITCH ALL DETAILS
```

If *switchname* is omitted from the QUERY VSWITCH command, the default is to provide information about all virtual switches.

Changing Virtual Switch Attributes

You can modify the attributes of a virtual switch using the command SET VSWITCH. Some attributes may only be changed when the virtual switch is in the disconnected state. For more information on these attributes, see SET VSWITCH in *z/VM: CP Commands and Utilities Reference*.

Automating Virtual Switch Configuration on IPL

Because virtual switches are not maintained across z/VM IPL, the following is a procedure to automate configuration of the virtual switch on IPL.

Define the Virtual Switch in the SYSTEM CONFIG file

Add the DEFINE VSWITCH statement to the SYSTEM CONFIG to ensure the virtual switch is defined during IPL. See Configuring Your System in *z/VM: CP Planning and Administration* for details on the SYSTEM CONFIG file.

Define and Couple Simulated NICs to the Virtual Switch

Using the NICDEF statement, you can define, authorize, configure and couple a simulated NIC to the Virtual Switch created during IPL. See NICDEF Directory Statement in *z/VM: CP Planning and Administration* for additional information.

The following is an example to centralize a VSwitch configuration:

1. Define the virtual switch in SYSTEM CONFIG and restart the system (or use the CP DEFINE VSWITCH command):

```
DEFINE VSWITCH PORTVSW PORTBASED VLAN AWARE
```

2. Configure the virtual NIC in USER DIRECT within the USER clause of LINUX1 and use DIRECTXA to generate a new object directory:

```
NICDEF 1200 TYPE QDIO LAN SYSTEM PORTVSW PORTTYPE ACCESS VLAN 202
```

Configuring a Guest LAN

To use a guest LAN, do the following tasks:

1. Create a guest LAN segment to act as the network between virtual machines.
2. Create a simulated network interface card (NIC) on each virtual machine to be connected to the guest LAN.
3. Couple each virtual machine's simulated NIC to the guest LAN segment.
4. Configure the network interface for the guest LAN.

These tasks are described in the following sections.

Creating a Guest LAN segment

Use the DEFINE LAN command to define a guest LAN segment. The command syntax is:

```
DEFINE LAN lanname [ operands ]
```

where:

lanname
is a 1 to 8 character alphanumeric name for the guest LAN segment.

operands
define the characteristics of the guest LAN.

When creating your guest LAN, you need to consider the mode of operation: IP or Ethernet. When an Ethernet guest LAN is to be created, you need to determine if the default MAC address assignment performed by z/VM is correct for your network.

For more information about the IP and Ethernet options, and about other DEFINE LAN operands, see DEFINE LAN in *z/VM: CP Commands and Utilities Reference*.

Establishing a Guest LAN Owner

The OWNERid operand determines both the owner of the LAN and the lifetime of the LAN. The OWNERid may assume values:

*****
the owner is invoker of the DEFINE LAN command (this is the default).

ownerid
a z/VM user that owns the LAN.

SYSTEM
the LAN is to be a persistent LAN.

Both the LAN name and its owner uniquely identify a guest LAN (this means you may define two distinct LANs, provided they are owned by different z/VM users).

Establishing a Guest LAN Lifetime

The lifetime of a guest LAN is determined by its owner. When assigned a z/VM user as an owner at creation, the LAN is defined to be transient. If assigned the special owner SYSTEM at creation, the LAN is defined to be persistent. If no owner is specified for the LAN, it is created as a transient LAN owned by the invoker.

Transient Guest LAN

A transient LAN exists as long as the owner is logged on. Once the owner logs off z/VM, a transient guest LAN is destroyed, provided no simulated NICs are coupled to the LAN (once all NICs are uncoupled, however, the LAN is destroyed).

Class G users can create a guest LAN segment. However, the owner must be specified as the invoker (thus, all LANs created by a class G user are transient).

Persistent Guest LAN

A persistent LAN can be destroyed only by using the DETACH LAN command. The DETACH LAN command can also be used to destroy transient LANs. A persistent guest LAN will not exist across a z/VM IPL. See "Automating Guest LAN Configuration on IPL" on page 81 for the procedure to define a guest LAN that is maintained across IPL.

Creating a Simulated NIC on Each Virtual Machine

Use the DEFINE NIC command to create a simulated NIC. The command syntax is:

```
DEFINE NIC vdev [ operands ]
```

where:

vdev
specifies the base virtual device address for the adapter.

operands
define the characteristics of the simulated NIC.

For more information about these operands, see DEFINE NIC in *z/VM: CP Commands and Utilities Reference*.

Restriction: Only QDIO simulated NICs can be defined to couple to a guest LAN. EQDIO simulated NICs are not supported.

Coupling the Simulated NIC to the Guest LAN

Use the COUPLE command to couple a simulated NIC to a guest LAN. The syntax of the COUPLE command is:

```
COUPLE vdev TO ownerid lanname
```

where:

vdev
specifies the base virtual device address for the adapter.

ownerid
specifies the owner of the guest LAN.

lanname
the name for the guest LAN.

You can couple a simulated NIC only to a guest LAN of the same type. For instance, if the guest LAN is defined to be of type QDIO, you can couple only simulated NICs of type QDIO to that LAN.

Restriction: Only QDIO simulated NICs can be defined to couple to a guest LAN. EQDIO simulated NICs are not supported.

Configuring the Network Interface for the Guest LAN

Once the guest LAN segment is defined and a simulated NIC is coupled to that LAN in the virtual machine, the network interface can then be configured and enabled. The steps to do this are different depending on whether you are running z/VM TCP/IP in the guest or another operating system such as Linux or z/OS.

See Configuring the TCP/IP Servers in *z/VM: TCP/IP Planning and Customization* for information about configuring z/VM TCP/IP.

Example — Creating and Destroying a QDIO Guest LAN

To create and then explicitly destroy a QDIO guest LAN, do the following:

1. Define a transient guest LAN of type QDIO.

   ```
   DEFINE LAN QDIOSAMP TYPE QDIO
   ```

2. Define a simulated NIC of type QDIO. The virtual device number is 0543.

   ```
   DEFINE NIC 0543 QDIO
   ```

3. Couple the simulated NIC to the guest LAN.

   ```
   COUPLE 0543 TO * QDIOSAMP
   ```

4. Explicitly destroy the guest LAN.

   ```
   DETACH LAN QDIOSAMP
   ```

At the end of this process, the simulated NIC at devices 0543-0545 still exists, and can be coupled to another guest LAN. The command DETACH NIC is used to destroy the simulated NIC. To do this, enter:

```
DETACH NIC 0543
```

The command DETACH 0543 cannot be used to detach a simulated NIC. You must issue DETACH NIC when you want to remove a simulated NIC.

Restriction: Only QDIO simulated NICs can be defined to couple to a guest LAN. EQDIO simulated NICs are not supported.

Guest LAN Attributes

The RESTricted operand on the DEFINE LAN command allows you to restrict z/VM users permitted to attach simulated NICs to the LAN. This security feature is enabled when the LAN is created. Use the SET LAN command to authorize z/VM users to attach to the LAN.

Viewing Guest LAN Attributes

You can view the attributes of a guest LAN using the QUERY LAN command. The syntax to get a detailed view of a specific guest LAN is:

```
QUERY LAN lanname DETAILS
```

You can get a detailed report on all guest LANs. To do this, enter:

```
QUERY LAN ALL DETAILS
```

If *lanname* is omitted, the default is a detailed report on all guest LANs.

Example — Detailed Report on Guest LAN

To request a detailed report on the guest LAN named QDIOSAMP, do the following:

```
QUERY LAN QDIOSAMP DETAILS

LAN LNX88    QDIOSAMP   Type: QDIO    Connected: 1    Maxconn: INFINITE
  TRANSIENT   UNRESTRICTED  IP                        Accounting: OFF
    Adapter Owner: LNX88    NIC: 0543  Name: UNASSIGNED
Ready;T=0.01/0.01 10:19:42
```

The guest LAN is uniquely identified by both the owner and the LAN name (LNX88 QDIOSAMP). It is of type QDIO with one active attached simulated NIC. No limit on the number of connections to the LAN is defined.

The guest LAN is transient and unrestricted, and the transport type is IP. Accounting is turned off. Although the MFS operand is valid only for HiperSocket LANs, type QDIO LANs have an effective MFS value of 8992.

An adapter owned by user LNX88 is attached to the LAN. The simulated NIC virtual device number is 0543, and it has no assigned port name (Name: UNASSIGNED).

Changing Guest LAN Attributes

You can modify the attributes of a guest LAN using the command SET LAN. Attributes that may be modified include:

- The LAN owner. Changing the owner from a z/VM user to special owner SYSTEM changes the LAN from transient to persistent.
- Accounting for the LAN (the ACCOUNTing operand of DEFINE LAN).
- The access list of authorized users for a restricted guest LAN.

The syntax to authorize a z/VM user to attach to a restricted LAN is:

```
SET LAN lanname GRANT userid
```

To revoke a z/VM user's authorization, use:

```
SET LAN lanname REVOKE userid
```

Example — Creating a Restricted Guest LAN of Type HiperSockets

To create a restricted guest LAN of type HiperSockets and modify its access list, do the following:

1. Create a restricted HiperSockets guest LAN named HIPRSAMP.

   ```
   DEFINE LAN HIPRSAMP OWNER * MAXCONN 4 MFS 64K REST
   ```

2. Query the LAN attributes. Note that the owner is the only authorized z/VM user.

   ```
   QUERY LAN HIPRSAMP DETAILS
   ```

3. Grant authorization to z/VM user LNX4.

   ```
   SET LAN HIPRSAMP GRANT LNX4
   ```

4. Query the LAN attributes. Note that LNX4 is added to the access list.

   ```
   QUERY LAN HIPRSAMP DETAILS
   ```

5. Revoke authorization for user LNX4.

   ```
   SET LAN HIPRSAMP REVOKE LNX4
   ```

6. LNX4 is no longer in the access list.

   ```
   QUERY LAN HIPRSAMP DETAILS
   ```

Automating Guest LAN Configuration on IPL

Because persistent guest LANs are not maintained across z/VM IPL, the following is a procedure to automate configuration of the LAN on IPL.

Define the Guest LAN in the SYSTEM CONFIG File

Add the DEFINE LAN statement to the SYSTEM CONFIG to ensure the LAN is defined on IPL. See Configuring Your System in *z/VM: CP Planning and Administration* for details on the SYSTEM CONFIG file.

Define and Couple Simulated NICs to the Guest LAN

Using the NICDEF statement, you can define and couple a simulated NIC to a defined guest LAN. See NICDEF Directory Statement in *z/VM: CP Planning and Administration* for additional information.

Remember to create a simulated NIC of the same type (HiperSocket or QDIO) as the guest LAN to which it is to be coupled.

The SPECIAL statement can also be used to define and couple a simulated NIC to a guest LAN. However, the NICDEF statement introduced in z/VM 4.4 is preferred because it has additional configuration options.

Chapter 5. Planning a z/VM Environment with OSA-Express and HiperSockets

This chapter provides guidance on planning and configuring a z/VM environment that uses OSA-Express and HiperSockets networking technologies. It explains the setup and configuration of OSA-Express adapters, including their types and modes (QDIO and non-QDIO). It explains port sharing and transport modes and how to define host programs and TCP/IP profiles for them.

The information also covers HiperSockets. HiperSockets memory-speed communication method, channel path identifier (CHPID) definitions, IOCP device attachment, and TCP/IP integration is discussed.

The information also outlines verification steps for connectivity at the channel path, device allocation, and TCP/IP levels. Sample configurations and tables for frame sizes and MTU values are included, along with references to related chapters and IBM documentation for further details.

Note: Although a Network Express (EQDIO) environment can run in conjunction with a Hipersockets (iQDIO) environment, this chapter does not provide examples of a z/VM environment that uses Network Express and HiperSockets networking technologies. For more information, see Construction of a Bridged HiperSockets LAN.

OSA-Express Network

The IBM OSA-Express is an integrated hardware feature that provides direct connection to clients on local area networks (LANs). The OSA-Express adapter plugs into an I/O slot just like a channel card. OSA-Express adapters are available for Gigabit Ethernet (GbE), 10 Gigabit Ethernet (10 GbE), 25 Gigabit Ethernet (25 GbE) and 1000Base-T Ethernet LAN connections. All adapters can use the IBM Queued Direct I/O (QDIO) architecture to eliminate the need for channel control words (CCWs) and interrupts, resulting in accelerated TCP/IP data packet transmission.

The OSA-Express adapter is identified in the hardware I/O configuration by its Channel Path Identifier (CHPID). The CHPID is assigned when the adapter is installed and is based on the number of features already installed in the server. For each OSA-Express CHPID installed, you must specify device numbers and unit addresses using Hardware Configuration Definition (HCD) or System Element (SE) panels. Gigabit Ethernet (GbE) connections can be defined as QDIO (OSD) only. 10 Gigabit Ethernet (10 GbE) and 25 Gigabit Ethernet (25 GbE) connections can be defined as QDIO (OSD) only. 1000Base-T connections can be defined as QDIO (OSD) or non-QDIO (OSE). For a non-QDIO OSA-Express channel, you cannot set up virtual networks like guest LAN.

Logical partitions (LPARs) can share the OSA-Express channel. In some OSA documentation this is referred to as port sharing. OSA-Express features defined as OSD channels do not require OSA/SF for port sharing.

You can define the OSA channel path to be shared among the LPARs to which it is defined in the system hardware I/O configuration (IOCDS).

z/VM is able to simulate OSA-Express devices that z/VM guest systems can use in a virtual environment. See Chapter 4, “Planning for Guest LANs and Virtual Switches,” on page 45 for more information.

Note: An OSA-Express adapter used for a Link Aggregation group in a VSWITCH requires exclusive use.

OSA-Express Multiple Ports Function

Some features of OSA-Express provide multiple ports per Network Interface Card (NIC), allowing more physical connections without requiring multiple channel path ids (CHPID) to be defined. For example, with an OSA-Express adapter having two channels, where each channel supports two ports, four physical ports are available for use. The z/VM TCP/IP stack can make use of the additional ports through specification of a port number in the DEVICE statement for QDIO devices. There are no changes required to the system

I/O definitions (eg. IOCDS, IOCP, or HCD) to utilize the additional ports. A device can only be associated with one port number at a time.

Interface Takeover for Local Area Networks

For information about the interface-takeover function, see Interface Takeover for Local Area Networks in *z/VM: TCP/IP Planning and Customization*.

Setting up an OSA-Express Network

To set up an OSA-Express network, perform the following steps:

1. Define host definitions.
2. Configuring OSA-Express modes.

Defining Host Definitions for Your OSA-Express Adapter

The host definitions required to set up your OSA-Express adapter vary depending on the following:

- OSA-Express adapter type
- OSA-Express mode

Table 3 on page 84 contains CHPID and TCP/IP profile definitions used when configuring the OSA-Express cards used in a z/VM virtual environment.

Table 3. Host Program Definition Summary for z/VM

OSA-Express adapter	Operation mode	CHPID type	TCP/IP device type	TCP/IP link type
GbE	QDIO TCP/IP	OSD	OSD	QDIOETHERNET
10 GbE	QDIO TCP/IP	OSD	OSD	QDIOETHERNET
25 GbE	QDIO TCP/IP	OSD	OSD	QDIOETHERNET
1000Base-T	QDIO TCP/IP	OSD	OSD	QDIOETHERNET
1000Base-T	Non-QDIO TCP/IP	OSE	LCS	ETHERNET, 802.3, or ETHEROR802.3

Configuring OSA-Express Modes

For OSA-Express adapters that run in non-QDIO mode (OSE channels), you must configure the physical port and create an OSA Address Table (OAT) to set up the data paths between the OSA and host programs. If any of these non-QDIO mode features use the default OAT without port sharing, OSA/SF is not required.

OSA-Express adapters that run in QDIO mode (OSD channels) require IOCDS definitions and host program setup. Once you have defined both the hardware and software definitions, the data path between any OSA-Direct Express adapter and the host programs are automatically set up in an OSA Address Table (OAT).

For additional information about OSA-Express and z/VM environments, see Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf). Also, see “Sample z/VM TCP/IP Profile” on page 85 for a sample TCP/IP profile containing statements related to OSA-Express and HiperSockets.

Configuring Transport Mode

A QDIO OSA-Express device can be configured for either the IP (Layer 3) transport type or the ETHERNET (Layer 2) transport type on the QDIOETHERNET LINK statement. IP is the default. When configured for

ETHERNET, the TCP/IP stack uses diagnose x'26C' to obtain a locally defined MAC address from CP. The MAC address will be generated in two parts. The first part is the MACPREFIX according to the VMLAN statement in the SYSTEM CONFIG file. The second part is automatically generated or can be set with the SET NIC command for a real network device. See "Media Access Control (MAC) address" on page 50 for more information about how MAC address are generated for z/VM.

Sample z/VM TCP/IP Profile

Figure 26 on page 85 shows only the statements related to OSA-Express and HiperSockets. You can use this sample z/VM TCP/IP profile to update your z/VM TCP/IP profile with the correct information.

```
DEVICE OSA2188   OSD 2188                           ; OSA Ethernet devices on CHPID 09
LINK   OSA2188L     QDIOETHERNET  OSA2188
;
DEVICE HIPERDEF HIPERS 7300                         ; HiperSockets CHPID EF
LINK   HIPLNKEF QDIOIP HIPERDEF
;
HOME

  192.168.1.62   OSA2188L
  10.10.1.62     HIPLNKEF
  ;
GATEWAY
; (IP) Network  Subnet       First      Link      Max. Packet
; Address       Mask         Hop        Name      Size (MTU)
; -----------   ------       --------   -------   -----------
;
; THESE ARE THE CURRENT GATEWAYS THAT ARE BEING USED TODAY IN PRODUCTION
192.168.1.0     255.255.255.0 =         OSA2188L      1500
10.0.0.0        255.255.255.0 =         HIPLNKEF      8192
;
START HIPERDEF
START OSA2188
```

Figure 26. Sample z/VM TCP/IP Profile

HiperSockets Network

HiperSockets microcode uses the OSA-Express queued direct input/output (QDIO) protocol (called the iQDIO), which establishes queues in the processor memory. The microcode emulates the link control layer of an OSA-Express QDIO interface. Communication occurs at memory speed and does not require the traditional I/O of external networks, which can result in overhead or delays.

HiperSockets does not need LAN frames, destination hosts, or routers. TCP/IP stacks are addressed by inbound data queue addresses. The microcode maintains a lookup table of IP addresses for each HiperSockets connection. The table represents a virtual LAN. When a TCP/IP stack starts a HiperSockets virtual device, the device is registered in the lookup table with its IP address and input/output data queues. When the TCP/IP stack terminates the link, the entry for the device is deleted from the table.

You configure HiperSockets TCP/IP devices in the same way you configure OSA-Express QDIO devices. You must define a channel path identifier (CHPID) for each HiperSockets connection. The CHPID type is IQD, and the number must be in the range from X'00 to X'FF'. AlthoughHiperSockets do not occupy any physical I/O connection positions, other I/O devices cannot use the CHPID defined for HiperSockets.

As with OSA-Express, z/VM is able to simulate HiperSockets devices that z/VM guest systems can use in a virtual environment. See Chapter 4, "Planning for Guest LANs and Virtual Switches," on page 45 for more information.

Note: HiperSockets operate only with the QDIO protocol. HiperSockets do not support the EQDIO protocol.

Setting up a HiperSockets Network

To set up a HiperSockets network, perform the following steps:

1. Define IOCP and attach the devices.
2. Define TCP/IP.

Defining IOCP for HiperSockets and Attaching the Devices

You need to define the IQD channel path to be shared and attach the devices.

Defining the channel path: Use the following IOCP definitions to define the channel path (CHPID) to use with a HiperSockets connection. You define the CHPID, the control unit, and the devices:

```
CHPID    PATH=(FE),PART=((LPAR1,LPAR2)),CHPARM=40,TYPE=IQD
CNTLUNIT CUNUMBR=FE00,PATH=(FE),UNIT=IQD
IODEVICE ADDRESS=(FE00,12),CUNUMBR=FE00,UNIT=IQD
```

FE

is the assigned channel path id. It is a common practice to assign HiperSockets CHPIDs starting at FF, in descending order.

LPAR1 and LPAR2

are the names of the partitions that have access to the HiperSocket at Power® on Reset (POR). Because no candidate partitions have been specified, any other partition can be added to the access list through dynamic I/O reconfiguration. The CHPID is defined implicitly as shared.

FE00

is an available device number. Its value was selected to clarify its relationship to the CHPID number.

12

is the number of HiperSocket devices available to each partition, providing each LPAR with four HiperSocket interfaces. If more devices are required, they can be added using dynamic I/O operations.

CHPARM=40

defines the HiperSocket frame size as 24KB, enabling use of a maximum transmission unit (MTU) value up to 16KB (16 384 bytes). An MTU value of 8992 enables TCP/IP to forward jumbo Ethernet frames across the HiperSocket interface without fragmentation. The CHPARM value may be selected depending on the largest MTU you require:

Table 4. Maximum frame size and TCP/IP maximum transmission unit (MTU). CHPARM defines the HiperSocket frame size. Maximum frame size and TCP/IP maximum transmission unit (MTU)

CHPARM value	Maximum frame size	TCP/IP maximum transmission unit (MTU) value
00 (default)	16 KB	8 KB
40	24 KB	16 KB
80	40 KB	32 KB
C0	64 KB	56 KB

You can use the HCD panels to define the CHPID, control unit, and devices.

Attaching the devices: To attach the HiperSockets I/O devices to a virtual machine when it is started, place DEDICATE or COMMAND ATTACH statements in the virtual machine's directory entry. Each HiperSockets interface requires three consecutive I/O devices on the same HiperSocket CHPID.

Using HiperSockets with TCP/IP

You need to modify the z/VM TCP/IP configuration to identify each HiperSockets interface.

As an alternative to using the DEDICATE or COMMAND ATTACH statements in the TCPIP directory entry, you may specify the `:attach.` tag in the SYSTEM DTCPARMS file. For example, :`attach.fe00-fe02` will define a single HiperSocket interface.

In the TCP/IP profile (PROFILE TCPIP), use the following statements:

- DEVICE with type HIPERS to identify the first address of the HiperSockets device triplet

- LINK with type QDIOIP to identify the specific network interface link name to the HiperSockets device whose name is specified in DEVICE
- HOME to identify a network address to the interface for each LINK statement
- GATEWAY to specify routing
- START to start the device

When the device starts, the device information associated with the CHPID is registered in the IP address lookup table.

With static routing, use a GATEWAY statement. Also, use a packet size that can accommodate the maximum frame size of the IQD CHPID. If you use dynamic routing, you do not need to code the GATEWAY statement.

Note: Within a Hipersockets CHPID, IP addresses of the Hipersocket interfaces can be in the same subnet or different subnets. Some applications can detect when hipersocket IP addresses are in different logical subnets and may issue warning or error messages.

Verifying z/VM TCP/IP Connectivity

You need to verify HiperSockets connectivity within your z/VM virtual network for the following functions:

- Verifying the channel path
- Verifying device allocation
- Verifying TCP/IP connectivity

Verifying the Channel Path

For an IOCP verification, use the following CP command to verify that the channel path to each device is online to the guest systems, where *xx* is the CHPID number:

```
q chpid xx
```

You receive PATH messages that indicate which devices are online.

Verifying Device Allocation

For information about how each device is allocated to TCP/IP and the guest systems, use the following CP command where *cu_num* is the device number address:

```
q cu_num
```

Note: An allocated device appears in the display message as an OSA-type device. The device driver is the same for QDIO and iQDIO in z/VM.

Verifying TCP/IP Connectivity

To verify TCP/IP connectivity, use the following command:

```
NETSTAT DEVL
```

The display message indicates that the proper I/O device and CHPID maximum frame size are defined to TCP/IP.

Use the following TCP/IP command to indicate that the correct routing table entry is added for the interface:

```
NETSTAT GATE
```

To verify IP connectivity, use the ping command for all TCP/IP stacks that participate in the network.

To verify the content of the OSA Address Table (OAT), use the NETSTAT OSAINFO command. For more information, see the NETSTAT Command in *z/VM: TCP/IP User's Guide*.

Chapter 6. Live Guest Relocation Networking Considerations

This chapter is a comprehensive technical guide focused on live guest relocation networking considerations within z/VM environments, particularly in single system image (SSI) clusters. It details the requirements and configurations necessary for relocating running virtual machines across systems without disrupting network connectivity. Key topics include handling real networking devices like QDIO and EQDIO with equivalency IDs (EQIDs), managing HiperSockets devices with unique MAC addresses, and ensuring virtual switch uplink ports have matching EQIDs.

The chapter also covers simulated network connections (NICs), virtual switch equivalency attributes required for relocation eligibility, and detailed checks that are performed to verify compatibility between origin and destination systems.

The VMRELOCATE command provides the ability for an eligible, running z/VM virtual machine to be moved transparently from one z/VM system to another within a z/VM single system image (SSI) cluster. More information about the VMRELOCATE command is available in *z/VM: CP Commands and Utilities Reference*. See VMRELOCATE.

There are several eligibility requirements related to networking devices that must be met when a guest is moved. A configuration mismatch between the origin and the destination of the live guest relocation might result in a CP flagged eligibility error, which will prevent the live guest relocation from commencing or yield connectivity errors when the guest is running on the destination system after relocation. More information about relocation requirements is available in *z/VM: CP Planning and Administration*. See Preparing for Live Guest Relocation in a z/VM SSI Cluster.

Real Networking Devices

This section describes live guest relocation requirements for real networking devices.

Dedicated Devices

In order to relocate a guest with associated real networking devices, each system within the SSI cluster must have connectivity to the same physical LAN segment. In z/VM, the system administrator must assign like equivalency IDs (EQIDs) to those network devices that are on the same LAN segment (broadcast domain). This manual setting indicates to z/VM that the customer ensures that the devices have equivalent networking connectivity and that a guest can be moved from one system to another using the same MAC address, IP addresses, or both addresses.

- Each guest that uses a dedicated QDIO device must be configured with three consecutive real and virtual networking devices, each configured with the same EQID.
- Each guest that uses a dedicated EQDIO device must be configured with one real and one virtual networking device, configured with a single EQID.
- An EQID is set with the SET RDEVICE command or the RDEVICE system configuration statement. See the RDEVICE Statement in *z/VM: CP Planning and Administration* for more information.

Live guest relocation of guests with dedicated networking devices requires that the guest OS issue a gratuitous ARP as part of its link initialization process on the destination.

A guest configured to a specific port of a multiport OSA feature on the origin member must also have equivalent connectivity to the LAN segment through a multiport card on the destination system. For example:

- If port 0 (default) is configured for the guest on the origin member, then guest will expect that port 0 on the destination is physically connected to the same LAN segment

- If port 1 is configured for the guest on the origin member, then the guest will expect that port 1 on the destination is physically connected to the same LAN segment
- If a guest connected to a single port card (port 0) on the origin member is to be relocated to a destination with a multiport card, then the guest will be using port 0 of the destination multiport card. Port 0 of the destination multiport card must have the same EQID as the origin single port card.
- If a guest is connected to port 0 on a multiport card on the origin member, this guest can be relocated to a single port card on the destination system. But, if the guest is connected to port 1 on a multiport card on the origin member, this guest cannot be relocated to a destination member with a single port card.

HiperSockets Devices

The HiperSockets (HS) firmware generates and manages their own locally administered MAC Addresses. Each real device number defined on a HS CHPID is assigned its own unique MAC Address, which includes both the CEC serial number and the physical device number. As a result, MAC Addresses assigned by HS firmware are unique and persistent for each device configured on a HS CHPID. MAC assignments for HS are totally outside the control of z/VM. When relocating a Linux guest, the MAC address currently in use will move with the guest to another system. Given a HS MAC Address is persistent per device, no other guest can use the relocated guest's device number while it's active in another SSI Member. Duplicate MACs across the SSI can occur if a guest is relocated to another system and then a different guest uses the same devices as the original guest.

There are two approaches to resolving this issue.

1. Customer Managed MAC Address Assignment Approach®

   Assign a unique MAC to each guest virtual device (for example, set LLADDR or MACADDR directives in configuration scripts) following these rules:

   - The X'02' bit has to be ON in byte 0 (for example, start with byte 0 values like 02, 12, etc to indicate managed-by-administrator).
   - The prefix (bytes 0-2) must NOT match any MACPREFIX or USERPREFIX in your Ethernet domain (for example, avoid MAC values that might be generated by CP).
   - Generate a unique MAC for each *user,vdev* combination (for example, avoid duplicate MAC addresses).

2. HiperSockets Managed MAC Address Assignment Approach

   In order for HiperSockets to automatically assign and manage a MAC Address, z/VM must be setup to insure a guest always use the same set of three devices on each system in the SSI whenever it's relocated. This means when z/VM relocates a guest back to the original system where the guest was first IPLed, it must select the same set of three HS devices. The following will configure z/VM to always use the same set of devices on each system across a relocation:

   - The three real HS devices assigned to a given guest must have the same EQID assigned within each SSI member. Assign a unique EQID for each guest in the z/VM's System Configuration File or dynamically by using a SET RDEV CP Command. This will force z/VM to always select the same set of HS devices on each system within the SSI when performing a guest relocation. Since no other guest will use this same set of real devices, HS firmware will always assign a unique MAC Address per guest.
   - The customer must insure only the guest assigned to the unique EQID ever uses the same set of three devices. Once the guest is relocated to another target system, his free HS devices on the source systems can't be attached to another guest.

Virtual Switch UPLINK Ports

As with dedicated devices, Virtual Switch UPLINK ports on the origin and destination must be assigned like equivalency IDs (EQIDs) when relocating guests with simulated network connections coupled to the virtual switch. The EQIDs indicate to z/VM that the virtual switches have connectivity to the same LAN segment and the guest can be moved. In the case of the UPLINK ports, the specific port used (on a single or multiport card) is not applicable.

Simulated Networking Connections (NICs)

Relocating guests with simulated network connections (NICs) has additional requirements/ considerations.

- A coupled NIC must be connected to a virtual switch with an active OSA UPLINK port.
- A NIC can be uncoupled.
- A NIC cannot be coupled to a guest LAN.
- Operands configured through the SET NIC command will be inherited on the destination system as part of the post live guest relocation process.
- The maximum number of devices supported for a single NIC is 8.

Virtual Switch Equivalency

The destination virtual switch of the live guest relocation must be equivalent operationally with the origin virtual switch for the following attributes:

- VSWITCH *name*
- TYPE QDIO. Virtual switches that are defined as TYPE IVL are not eligible for relocation. The live guest relocation feature supports TYPE QDIO virtual switches that operate in QDIO or EQDIO mode. QDIO mode at the origin can relocate to QDIO or EQDIO mode at the destination. Similarly, EQDIO mode at the origin can relocate to QDIO or EQDIO mode at the destination.
- ETHERNET/IP
- USERBASED/PORTBASED
- VLAN AWARE/VLAN UNAWARE
- NATIVE *vid*
- ISOLATION OFF/ON
- VEPA OFF/ON
- Operational UPLINK port and matching EQID
- RDEV *nnnn*.P*n*. For multiport OSA features, connectivity to the same LAN segment is required irrespective of the port numbers configured for the destination and origin virtual switch RDEV (UPLINK) ports.
- USERID and VLAN authorization (CP or ESM).

Relocation Eligibility Checks

Relocation eligibility checks ensure that the destination networking configuration matches, or is compatible with, the origin for simulated network devices.

When a guest has a simulated NIC (network interface) coupled to a VSWITCH, the source and destination network definitions are compared to determine if the connection disqualifies the guest from relocation. Table 5 on page 91 shows the relocation eligibility checks performed for the following VSWITCH attributes.

Table 5. VSWITCH attributes that are compared for relocation eligibility

Attribute	Requirement	First release supported	Notes
VSWITCH name	Name must be identical.	z/VM 6.2.0	

Table 5. VSWITCH attributes that are compared for relocation eligibility (continued)

Attribute	Requirement	First release supported	Notes
VSWITCH type	The origin and destination virtual switches must be defined as *TYPE* QDIO. QDIO *mode* at the origin can relocate to QDIO or EQDIO mode at the destination. Similarly, EQDIO mode at the origin can relocate to QDIO or EQDIO mode at the destination.	z/VM 6.2.0	Enforced in z/VM 6.3.0. Enforced in z/VM 7.3 with APAR VM66823 (for support of EQDIO mode).
UPLINK port	UPLINK port must be operational on both members and EQIDs must be identical.	z/VM 6.2.0	For multiport OSA features, connectivity to the same LAN segment is required irrespective of the port numbers configured for the destination and source virtual switch RDEV (UPLINK) ports.
USERBASED VSWITCH	Both must be USERBASED.	z/VM 6.2.0	Port numbers are not guaranteed to be preserved or checked across a relocation. This applies both to port numbers assigned by z/VM (those outside of the user-defined range) and user assigned port numbers (those within the user-defined range of 1 to 2048). If multiple ports are configured for the guest with different attributes, the ports can only be relocated to a destination system that supports Directory Network Authorization (DNA).
PORTBASED VSWITCH	Both must be PORTBASED.	z/VM 6.3.0	User-defined port numbers in the range of 1 to 2048 are preserved, checked and created as needed across a relocation. Port numbers assigned by z/VM (those outside of the user-defined range) are not preserved across a relocation. In addition, the z/VM assigned ports can only be relocated to a destination system which supports Directory Network Authorization (DNA).
Transport (IP / ETHERNET)	Transport type must be identical.	z/VM 6.2.0	Enforced in z/VM 6.3.0.
VLAN awareness	VLAN (AWARE/UNAWARE) must be identical.	z/VM 6.2.0	
NATIVE VLAN	Native VLAN id must be identical.	z/VM 6.2.0	Enforced in z/VM 6.3.0.
ISOLATION (OFF / ON)	Port isolation attribute must be identical.	z/VM 6.2.0	Enforced in z/VM 6.3.0.

Table 5. VSWITCH attributes that are compared for relocation eligibility (continued)

Attribute	Requirement	First release supported	Notes
VEPA (OFF / ON)	VEPA attribute must be identical.	z/VM 6.3.0	
USERBASED authorization	User access is inherited[1] from the source system (if not already configured on the destination system).	z/VM 6.2.0	Enhanced in z/VM 6.3.0 to update destination configuration.
PORTBASED authorization	User access by PORTNUM is inherited[1] from the source (unless another user has been granted the same PORTNUM).	z/VM 6.3.0	Enhanced in z/VM 6.3.0 to update destination configuration.
MACPROTECT (UNSPECIFIED / ON / OFF)	NIC setting is inherited[1] from the source configuration (but destination VSWITCH and SYSTEM settings might override NIC).	z/VM 6.2.0	
PORTTYPE (ACCESS / TRUNK)	Inherited[1] from the source configuration (if not already configured on the destination system).	z/VM 6.2.0	Enhanced in z/VM 6.3.0 to update destination configuration
VLAN vidset	Inherited[1] from the source configuration (if not already configured on the destination system).	z/VM 6.2.0	Enhanced in z/VM 6.3.0 to update destination configuration
PROMISCUOUS (OFF / ON)	PROMISCUOUS ON authority is inherited[1] from the source configuration (if not already configured on the destination system).	z/VM 6.3.0	
PRIQUEUEING ON	If the source virtual switch has PRIQUEUING ON, and the guest is exploiting priority queuing (PQUPLINKTX setting of HIGH or LOW), then the destination virtual switch must be able to exploit priority queuing in order to relocate the guest (PRIQUEUING ON). If the destination virtual switch has PRIQUEUING OFF, then the source guest PQUPLINKTX setting must be changed to NORMAL so that the guest is no longer exploiting priority queuing before relocation.	z/VM 7.1.0	See "Relocation Information for Priority Queuing" on page 56 for details.
PRIQUEUEING OFF	If the source virtual switch has PRIQUEUING OFF, then relocation will occur if the destination virtual switch has PRIQUEUING ON or OFF.	z/VM 7.1.0	See "Relocation Information for Priority Queuing" on page 56 for details.

[1] The relocation process automatically inherits certain attributes from the original configuration. When authorization is managed by CP commands, the configuration is updated to correct the difference between origin and destination networks. A warning message is issued to the VMRELOCATE command invoker to indicate that changes were made to the destination VSWITCH.

Chapter 7. Bridging a HiperSockets LAN with a z/VM Virtual Switch

This chapter details how to bridge a HiperSockets LAN by using a z/VM virtual switch. While HiperSockets channels provide intra-CPC communication, bridging via a virtual switch allows transparent Layer 2 connectivity between guests inside and outside the CPC without requiring extra routers. The virtual switch Bridge Port supports robust L2 bridging with features like single NIC configuration, live guest relocation, and cross-CEC broadcast domains for high availability. Only bridge-capable ports that support QEBSM can participate in bridging; non-bridge-capable guests (e.g., z/OS) require separate OSA adapters.

The chapter covers configuring primary and secondary Bridge Ports for failover, NIC distribution options for load balancing, and necessary hardware channel definitions (IQD CHPID with EXTERNAL_BRIDGED). It also emphasizes the importance of Path MTU Discovery (PMTUD) for optimal performance and describes constructing bridged HiperSockets LANs with examples of defining virtual switches, uplink ports, and bridge ports. Finally, it discusses cross-CPC bridged networks, which extend the LAN across multiple systems and ensure seamless, highly available Layer 2 connectivity among z/VM guests.

The virtual switch HiperSockets Bridge support expands the use cases and capabilities of the HiperSockets channel:

- Full function, industry standard robust L2 bridging technology.
- Single NIC configuration simplifies network connectivity and management
- No guest configuration changes required for exploitation (transparent to guest OS).
- Provides Live Guest Relocation (LGR) of guests with real HiperSockets Bridge Capable IQD connections within and between bridged CECs.
- There is no limit in the number of z/VM LPARs that can participate in a bridged HiperSockets LAN.
- Ability to create a single broadcast domain across multiple CECs (Cross CEC bridged HiperSockets channel network).
- By default the z/VM virtual switch provides a highly available network connection to the external network.

Only HiperSockets Bridge Capable Ports as defined in Chapter 2, “Introduction to Connectivity Terminology,” on page 15, are allowed to be bridged. Although non HiperSockets Bridge Capable Ports can also use the HiperSockets channel, no bridging will be performed for these ports.

See Figure 27 on page 96 for an example of a bridged HiperSockets configuration.

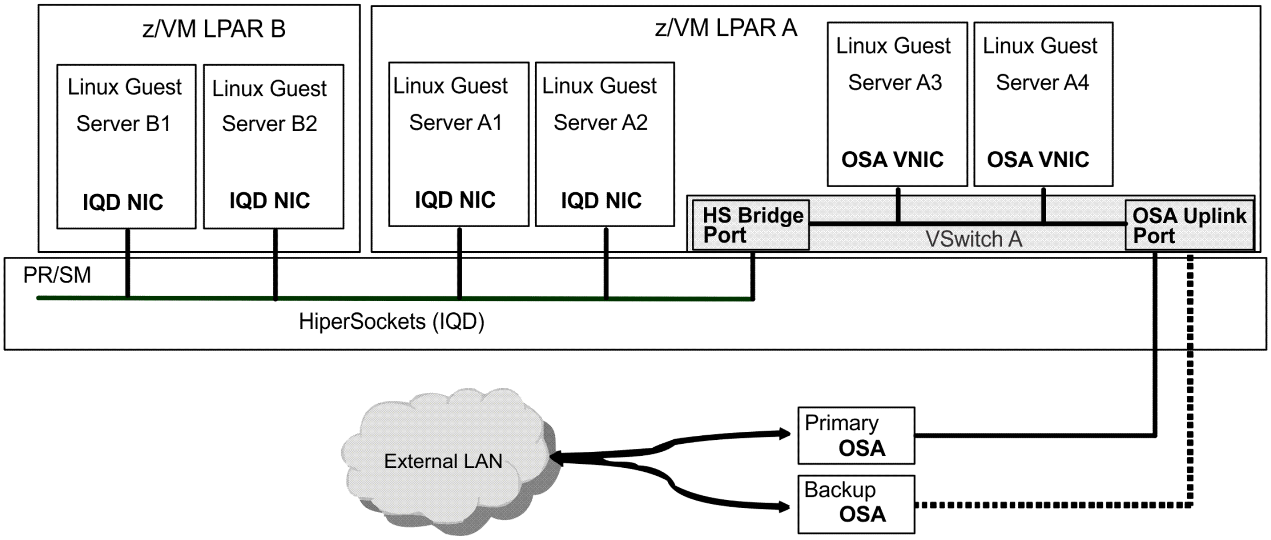


Figure 27. Bridged HiperSockets channel

z/VM Guest Networking Support Paradigm Shift with the Virtual Switch HiperSockets Bridge

When considering the deployment of virtual switch as a HiperSockets Bridge, a change in the way guests are provided access to the network should also be considered. Prior to the bridge support, guests may have simulated vNIC connections to a z/VM virtual switch for external connectivity. Now that the HiperSockets channel is bridged transparently to the external LAN segment, these guests may no longer need to be directly connected to the virtual switch for external connectivity. Instead, an option is that the bridge capable guests be moved from simulated vNIC connections (VSwitch) to HiperSockets dedicated connections on the bridged channel.

Deploying guests with dedicated HiperSockets connections reduces the amount of z/VM memory to support multiple interfaces (8M for input Queues). This configuration also exploits QEBSM, eliminating CP involvement with mainline network communication (reduced CPU utilization and decreased latency):

- Firmware performs all data moves between guests.
- CP only involved with guest to external network communications.
- Minimal z/VM overhead to provide bridge. Firmware will do most of the work via QEBSM.
- The QEBSM guests will operate similar to the native environment (bridging is transparent).

For IQD EXTERNAL_BRIDGED HiperSockets channels, there is no VLAN usage per guest authorization performed by firmware. If VLAN guest authorization is required, then the guest must remain connected to the z/VM virtual switch.

z/VM Virtual Switch Bridge Port (BRIDGEPORT)

The Bridge Port is an optional special purpose port on a z/VM virtual switch. This virtual switch port construct supports IQD HiperSockets channel attachment configured by IOCP or dynamic I/O to support a bridge connection. See "IQD CHPID Definition" on page 101 for more information. All traffic flowing on this port will be using a synchronous architecture, while the virtual switch LAN and externally directed traffic will continue to use an asynchronous architecture. All ingress traffic targeted for the HiperSockets channel from the virtual switch will be translated by CP from asynchronous to synchronous, insuring transparent bridging of these architectures.

This Bridge Port from a z/VM virtual switch perspective is a special type of guest port and not a virtual switch Uplink port. The OSA adapter will maintain the role as the virtual switch Uplink port. Although from a HiperSockets channel perspective, the new Bridge Port appears as a HiperSockets Uplink port. The HiperSockets Uplink port is created by the HiperSockets firmware as part of the creation of the virtual switch Bridge Port. All frames with unknown destinations will be sent by HiperSockets firmware to this Uplink port instead of being discarded. The virtual switch will route this frame based on its destination MAC address to one of its simulated NIC ports or the external LAN through its Uplink port. It is not required that a virtual switch that is configured with a HiperSockets Bridge Port must also have an OSA Uplink port configured.

The operational aspects of Bridge Ports are represented by the following:

Role
(PRIMARY/SECONDARY) Set by the z/VM system administrator when more than one virtual switch will be connected to a HiperSockets Channel for high availability. Only a single Bridge Port can be active on a HiperSockets Channel at any point in time. The role assigned to a Bridge Port determines the virtual switch that is preferred by the system administrator to provide the active bridge connection.

Status
(ACTIVE/STANDBY/INACTIVE) Set by HiperSockets firmware when a virtual switch is providing the *active* bridging for the HiperSockets channel or is in *standby* to provide the *active* bridging when necessary (for example, failover). A Bridge Port that is in *inactive* status is not ready or *disconnect* state.

State
(READY +others) Similar to the OSA Uplink port state, this reports the state of the controller BRIDGEPORT connection. The controller uses this connection to communicate with the HiperSockets firmware similar to its communications with the OSA adapter for the Uplink port. For more details on the actual state that can be reported, see the Usage Notes for the QUERY VSWITCH command in *z/VM: CP Commands and Utilities Reference*.

High Availability for Bridged HiperSockets Channel (LAN)

To insure continuous network availability for a bridged HiperSockets channel, it is recommended that every z/VM LPAR that has bridge capable guests connected to the HiperSockets channel, also have a virtual switch connected to the same layer 2 network with a secondary Bridge Port in standby status. There are two layers of failover operations that are in place for a bridged HiperSockets channel that insure sustained connectivity for the bridged channel:

HiperSockets Directed Failover Operation
When the HiperSockets firmware detects that the currently *active* Bridge Port is non-functional, it will immediately initiate an adapter initiated state change to place the subject Bridge Port into *inactive* status and select a Bridge Port currently in *standby* status and make it *active*. Once notified of its change to *active* status, the virtual switch Bridge Port assumes the bridging function for the HiperSockets channel. This unanticipated recovery action or system administrator initiated configuration change commences transparently to all HiperSockets bridge capable guest ports. Note if the failing Bridge Port is assigned the role of *primary*, then upon its subsequent recovery it will be returned from *standby* status to *active* status. If the failing Bridge Port was assigned a *secondary* role, then the HiperSockets firmware will maintain the current active configuration. If the secondary Bridge Port that went down recovers, it will be placed in *standby* status.

Virtual Switch Directed Failover Operation
The virtual switch has indigenous recovery operations for OSA adapter failures and controller failures. Uplink port (RDEV) redundancy is dependent on additional resources (RDEVs) being available (configured). An additional controller is already provided with z/VM to be deployed by the virtual switch as needed. Additional Controllers may be created by the z/VM System Administrator as needed. For more information on creating additional Controllers, see Usage Note 3 for the DEFINE VSWITCH statement in *z/VM: CP Commands and Utilities Reference*.

Primary/Secondary Roles for Bridge Ports

The virtual switch can be configured with the role of either PRIMARY or SECONDARY Bridge Port. The HiperSockets firmware supports the establishment of one primary and up to four SECONDARY Bridge Ports per HiperSockets channel. Although the capability exists for the specification of a PRIMARY Bridge Port, a PRIMARY Bridge Port is not required. The virtual switch defaults to SECONDARY Bridge Port when a BRIDGEPORT is defined. When a PRIMARY is not specified, then up to five SECONDARY Bridge Ports are supported per HiperSockets channel. A Bridge Port role, either PRIMARY or SECONDARY, is established for a virtual switch when a Bridge Port is created. This role can only be changed by the owning z/VM image and remains assigned as long as the Bridge Port is established. The role is relinquished when the Bridge Port is removed from the virtual switch by the SET VSWITCH BRIDGEPORT RDEV NONE command.

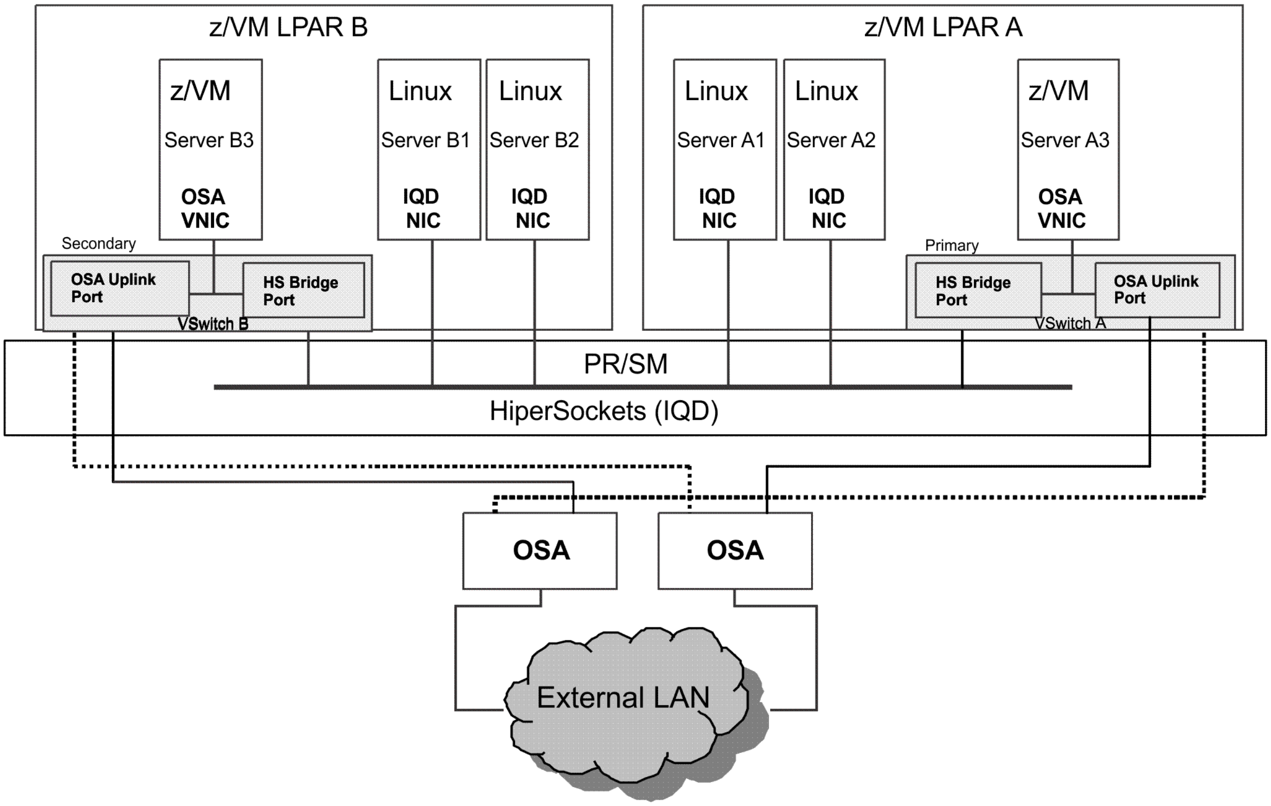


Figure 28. HiperSockets Channel with Primary and Secondary VSwitch Bridge Ports

Based on the configuration in Figure 28 on page 98, the QUERY VSWITCH A in Figure 29 on page 99 shows that its *bridge* port *Role* is defined as Primary and the HiperSockets channel *Status* for this Bridge Port is Active. As the active Bridge Port, the HiperSockets firmware will send all frames sent by Servers B1, B2, A1 and A2 with unknown destination MAC addresses to the active Bridge Port on VSwitch A. VSwitch A will either deliver the frame to A3 or send it to its Uplink port for delivery on the external network. Even though the Bridge Port for VSwitch B is in standby, VSwitch B still operates as traditional layer 2 bridge for any active simulated ports (B3). In this example VSwitch B will still send and receive frames for Server B3 over its Uplink port. When Server B3 needs to communicate with guests B1, B2, A1, A2 and A3 it will be over the external network.

The VSwitch Queries in Figure 29 on page 99 and Figure 30 on page 99 display the Bridge Port *Role, State* and *Status* for VSwitch A and VSwitch B from Figure 28 on page 98. This data also displays the LPAR that has the VSwitch with the active Bridge Port for this HiperSockets channel. The Bridge Port RDEV and VDEV devices are also displayed along with the controller that is providing the control link between the virtual switch and the HiperSockets firmware.

```
q vswitch A
VSWITCH SYSTEM A      Type: QDIO   Connected: 3  Maxconn: INFINITE
 PERSISTENT RESTRICTED  ETHERNET                  Accounting: OFF
  USERBASED
  VLAN Unaware
  MAC address: 02-00-00-00-00-A1 MAC Protection: Unspecified
  IPTimeout: 5    QueueStorage: 8
  Isolation Status: OFF
  Unicast IP address count: 0
 Uplink Port:
  State: Ready
  PMTUD setting: EXTERNAL PMTUD value: 8992
  RDEV: 4120.P00 VDEV: 4120 Controller: DTCVSW1 ACTIVE
 Bridge Port:
  Role: Primary   Status: Active   Active LPAR: A
  State: Ready
  RDEV: F200   VDEV: F200 Controller: DTCVSW2
 Ready;
```

Figure 29. Query of VSwitch A showing Bridge Port data

Notice in this Query VSwitch B Bridge Port display, LPAR A is where the primary Bridge Port resides for this HiperSockets LAN (see Figure 29 on page 99).

```
q vswitch B
VSWITCH SYSTEM B      Type: QDIO   Connected: 3  Maxconn: INFINITE
 PERSISTENT RESTRICTED  ETHERNET                  Accounting: OFF
  USERBASED
  VLAN Unaware
  MAC address: 02-00-00-00-00-01 MAC Protection: Unspecified
  IPTimeout: 5    QueueStorage: 8
  Isolation Status: OFF
  Unicast IP address count: 0
 Uplink Port:
  State: Ready
  PMTUD setting: EXTERNAL PMTUD value: 8992
  RDEV: 4110.P01 VDEV: 4110 Controller: DTCVSW2 ACTIVE
 Bridge Port:
  Role: Secondary  Status: Standby   Active LPAR: A
  State: Ready
  RDEV: F200   VDEV: F200 Controller: DTCVSW1
 Ready;
```

Figure 30. Query of VSwitch B showing Bridge Port data

Primary BRIDGEPORT Role

A functional primary Bridge Port is always selected by HiperSockets firmware for its active Bridge Port. HiperSockets firmware will select the primary Bridge Port as the *active* Bridge for a given HiperSockets channel anytime a primary Bridge Port becomes functional. For example, primary Bridge Port on VSwitch A in Figure 28 on page 98 will always be selected as the active Bridge connection for this HiperSockets channel even though a secondary Bridge Port on VSwitch B is currently active. In this example, the secondary Bridge Port on VSwitch B is transitioned by HiperSockets firmware from *active* to *standby* status whenever VSwitch A's Bridge Port is made functional (Ready State). The HiperSockets firmware will place VSwitch A Bridge Port into active status.

Configuring a specific virtual switch with a primary Bridge Port is only required from a predictability perspective when an optimum configuration exists to support the bridging of a loaded HiperSockets channel. As stated earlier, the primary designation insures HiperSockets firmware selection of the virtual switch as the active Bridge Port when it becomes functional.

Secondary BRIDGEPORT Role

A Bridge Port configured as secondary indicates to the HiperSockets firmware that the port can be assigned the active Bridge Port. Within a configuration where all the Bridge Ports for a bridged

HiperSockets channel are secondary, the HiperSockets firmware will choose which will become the *active* Bridge Port and the remaining secondary Bridge Ports will be in *standby* status. Typically the first functional secondary Bridge Port found by HiperSockets firmware will become the active Bridge port.

When a primary Bridge Port is functional and active, a secondary Bridge Port will be placed by HiperSockets firmware in *standby* status. When a primary Bridge Port goes into *standby* or *inactive* status, the HiperSockets firmware will select any one of the standby secondary Bridge Ports and place it in *active* status. Once a secondary is placed in *active* status by HiperSockets firmware, it will remain the active Bridge Port connection for the HiperSockets channel until the primary becomes functional again or itself incurs an error and loses connectivity.

Secondary Bridge Ports are required on each z/VM image in support of high availability where multiple z/VM images are configured with bridge capable guest ports connected to the same HiperSockets Channel. For example, a z/VM SSI Cluster where a guest can run in any of a number of different SSI members. In this case only a single SSI member can provide the bridge function for the same HiperSockets Channel within the cluster. By configuring the HiperSockets Channel as a secondary Bridge Port to a virtual switch in each SSI member, any z/VM system within the cluster can provide the bridge function for the HiperSockets Channel. This will allow any SSI member to be taken down even when there is a virtual switch within that image providing the active bridge connection. When an active bridge connection goes down, firmware will automatically select a virtual switch in another SSI member to resume the active bridge connection. With each virtual switch being configured equally within each SSI member, there is no compelling reason that one of these systems be configured as a primary Bridge Port. The recommended practice is to let all virtual switches default to secondary Bridge Ports and permit firmware to determine the active Bridge Port connection.

NIC Distribution for Bridge Ports

The virtual switch can be configured with NICDISTRIBUTION ON or OFF. The NIC distribution setting controls how the virtual switch distinguishes Ethernet frames that arrive from HiperSockets bridge capable ports to the bridge port.

When NICDISTRIBUTION is OFF (the default), the bridge port is considered a single source when transmitting packets from the HiperSockets guest ports to the virtual switch.

When NICDISTRIBUTION is set ON, the virtual switch examines each Ethernet frame to identify the HiperSockets logical guest port that sent the frame. The virtual switch treats each guest port on the HiperSockets CHPID as a distinct interface and uses these sources when distributing to the virtual switch uplinks or other simulated guests that are coupled to the virtual switch. This distinction can improve load balancing in a link aggregration configuration and when most guests are on the HiperSockets CHPID.

The NICDISTRIBUTION option is set independently on each bridge port in a Global VSwitch. It is recommended that the same setting be specified in each instance, especially when used with a Multi-VSwitch LAG configuration.

Construction of a Bridged HiperSockets LAN

The building of a bridged HiperSockets LAN requires configuration actions in both hardware IQD CHPID and z/VM VSwitch definitions.

Bridging of a HiperSockets channel is only supported in layer 2 (link layer) mode. That means from a configuration standpoint the virtual switch providing the bridging must be configured in Ethernet mode and the target HiperSockets CHPID (channel) will be operating in layer 2 transport mode when either of the following IQD CHPID channel parameters is configured. Operating Systems that do not support layer 2 mode will not be able to establish a network connection on a target HiperSockets channel. z/VM virtual switches that are configured to operate in Ethernet mode all require that the guest OS support layer 2 transport mode. For more information on z/VM virtual switch transport mode see “Considerations for Virtual Switch or Guest LAN Configuration” on page 49.

The following section gives an overview and examples of the steps required to build a bridged HiperSockets LAN.

Note: The example assumes a QDIO (OSD) type virtual switch, which is connected to an OSA-Express adapter. The construction of a bridged HiperSockets LAN with EQDIO (OSH) type virtual switch and Network Express adapter is similar and the same instructions can be used with the following caveats:

- In the example text and diagrams, substitute QDIO/OSD/OSA-Express terms with EQDIO/OSH/Network Express terms:

The example, which assumes QDIO/OSD/OSA-Express, uses this term:	For a bridged HiperSockets LAN with EQDIO/OSH/Network Express, substitute the following term:
QDIO	EQDIO
OSD	OSH
OSA-Express	Network Express

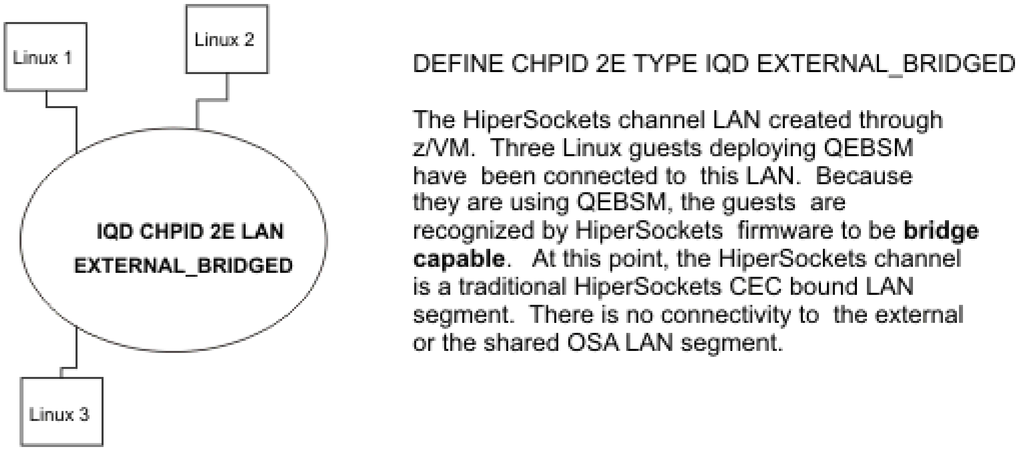
- A QDIO virtual switch requires virtual switch controllers. An EQDIO virtual switch has no requirement for virtual switch controllers.
- Responses from the CP QUERY virtual switch command contain some information that is specific to the VSwitch type (QDIO or EQDIO).

IQD CHPID Definition

The channel parameter for the IQD CHPID definition that enables HiperSockets bridging is EXTERNAL_BRIDGED for the traditional z/VM network and is specified in addition to setting the MFS size on the IQD CHPID definition. For more information on the DEFINE CHPID / PATH command, see *z/VM: CP Commands and Utilities Reference*.

EXTERNAL_BRIDGED

Specifically indicates the IQD HiperSockets channel is allowed to be bridged to a z/VM virtual switch irrespective of whether the CEC is or is not managed by zManager. There is no restriction on the number of IQD CHPIDs that can be defined with this channel parameter.



DEFINE CHPID 2E TYPE IQD EXTERNAL_BRIDGED

The HiperSockets channel LAN created through z/VM. Three Linux guests deploying QEBSM have been connected to this LAN. Because they are using QEBSM, the guests are recognized by HiperSockets firmware to be **bridge capable**. At this point, the HiperSockets channel is a traditional HiperSockets CEC bound LAN segment. There is no connectivity to the external or the shared OSA LAN segment.

Figure 31. IQD HiperSockets EXTERNAL_BRIDGED Channel with bridge capable ports

Only z/VM guests with dedicated HiperSockets connections that support QEBSM (QDIO Enhanced Buffer State Management Facility) are eligible to be bridged (Bridge Capable). The HiperSockets firmware will permit the guest to be bridged when that guest has deployed this facility. If disabled or if a given guest OS does not support this facility (for example, z/OS), the HiperSockets firmware will not consider the guest to be bridge capable. Linux by default enables this facility and will be bridge capable unless the facility is disabled. An OS that is running in an LPAR (not under z/VM) is also not bridge capable. Figure 31 on page 101 shows three Bridge Capable Ports (Linux) that have been attached to the HiperSockets channel.

z/OS guests are not bridge capable (do not support QEBSM). See “z/OS Guest Considerations in a Bridged HiperSockets Configuration” on page 110 for more information.

OSD and OSH CHPID Definitions

Based on the external network (LAN) you plan to connect the HiperSockets channel to, you will need to create the appropriate devices on an OSD or OSH CHPID. These devices will be configured to the virtual switch as its Uplink port and backup.

OSD CHPID Definition

Use the DEFINE CHPID command. For example: `define chpid 3E type osd`

Connectivity to the external LAN as well as destinations in other LPARs within the CPC is provided through an OSA-Express adapter. A device that is created through z/VM or IOCP is deployed to provide a connection to the OSA LAN segment. At this point, the OSA channel is shared by a z/OS LPAR and Linux LPAR.

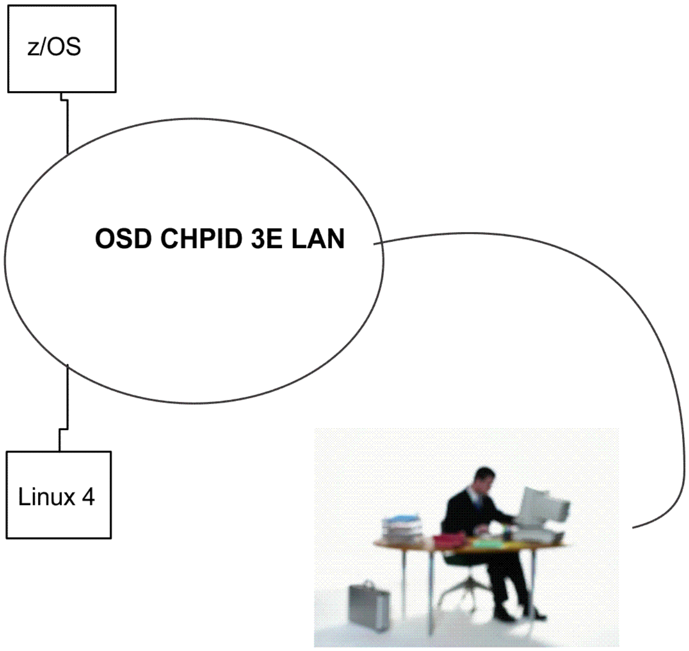


Figure 32. OSD CHPID to provide external network connectivity for virtual switch

OSH CHPID Definition

Use the DEFINE CHPID command. For example: `define chpid 4f type osh`

Connectivity to the external LAN as well as destinations in other LPARs within the CPC is provided through a Network Express adapter. A device that is created through z/VM or IOCP is deployed to provide a connection to the OSA LAN segment. At this point, the OSA channel is shared by two z/OS LPARs.

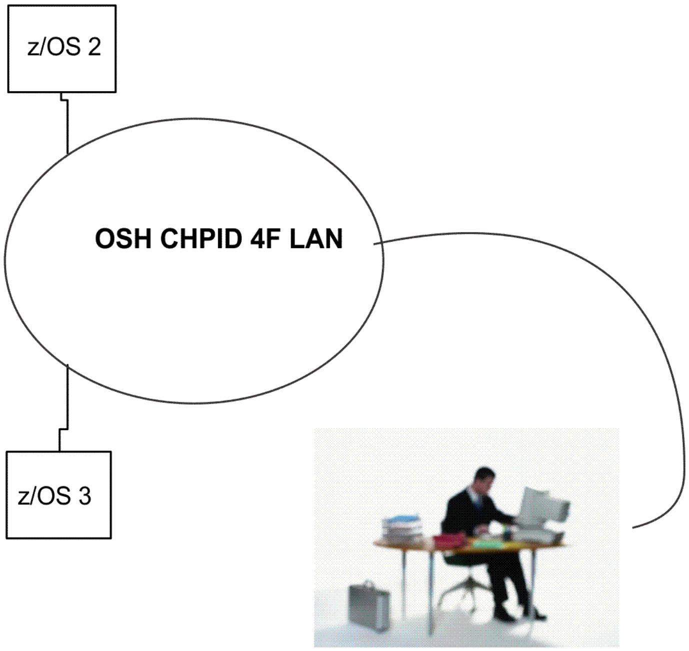


Figure 33. OSH CHPID to provide external network connectivity for virtual switch

z/VM Virtual Switch Bridge Port Definition

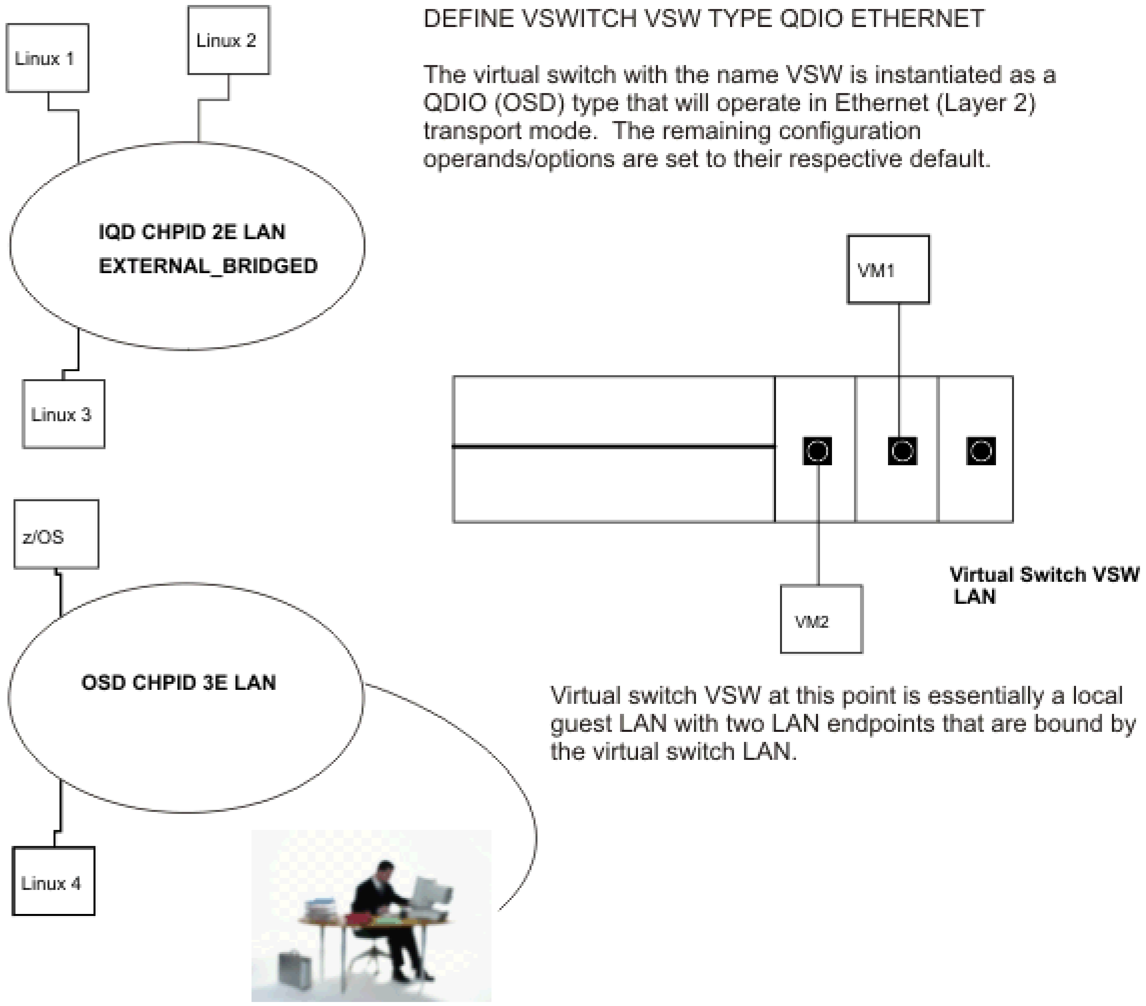
When defining a virtual switch to provide the bridging function for a target HiperSockets channel the following items need to be considered. This configuration procedure presents virtual switch functions that directly support or affect the instantiation of a Bridge Port. Those remaining virtual switch functions that are not presented here operate independently of the Bridge Port. It is recommended that all the DEFINE VSWITCH and SET VSWITCH operands be read in preparation to configuring the virtual switch. These can be found in *z/VM: CP Commands and Utilities Reference*.

A single z/VM DEFINE VSWITCH Command can create a virtual switch with both a Uplink and a Bridge Port. To simplify the explanation, we will perform the function here using multiple commands issued by privileged class "B" z/VM System Administrator user ID. The following four steps is an example of how to create and modify a virtual switch that will bridge a HiperSockets channel to a physical network connected by an OSA-Express adapter:

1. **DEFINE VSWITCH VSW TYPE QDIO ETHERNET**

   This command creates a layer 2 Ethernet virtual switch named VSW. At this point, Figure 34 on page 104 shows we have 3 independent layer 2 LAN segments. Network connectivity is only between guests connected to their same LAN segment. Linux 1, Linux 2, and Linux 3 can talk to each other, but they can't talk to Linux 4.

   - Define TYPE of Virtual Switch (QDIO). Note that the operational *mode* of the virtual switch will impose a corresponding type of RDEV device (QDIO mode requires OSD RDEV; EQDIO mode requires OSH RDEV).
   - If planning to use VLANs on the HiperSockets channel, then define the virtual switch as VLAN AWARE. Configure any additional VLAN options required by your installation.
   - Define virtual switch to be of Ethernet transport type.
   - Do not specify the NOUPLINK option because this will disable the ability to define an Uplink port.
   - Define any additional desired operands.



DEFINE VSWITCH VSW TYPE QDIO ETHERNET

The virtual switch with the name VSW is instantiated as a QDIO (OSD) type that will operate in Ethernet (Layer 2) transport mode. The remaining configuration operands/options are set to their respective default.

Virtual switch VSW at this point is essentially a local guest LAN with two LAN endpoints that are bound by the virtual switch LAN.

Figure 34. Instantiated virtual switch with no external connectivity

As shown in Figure 34 on page 104, guests VM1 and VM2 with OSD vNICs have also been COUPLED to virtual switch VSW through the following steps:

- Authorize guest to connect by SET VSWITCH VSW GRANT VM1
- Define a QDIO vNIC for guest by DEFINE NIC 710 QDIO
- Connect guest to VSW by COUPLE 710 SYSTEM VSW
- Repeat the same sequence of commands for guest VM2

2. **SET VSWITCH VSW UPLINK RDEV 0500**

   This command creates an Uplink Port on the virtual switch as shown by Figure 35 on page 105. If there are VSwitch controllers currently running on this image of z/VM, this same command will automatically establish a network connect between the virtual switch and the physical network connected to device 0500.

   - To continue to deploy a large HiperSockets MTU (MFS) between HiperSockets channel ports and also be able to deploy an acceptable MTU for external destinations PMTUD (PATH MTU Discovery) needs to be configured.

     a. Operating systems connected to Bridge Capable Ports on the HiperSockets channel, should have their TCP/IP stacks configured for PMTUD.

     b. The acceptable MTU for the external LAN provided by the OSA-Express Uplink port must be configured (if the default is not acceptable) on the virtual switch through the PATHMTUDISCOVERY operand. A virtual switch that is configured to provide the bridging function for a HiperSockets channel needs to have PTMUD operational. As described in “Path MTU

Discovery Protocol (PMTUD)" on page 53, this function provides the ability for a HiperSockets Bridge Capable port to deploy a large HiperSockets MTUs (MFS) for HiperSockets channel communications and deploy a lower MTU for external destinations over the OSA Uplink port. This is critical in maintaining the high bandwidth performance attribute of the HiperSockets channel while maximizing the allowable MTU of the external LAN. The virtual switch currently defaults to the maximum MTU supported by the OSA-Express adapter which is significantly larger than the traditional 1500 MTU. This will be fine or might be too large for your existing OSD network. The virtual switch MTU can be changed using the PATHMTUDISCOVERY VALUE operand of the SET VSWITCH CP command. The PTMUD default setting of EXTERNAL sets an MTU value of 8992. This is the MTU that will be used by the virtual switch to verify the payload targeted for the external network.

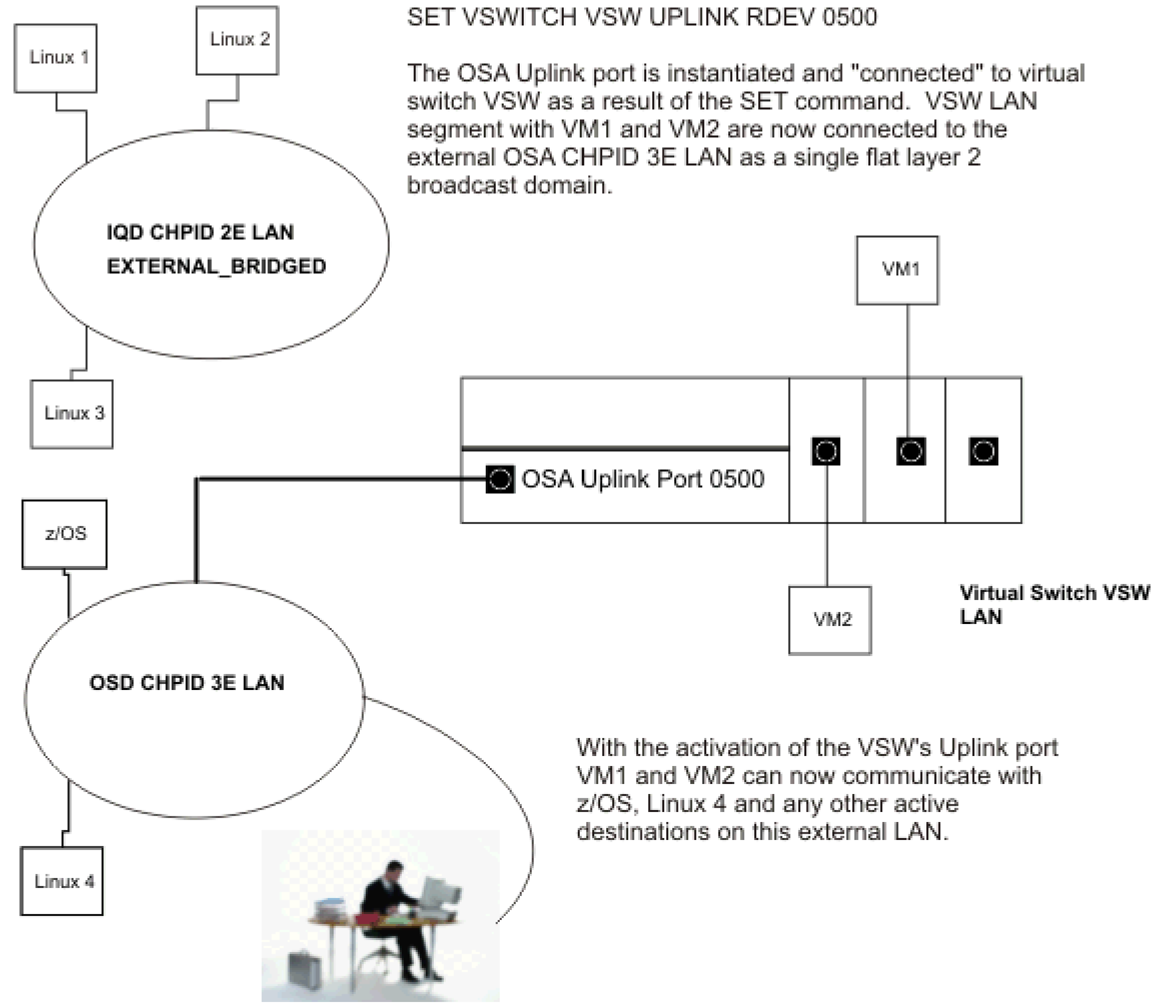
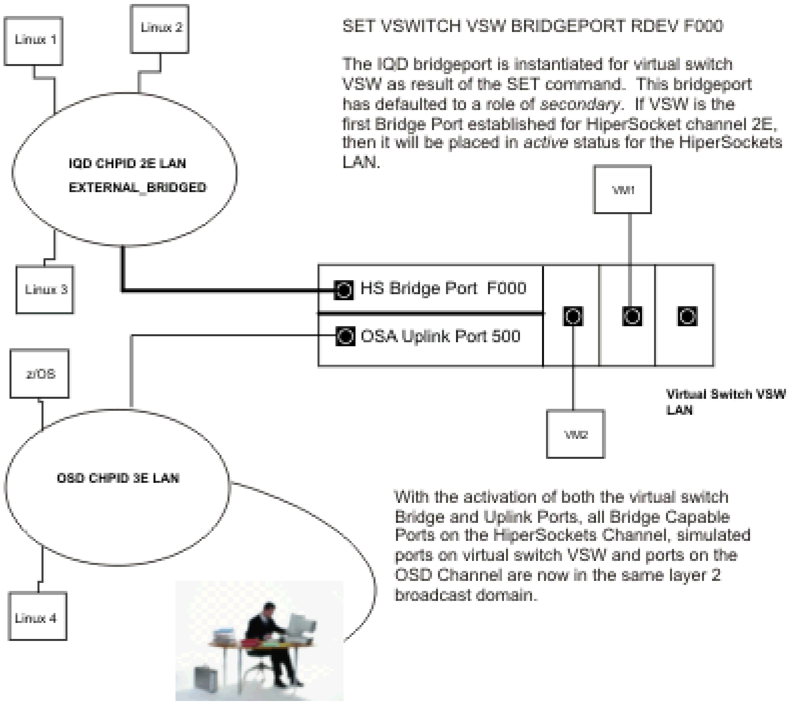

SET VSWITCH VSW UPLINK RDEV 0500
The OSA Uplink port is instantiated and "connected" to virtual switch VSW as a result of the SET command. VSW LAN segment with VM1 and VM2 are now connected to the external OSA CHPID 3E LAN as a single flat layer 2 broadcast domain.
Linux 1
Linux 2
IQD CHPID 2E LAN
EXTERNAL_BRIDGED
VM1
Linux 3
OSA Uplink Port 0500
z/OS
VM2
Virtual Switch VSW LAN
OSD CHPID 3E LAN
With the activation of the VSW's Uplink port VM1 and VM2 can now communicate with z/OS, Linux 4 and any other active destinations on this external LAN.
Linux 4

Figure 35. Definition of Activation of virtual switch Uplink port

3. **SET VSWITCH VSW BRIDGEPORT RDEV F000**

   This command creates a Bridge Port on the virtual switch as shown by Figure 36 on page 106. If there are VSwitch controllers currently running on this image of z/VM, this same command will automatically establish a network connect between the virtual switch and the HiperSockets Channel connected to device F000. As long as the Bridge and Uplink Ports are connected and in the READY state, Linux 1, Linux 2, Linux 3, VM1, z/OS, VM2 and Linux 4 can communicate with each other.

   - RDEV – Select a HiperSockets device from the configured devices (IOCDS) for the target IQD CHPID.
   - By default the Bridge Port may be placed in active or standby status. If this is not desirable then DISCONNECT must be specified on the SET VSWITCH command.



SET VSWITCH VSW BRIDGEPORT RDEV F000

The IQD bridgeport is instantiated for virtual switch VSW as result of the SET command. This bridgeport has defaulted to a role of *secondary*. If VSW is the first Bridge Port established for HiperSocket channel 2E, then it will be placed in *active* status for the HiperSockets LAN.

With the activation of both the virtual switch Bridge and Uplink Ports, all Bridge Capable Ports on the HiperSockets Channel, simulated ports on virtual switch VSW and ports on the OSD Channel are now in the same layer 2 broadcast domain.

Figure 36. Virtual Switch Bridged HiperSockets LAN

4. **Additional BRIDGEPORT operands**

SECONDARY / PRIMARY

Specifies the role this virtual switch Bridge Port plays when connecting multiple virtual switches to the same HiperSockets LAN segment. A PRIMARY Bridge Port will always provide the active bridge connection whenever it is active. If another virtual switch is currently providing the bridge connection to the HiperSockets LAN segment, and it's configured as a SECONDARY Bridge Port connection, then when the PRIMARY Bridge Port connection becomes active, the previous SECONDARY connection will cease to be used. The SECONDARY connection will go into standby status, ready to take over if the active Bridge Port connection goes down. Typically, a virtual switch Bridge Port should be configured as SECONDARY to allow any one of the virtual switches connected to the same HiperSockets CHPID to provide Bridge Port connectivity and backup. In this case the first Bridge Port made active on a HiperSockets CHPID will pick up the role of the active bridge connection. Only one active Bridge Port connection can be active at any point in time. All other SECONDARY connections will be in standby status, waiting to take over the bridge connection when the active Bridge Port goes down.

One reason for configuring a primary Bridge Port could be the necessity for predictability in which a specific virtual switch is desired to providing bridge connectivity when it is active. Or, it would be desirable for the virtual switch with the optimum bandwidth and throughput to be assigned the primary provider of LAN access to the HiperSockets channel. But usually for the majority of cases its recommended to allow the Bridge Port to default to secondary.

BUFFERS

Specifies the maximum number of buffers which may be allocated within z/VM memory for asynchronous Bridge Port data transfers. These buffers are used by the virtual switch to buffer unicast datagrams, which can't be immediately delivered to a HiperSockets Bridge Capable Guest. Specifically in the case where the HiperSockets Bridge Capable Guest can not supply empty buffers fast enough for arriving incoming data. Each buffer can hold up to 64K bytes of data, but the actual size is typically the MTU size configured for the network.

If a HiperSockets Bridge Capable guest cannot supply empty buffers fast enough for incoming data, the data must be resent by sender. To avoid this additional recovery overhead, the Bridge Port has the ability to buffer the data until the guest supplies an empty buffer. As long as the BUFFERS value is greater than zero and less than or equal to its set value, the pending data will be buffered in z/VM's memory, when it cannot be immediately delivered to a HiperSockets Bridge

Capable guest. When the guest supplies an empty buffer, the data is transferred at that time freeing the buffer for another data transfer.

The QUERY VSWITCH DETAILS CP command will display in the "Bridge Port" section the number of buffers configured in the "BUFFERS" field, the current number of buffers being used at the moment in the "In Use" field and the cumulative count of buffers used since the Bridge Port went active in the "Asynchronous Requests" field. If the number of "In Use" buffers equals the current set "BUFFERS" value, the Bridge Port will stop buffering any more data. This will result in additional recovery overhead to occur for data which cannot be immediately delivered. The cumulative count of the number of times this additional overhead occurs will be displayed in the "Unavailable Buffers" field of the query. For a properly tuned system, the "Unavailable Buffers" counter should be zero or as low as possible. If this counter is greater then zero, you may want to consider increasing the BUFFERS value to reduce the additional overhead.

NICDISTRIBUTION

Specifies the NIC distribution setting that the virtual switch will use for this bridge port. When set ON, the virtual switch identifies the HiperSockets bridge capable port from which each data transmission originated. The virtual switch uses the identification information to distribute the data transmissions among the virtual switch uplinks or other simulated guests that are coupled to the virtual switch. Such distribution achieves more balanced loads in a link aggregation configuration. Also, the virtual switch can observe and record IP addresses and maintain packet counters for each port.

5. **QUERY VSWITCH VSW DETAILS**

   This output displays the HiperSockets Bridge Port and HiperSockets Bridge Capable Ports:

```
q vswitch vsw det
VSWITCH SYSTEM VSW      Type: QDIO   Connected: 5  Maxconn: INFINITE
 PERSISTENT RESTRICTED  ETHERNET                  Accounting: OFF
 USERBASED
 VLAN Unaware
 MAC address: 02-00-00-00-0A MAC Protection: Unspecified
 IPTimeout: 5    QueueStorage: 8
 Isolation Status: OFF
  Unicast IP address count: 2
 Uplink Port:
  State: Ready
  PMTUD setting: EXTERNAL PMTUD value: 8992
  RDEV: 0500.P01 VDEV: 0500 Controller: DTCVSW2 ACTIVE
   Uplink Port Connection:
    RX Packets: 12        Discarded: 406  Errors: 0
    TX Packets: 23        Discarded: 0    Errors: 0
    RX Bytes: 888              TX Bytes: 1914
    Device: 0500  Unit: 000    Role: DATA    Port: 2049
  Bridge Port:
  Role: Secondary  Status: Active   Active LPAR: BCT2
  State: Ready
  RDEV: F000   VDEV: F000  Controller: DTCVSW1
   Bridge Port Connection:
    MFS: 16K    Buffer Limit: 2048 InUse: 0  Trace Pages: 8
    RX Packets: 8      Discarded: 0       Errors: 0
    TX Packets: 16809  Discarded: 0       Errors: 0
    RX Bytes: 792         TX Bytes: 949486
    Asynchronous Requests: 0     Unavailable Buffers: 0
    Device: F000  Unit: 000    Role: DATA    Port: 2057
   HiperSockets Bridge Capable Ports:         Connected: 3
    ID: LINUX1.F012
      Options: Ethernet Broadcast
       Unicast MAC Addresses:
        06-00-F0-03-00-12
       Multicast MAC Addresses:
        01-00-5E-00-00-01
        01-80-C2-00-00-21
        33-33-00-00-00-01
        33-33-FF-03-00-12
    ID: LINUX2.F015
      Options: Ethernet Broadcast
       Unicast MAC Addresses:
        06-00-F0-03-00-15
       Multicast MAC Addresses:
        01-00-5E-00-00-01
        01-80-C2-00-00-51
        33-33-00-00-00-01
        33-33-FF-03-00-15
```

Figure 37. Query VSwitch VSW output part 1

```
   ID: LINUX3.F018
     Options: Ethernet Broadcast
      Unicast MAC Addresses:
       06-00-F0-03-00-18
      Multicast MAC Addresses:
       01-00-5E-00-00-01
       01-80-C2-00-00-81
       33-33-00-00-00-01
       33-33-FF-03-00-18
Adapter Connections:            Connected: 2
Adapter Owner: VM1   NIC: 0710.P00 Name: UNASSIGNED Type: QDIO
 Porttype: Access
 RX Packets: 232   Discarded: 0   Errors: 0
 TX Packets: 40    Discarded: 0   Errors: 0
 RX Bytes: 11202        TX Bytes: 2448
 Device: 0712 Unit: 002  Role: DATA   Port: 0001
 Options: Ethernet Broadcast
  Unicast MAC Addresses:
   02-00-44-00-00-12 IP: 10.1.7.3
  Multicast MAC Addresses:
   01-00-5E-00-00-01
   33-33-00-00-00-01
   33-33-FF-00-00-12
Adapter Owner: VM2  NIC: 0710.P00 Name: UNASSIGNED Type: QDIO
 Porttype: Access
 RX Packets: 232   Discarded: 0     Errors: 0
 TX Packets: 40    Discarded: 0     Errors: 0
 RX Bytes: 11202         TX Bytes: 2448
 Device: 0712 Unit: 002  Role: DATA  Port: 0001
 Options: Ethernet Broadcast
  Unicast MAC Addresses:
   02-00-44-00-00-13 IP: 10.1.7.4
  Multicast MAC Addresses:
   01-00-5E-00-00-01
   33-33-00-00-00-01
   33-33-FF-00-00-13
Ready;  T=0.01/0.01 16:58:03
```

Figure 38. Query VSwitch VSW output part 2

Note: The same commands work for a bridged HiperSockets LAN with EQDIO (OSH) mode virtual switch and Network Express adapter. In an OSH/EQDIO LAN environment, no virtual switch controllers are required for the commands to establish network connections, and the command responses are slightly different.

Networking Connectivity in a Virtual Switch HiperSockets Bridge Configuration

As shown in Figure 39 on page 110 communications between LAN endpoints on HiperSockets channel 2E commence as they would without a virtual switch Bridge Port between HiperSockets channel LAN endpoints Linux 1, Linux 2, and Linux 3. The same is true for simulated virtual switch vNIC LAN endpoints like VM2 communicating on the virtual switch LAN and with external LAN endpoints like z/OS through the virtual switch Uplink OSA-Express port. Also shown are some examples of the connectivity that could potentially exist between the LAN endpoints through the virtual switch Bridge Port.

Linux 2 communications with VM1

Linux 2 is a bridge capable guest connected to the HiperSockets channel 2E. VM1 is a non-bridge capable guest connected to virtual switch SW1 through a simulated vNIC. All communications between these two LAN endpoints is across the virtual switch Bridge Port (BRIDGEPORT). The MTU for this communication path is either based on the configured MFS configured for HiperSockets Channel 2E or negotiated value between the two endpoints through PMTUD.

Linux 1 communications with Linux 4

Linux 1 is a bridge capable guest connected to the HiperSockets channel 2E. Linux 4 is an external LAN endpoint. All communications between these LAN endpoints is across the virtual switch's Bridge Port and Uplink port. The MTU for this communication is determined by the PMTUD that is configured for virtual switch VSW. This MTU will be presented to Linux 1 by virtual switch VSW through the dynamic path MTU discovery process.

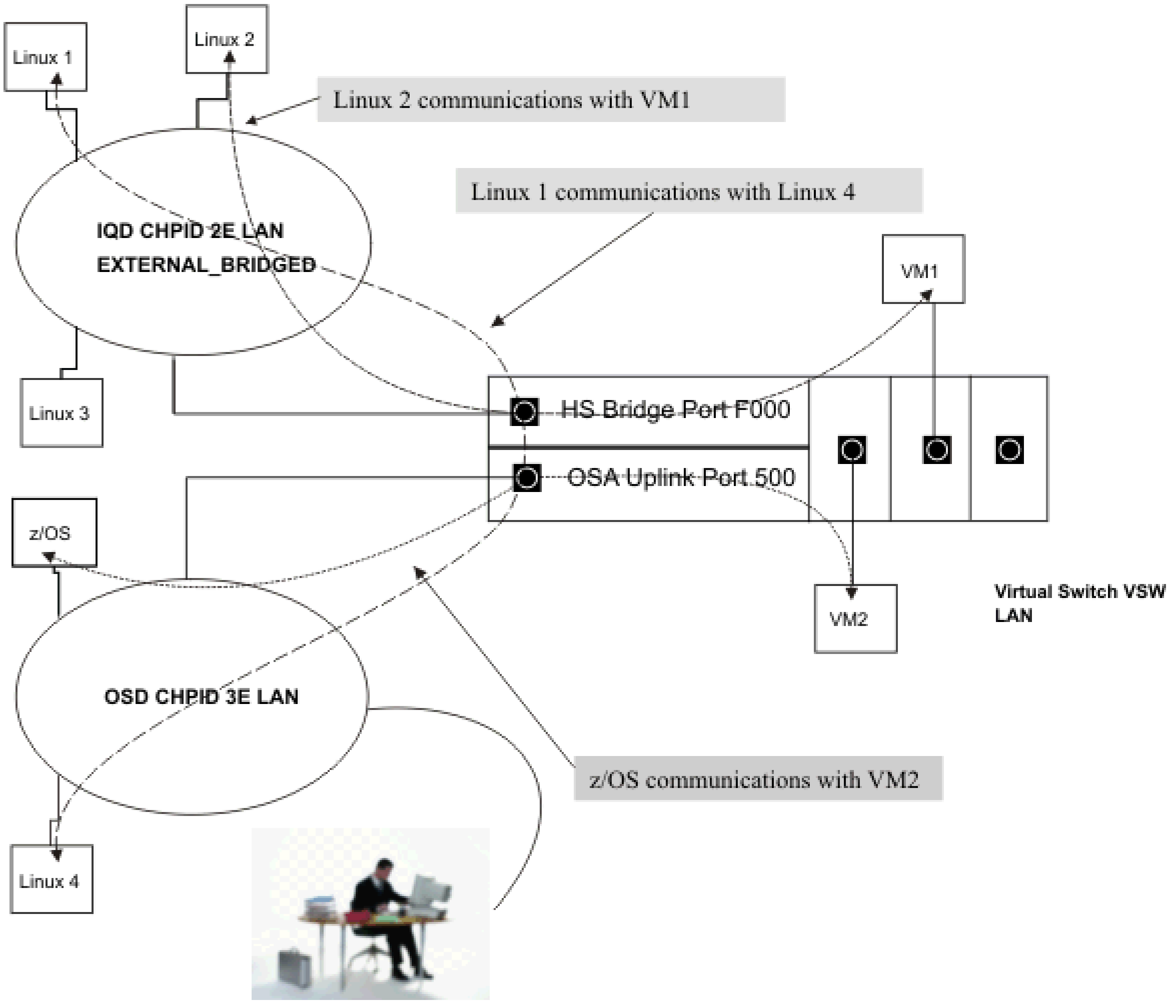


Figure 39. Virtual switch HiperSockets Bridge connectivity examples

z/OS Guest Considerations in a Bridged HiperSockets Configuration

As mentioned earlier, a HiperSockets channel connected z/OS guest is not bridge capable and therefore will not have connectivity to the external LAN through virtual switch HiperSockets Bridge Port. This means that z/OS requires its own dedicated OSA adapter (OSD or OSH CHPID) for connectivity to the external LAN.

- z/OS does not support the bridging of non managed (zManager) networks, that is IQD configured as EXTERNAL_BRIDGED. For a Bridge HiperSockets (IQD) channel, explicit OSD, OSH, and/or IQD interfaces must be configured.
- Virtual switches (QDIO and EQDIO) that are deployed as HiperSockets bridges are Layer 2 transports only. z/OS does not support Layer 2 mode for OSD or OSH vNICs. Connectivity through a virtual switch to the bridged HiperSockets LAN is not possible.

⚠ **Attention:** Do not configure z/OS or any other guest that has both HiperSockets and OSA interfaces to forward traffic from one link to the other, for this might cause duplicate packets to be generated and in some cases initiate a broadcast storm.

Path MTU Discovery (PMTUD) Requirement for Bridge Capable Guests

A virtual switch that is configured to provide the bridging function for a HiperSockets channel needs to have PMTUD operational. As described in “Path MTU Discovery Protocol (PMTUD)” on page 53, this function provides the ability for HiperSockets Bridge Capable Guest Ports to be able to deploy large HiperSockets MTUs (MFS) for HiperSockets channel communications and deploy a lower MTU for external destinations over the OSA Uplink port. This is critical in maintaining the high bandwidth performance attribute of the HiperSockets channel while maximizing the allowable MTU of the external LAN. The virtual switch currently defaults to the maximum hardware MTU supported by the OSA adapter, which is significantly larger than the traditional 1500 MTU. This will be fine or might be too large for your existing OSD or OSH network. The virtual switch MTU can be changed using the PATHMTUDISCOVERY VALUE operand of the SET VSWITCH CP command. By referencing Figure 40 on page 111, the QUERY VSWITCH command displays the current PMTUD default setting of EXTERNAL which indicates the maximum MTU value supported by the connected OSA Uplink port (8992). This is the MTU that will be used by the virtual switch to verify a payload targeted for transmission to the external network.

```
q vswitch A
VSWITCH SYSTEM A      Type: QDIO   Connected: 3  Maxconn: INFINITE
 PERSISTENT RESTRICTED  ETHERNET                  Accounting: OFF
  USERBASED
  VLAN Unaware
  MAC address: 02-00-00-00-00-A1 MAC Protection: Unspecified
  IPTimeout: 5    QueueStorage: 8
  Isolation Status: OFF
  Unicast IP address count: 0
 Uplink Port:
  State: Ready
  PMTUD setting: EXTERNAL PMTUD value: 8992
  RDEV: 4120.P00 VDEV: 4120 Controller: DTCVSW1 ACTIVE
 Bridge Port:
  Role: Primary   Status: Active   Active LPAR: A
  State: Ready
  RDEV: F200   VDEV: F200 Controller: DTCVSW2
 Ready;
```

Figure 40. Path MTU Virtual Switch setting and value

Cross CEC Bridged HiperSockets Channel Network

The virtual switch HiperSockets Bridge Port provides the ability to extend a HiperSockets LAN beyond the boundary of a single CEC. With the appropriate Layer 2 connectivity in place a virtual switch in each CEC can bridge separate HiperSockets channels into a single broadcast domain. This would provide the ability for bridge capable guests to communicate directly without the overhead and configuration complexity of next-hop routers. The cross CEC bridged HiperSockets channel network is a flat layer 2 broadcast domain.

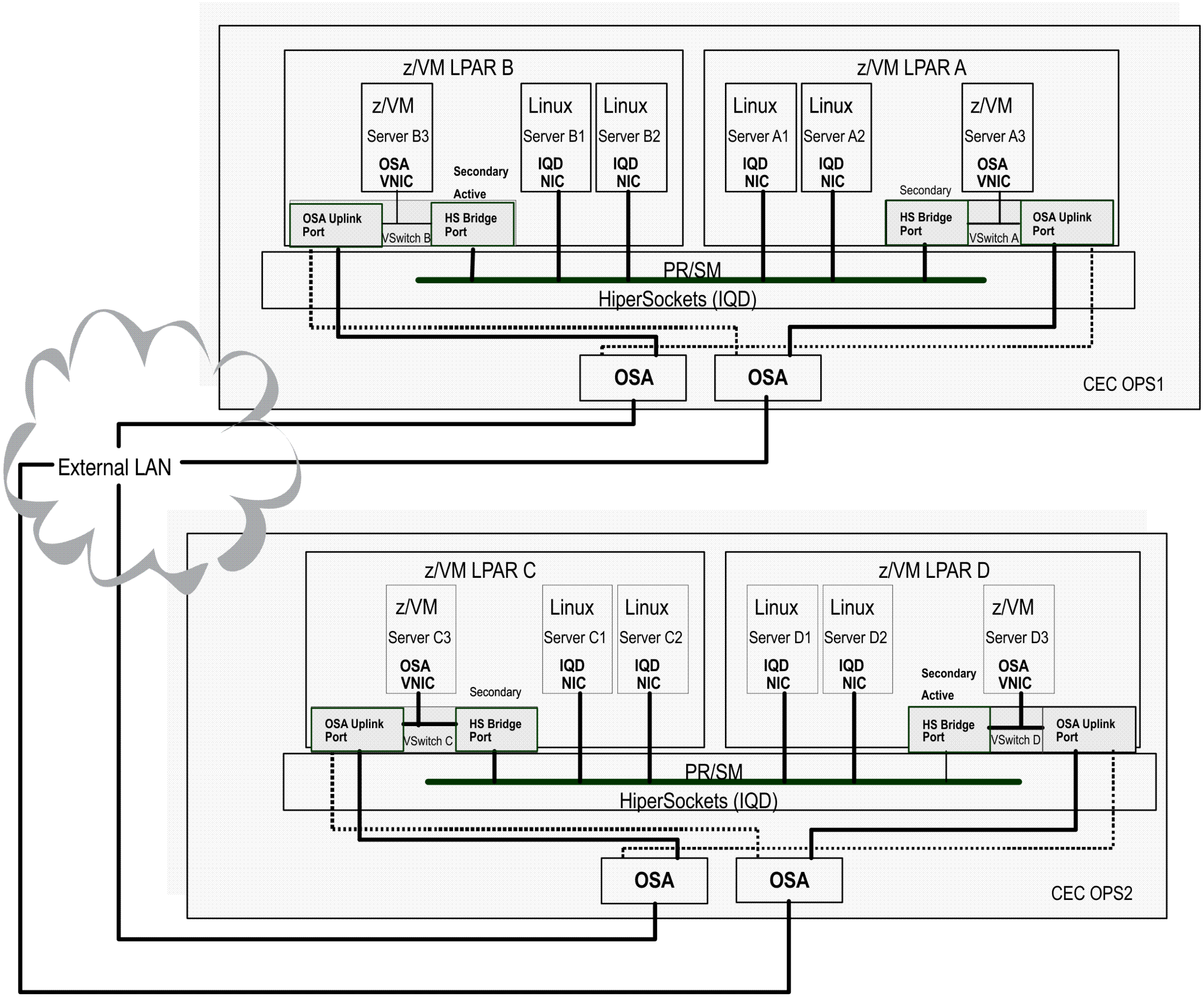


Figure 41. HiperSockets LAN spanning multiple CECs

Figure 41 on page 112 presents a cross CEC HiperSockets LAN configuration. In this configuration VSwitch B in LPAR B and VSwitch D in LPAR D are the active Bridge Ports providing external connectivity between the external bridged IQD channel in CEC OPS1 and the external bridged IQD channel in CEC OPS2. This flat layer 2 LAN essentially joins/extends the HiperSockets LAN between CECs across the external Ethernet network through VSwitch B Uplink port and VSwitch D Uplink port. Note that each CEC needs to have an *active* virtual switch Bridge Port with an Uplink port to provided connectivity to the external LAN. This configuration provides direct communications between all bridge capable guests and virtual switch guests within this bridged LAN. High availability is sustained transparently by virtual switch Uplink port failover operations and HiperSockets failover operations within each CEC.

Chapter 8. Virtual Switch Link Aggregation

This chapter details z/VM support for virtual switch Link Aggregation Groups (LAGs) and focuses on configuring and managing LAGs using IEEE 802.3ad standards with OSA adapters. It explains key concepts such as Link Aggregation Control Protocol (LACP), port groups (exclusive vs shared), and differences between exclusive LAGs and multi-VSwitch LAGs.

Exclusive LAGs are dedicated to a single virtual switch. Multi-VSwitch LAGs allow multiple virtual switches across one or more z/VM systems to share physical network resources for improved utilization and high availability. The chapter provides step-by-step examples for setting up exclusive and multi-VSwitch LAG configurations, including partner switch setup, port group creation, and verification commands.

The information also covers advanced features like LACP enhancements for multi-VSwitch environments, automatic device allocation, load balancing methods (independent vs collaborative), and recovery mechanisms for failover scenarios. Transparent operation to guest VMs and efficient bandwidth management within virtualized networks is emphasized.

Link Aggregation Overview

The Ethernet mode virtual switch supports the aggregating of up to eight OSA adapters with a switch that supports the IEEE 802.3ad Link Aggregation specification. For devices to be eligible for a particular group assignment they must have the following characteristics:

- The OSA adapters must support IEEE 802.3ad Link Aggregation.
- All OSA adapters have the same data rate specification (10, 100, 1000 mbps). The virtual switch will enforce and present a return code and message when a discrepancy is identified.
- Each OSA adapters must be connected to the same physical LAG. The physical LAG may consist of a single Ethernet switch or a set of Ethernet switches that are stacked or connected to each other to form a single logical switch form factor (conceptually a single switch). Some examples are Virtual Chassis, Virtual Switching System, vPC or etc.
- Target OSA adapters and the physical switch devices must be in full duplex mode. The virtual switch will enforce and present a return code and message when a discrepancy is identified.
- When the port group is configured as LACP Inactive, all ports in the group must support the network disablement function of the OSA adapters. OSA adapters that do not support the network disablement function may be used only in a port group configured with LACP Active. The network disablement function allows the virtual switch to inform the partner switch that a port is no longer functional and allows the partner switch to react and perform port recovery. Without this feature, failover time will be increased. If you are using these cards that do not support network disablement in a port group, you must configure LACP Active.
- When VLANs are deployed over the aggregated link, all member OSA ports within the group must be trunk ports to provide the virtual LAN connectivity in which to flow tagged traffic. The aggregated link should be viewed as one logical trunk port containing all the VLANs required by the LAN segment.

Link aggregation configurations provide increased bandwidth by utilizing multiple ports to transfer and receive data. The virtual switch provides this support through its virtualization structures transparently to connected guest VMs. VMs that are coupled to a virtual switch that is configured for link aggregation require no configuration or device driver changes.

Link Aggregation Control Protocol (LACP)

The 802.3ad IEEE standard presents the means for the forming of a single Ethernet link automatically from two or more Ethernet links using LACP. The LACP bonding between participating switches is a point to point connection using a predefined multicast MAC address for controlling LACP Protocol between each port. This protocol provides the means of assuring that both ends of the Ethernet link are functional and agree to be members of the aggregation group before the link is added to the group. These LACP operations are the responsibility of the active LAG Port Controller. LACP must be enabled at both ends of the link to be operational. LACP provides for the controlled addition and removal of physical links for the aggregation group such that no frames are lost or duplicated. The 802.3ad specification also provides for manual aggregation for the deployment of multiple links between switches without performing the LACP message exchanges. Manual aggregation is not as reliable or manageable as an LACP negotiated link.

The virtual switch supports the following LACP modes of operation:

- ACTIVE (default) – Full LACP negotiation with the partner switch, including the initiation of negotiations. (Active LAG Port Controller)
- INACTIVE – No LACP negotiations with the partner switch and any LACP packets received from the partner switch are ignored (PASSIVE mode is not provided). (There is no LAG Port Controller)

In order for LACP to determine whether a set of links can aggregate, several identifiers are required as outlined below. These identifiers allow exchanges to occur between systems so that on an ongoing basis, the maximum level of aggregation capability can be achieved. LACP communications occur on all OSA-Express ports that are part of the aggregated group.

1. System ID

   This identifier uniquely identifies the virtual switch and consists of a MAC address concatenated with a System priority. The system administrator can specify a MACID (to be concatenated with the MACPREFIX) on the virtual switch definition if a predictable MAC address is desired. Otherwise, the MAC address is assigned from within the system range.

2. Aggregation ID

   Each Aggregator (the function comprised of the Frame Collector and the Frame Distributor) requires a unique MAC address. This MAC address is the same address that is used to create the System ID.

3. Port ID and Capabilities

   This ID consists of a Port Priority concatenated with a Port Number. Port Numbers are unique within the VSWITCH and are assigned by CP. Each port also requires a capability identification, known as a Key that is used during negotiation to compare aggregate capabilities. There are two Keys - an operational Key and an administrative Key. These keys will be assigned from CP. These keys along with the other IDs are assigned by CP because the virtual switch supports only a single aggregation group per instance so there is no requirement for the System Programmer to differentiate between multiple groups.

4. Link Aggregation Group Id (LAG ID)

   The LAG ID is constructed of:

   a. The System ID

   b. The operational Key assigned to all ports in the LAG

   c. The Port ID

Link Aggregation Port Group

The grouping of links (OSA adapters) provides a single logical link from which the virtual switch will send and receive data from a partner switch device. The Port Group defines the set of OSA ports that will comprise the logical aggregated link. A LAG is the definition of an ETHERNET virtual switch with the GROUP attribute.

The scope of the port group is established by the EXCLUSIVE/SHARED operands on the SET PORT GROUP command. The EXCLUSIVE operand is the default operational mode. When specified, the OSA adapters that comprise the port group are configured to only support a single virtual switch connection (no sharing). Conversely when the SHARED operand is specified, the OSA adapters are configured to allow one or more Global virtual switches to share the same LAG.

The proper construction of a Port Group is a 4 step process:

1. Select the scope (EXCLUSIVE/SHARED).

2. Choose the OSA adapters and attributes.
3. Create the port group with appropriate scope, devices and attributes.
4. Configure the port group to a virtual switch (Uplink Port).

The SET PORT GROUP command can also be used to make changes to the list of devices associated with the virtual switch. Options are provided to add (JOIN) or remove (LEAVE) a device. The processing to remove a device involves suspending outbound traffic for the conversations associated with the link, sending a marker Protocol Data Unit (PDU), and reassigning all conversations when the marker PDU response is received or it times out.

When adding, deleting or using an OSA port in an aggregated group, the port will be in one of the following states:

Active
VSwitch Uplink port is fully operational (active).

Attached
OSA port is currently available and in use for data communication within the aggregated group.

Waiting
OSA port is available but is not currently being used for data communication within the aggregated group.

Suspended
All new data transmissions are suspended until pending requests complete or they can be moved to another OSA port.

Error
OSA port is unavailable due to a problem with the OSA adapters connection. The VSwitch controller is having a problem or the OSA port is not compatible with the aggregated group.

The current state of an OSA port is reported by the QUERY VSWITCH and QUERY PORT CP commands. In addition to the state, the reason the port is in the specific state is also returned in the response.

Exclusive Port Group

An exclusive port group is a LAG that is dedicated to a single virtual switch within a single z/VM system. The OSA adapter enforces this exclusive use, by not accepting any additional connections from the local z/VM system or any other z/VM systems within the CPC.

The SET and QUERY PORT GROUP CP commands are used to manage a LAG Uplink Port configuration. When the port group is defined with the EXCLUSIVE operand, the command's scope is local with only a single point of control from the local z/VM system. Any configuration changes made to the port group will remain local to the image of z/VM it was issued.

Shared Port Group

A shared port group is a LAG that can be shared by global virtual switches with the same z/VM system or other z/VM systems within the same CEC. Cross CEC sharing is not supported.

The SET and QUERY PORT GROUP CP commands are used to manage a LAG Uplink Port configuration. For a port group to be shared, it must be defined with SHARED operand so that its scope is global. This also provides the ability for these commands to be issued from any active z/VM system in the same IVL domain. Any configuration changes made to the port group are propagated to all systems. The operands that are supported as global scope are: JOIN, LEAVE, DELETE, LACP ACTIVE SHARED and LOADBALANCING. Additionally, the INTERVAL operand becomes global in scope when the LOADBALANCING operand is set to COLLABORATIVE.

Note: An OSA-Express adapter that is a multiport card may have one of its ports configured in a shared port group. The remaining port(s) may be configured to another shared port group, an exclusive group or directly to a VSwitch. But, sharing of the remaining ports(s) of the OSA feature with other guest virtual machines or operating systems is NOT supported.

The sharing capability of a shared port group:

- A port group (LAG) can be connected to up to 16 LPARS (single CEC). A port group cannot span multiple CECs.
- A total of 64 VSwitches can share this Port Group (LAG) across these 16 LPARs.
- Up to 4 VSwitches (instances) per z/VM system can share this Port Group (LAG).

So 4 VSwitches per LPAR * 16 LPARS = 64 Total virtual switches sharing the same Port Group (LAG).

Exclusive Link Aggregation vs Multi-VSwitch Link Aggregation

Both Exclusive and Multi-VSwitch LAG configurations provide the same industry standard IEEE 802.3ad Link Aggregation protocol support with an external partner switch. Both of these LAG configurations are completely transparent to z/VM guest hosted by either a simulated NIC connected directly to the VSwitch or a HiperSockets Bridge NIC connection.

The primary difference between these two configurations is the LAG sharing characteristics. A LAG (Link Aggregation Group) is **exclusively** used by a single virtual switch or **shared** by multiple virtual switches. Some comparisons between the two LAG deployments:

Comparison Items	Exclusive LAG	Multi-VSwitch LAG
Application	Simpler for a single system LAG.	More and geared for multiple VSwitch access to the same network. Typical but not limited to multiple z/VM systems.
OSA Resources	Dedicated networking ports service one virtual switch. Volume of network traffic requires high bandwidth and throughput.	Shared resource result in a reduction of physical ports that are more fully utilized.
Network Access	Network access is limited to a single z/VM system and VSwitch.	Access to the network can span multiple VSwitches within a z/VM system and across many z/VM systems within the CEC.
Port Group Management	Must be managed on the z/VM system that the port group was defined.	Can be managed on any z/VM system within the same IVL domain.
Physical Switch Partner	Each Exclusive LAG requires a separate and distinct physical LAG to be configured and managed on the partner switch.	Only a single physical LAG configuration required to support a Multi-VSwitch LAG configuration. The single physical LAG provides network connectivity and high availability for all z/VM systems within the same CEC.

Exclusive Link Aggregation (default)

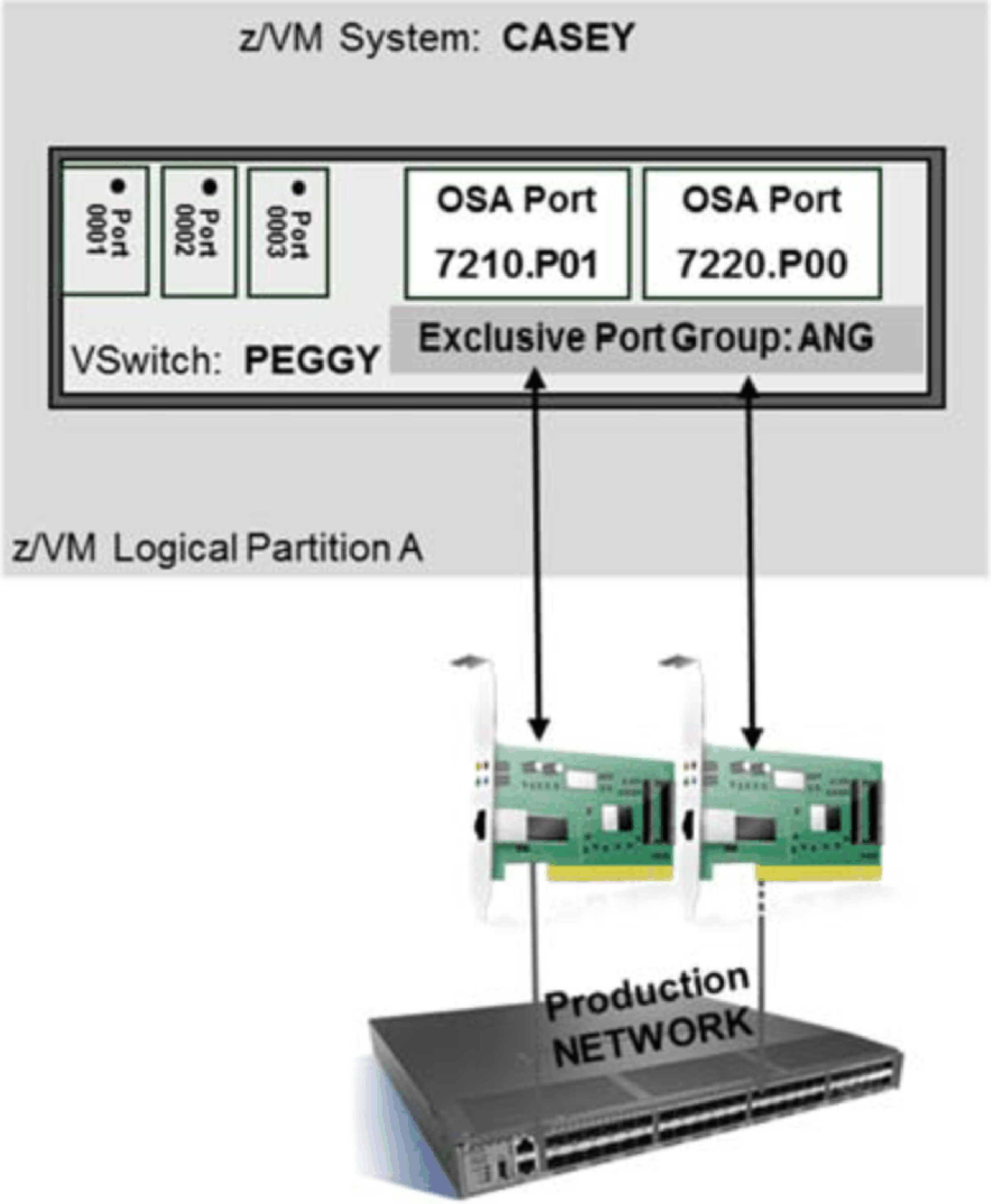


Figure 42. Exclusive Link Aggregation Configuration

As shown in the Figure 42 on page 117, VSwitch PEGGY has its own autonomous physical LAG with a discrete set of OSA-Express adapters to service its specific LAG. Each OSA-Express adapter is in exclusive single use mode (dedicated) to a single VSwitch. No sharing of the OSA-Express adapter is permitted with other VSwitches, guest virtual machines or operating systems. The operational characteristics of an Exclusive LAG are:

- VSwitch PEGGY establishes and maintains active LACP communications with its perspective partner switch through both devices in exclusive port group ANG. These communications include the LACP Marker Protocol to move traffic from one OSA-Express adapter to the other during load balancing operations. This insures total isolation of data and control (LACP) traffic between virtual switch LAG configurations. Absolutely no sharing of this port group with another virtual switch.
- Load balancing of outbound traffic over the devices in port group ANG is performed by VSwitch PEGGY. The VSwitch's load balancing scope is its local guest ports. Load balancing of external traffic (inbound) is provided by the external partner switch.
- When multiple z/VM systems are deployed the system administrator is required to configure and maintain distinct LAGs on both physical and virtual switches on each system. With two physical LAGs, high availability is achieved by building Uplink Port redundancy within each VSwitch. As a result a minimum of four OSA-Express adapters must be configured for exclusive VSwitch use. This is only efficient if each virtual switch can drive all of its OSA-Express adapters consistently close to one hundred percent bandwidth. Typically the network is not set up to consistently run at a hundred percent capacity. Therefore it is more practical to add or remove network capacity on demand, while providing high availability at all times. It may be possible to reduce both the management and financial burden of the default Virtual Switch Link Aggregation configuration by sharing the OSA-Express adapters across multiple Virtual Switches.

Multi-VSwitch LAG Configuration

A Multi-VSwitch LAG requires additional communication resources to provide the environment.

- A fully operational IVL (Inter-VSwitch Link) must exist to define a global VSwitch and a shared port group. A fully operational IVL consists of a successfully defined IVL VSwitch with an uplink port that is connected and actively communicating on the IVL domain across multiple systems.
- One or more global VSwitches within a single z/VM system do not need an external IVL network to communicate with each other.

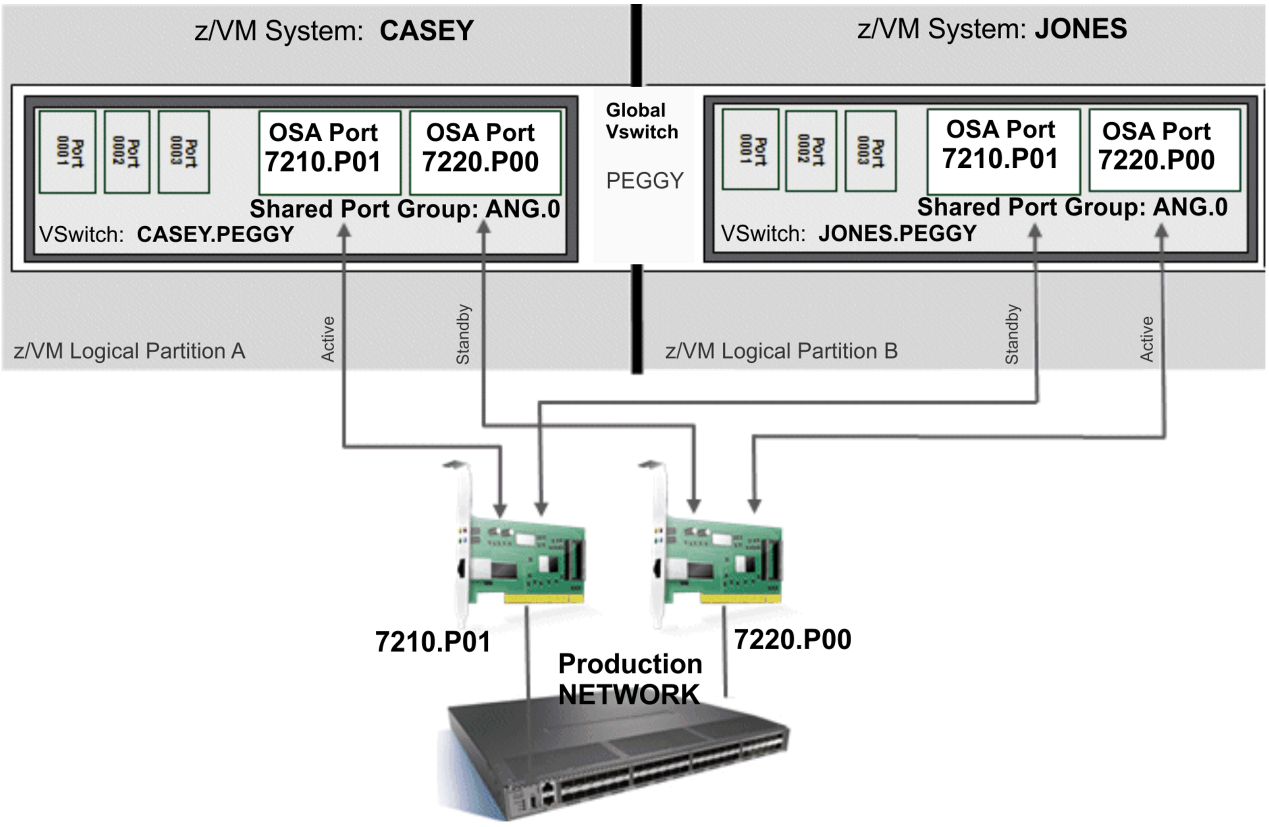


Figure 43. Multi-VSwitches LAG Configuration

As shown in Figure 43 on page 118, virtual switch PEGGY in each z/VM System is connected to the same physical LAG (ANG) through shared OSA-Express adapters. Sharing a LAG with multiple virtual switches increases both optimization and utilization of the OSA-Express adapter by allowing it to handle larger traffic loads.

The multiple z/VM images sharing the same OSA-Express adapter within the LAG believe the port is dedicated for their exclusive VSwitch use when in reality, the virtualization layer is allowing the port to be shared without diminishing the HA (High Availability) or bandwidth characteristics provided by the IEEE 802.3ad industry standard.

Exclusive LAG Configuration

An exclusive LAG configuration is the default z/VM configuration and includes the following elements:

- Configuration of a partner switch
- Configuration of an exclusive port group
- Configuration of the port group as an uplink for the virtual switch

The following topics provide an example of configuring an exclusive LAG. In this example, the adapter is an OSA-Express 1G two-port network interface card.

The instructions in the following topics yield the configuration that is pictured in Figure 44 on page 120. The exclusive port group **ANG** provides the external LAG connectivity for the virtual switch **PEGGY**.

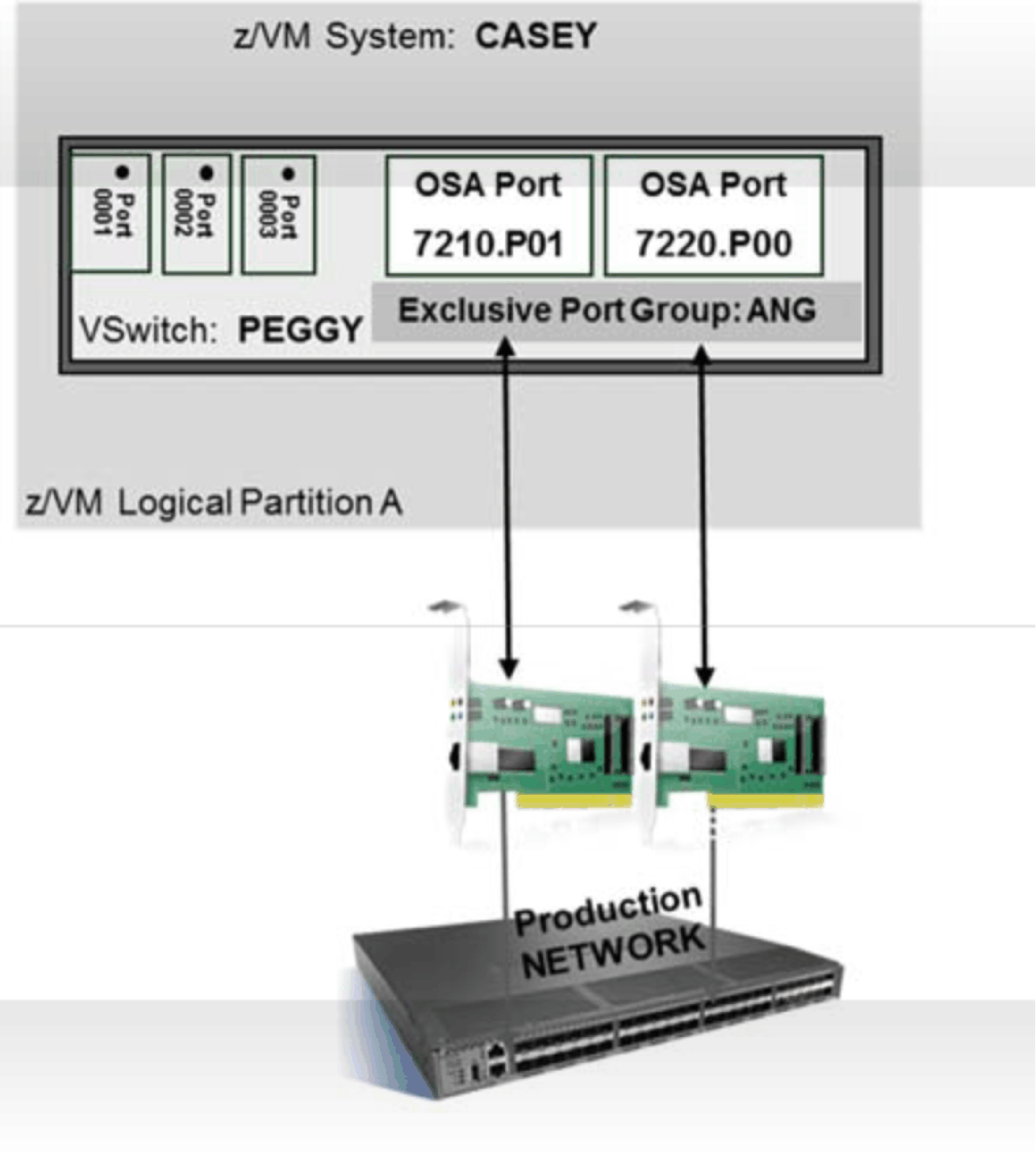


Figure 44. Exclusive LAG Configuration

Configure the Partner Switch

Select OSA adapters for your production LAG environment. Configure a LAG on the target physical partner switch. Each switch manufacturer has its own unique method of configuring a LACP LAG configuration.

Create an Exclusive Port Group

The target virtual switch PEGGY will be configured with a port group with exclusive scope. The OSA adapter that is chosen for this port group (ANG) will provide network connectivity exclusively for virtual switch PEGGY. In this example, the adapter is an OSA-Express 1G two-port network interface card. Active LACP communications with the partner switch is configured. LACP ACTIVE is recommended for load balancing and to insure timely failover in the event of a port failure. The port group is created by using the following commands:

- **SET PORT GROUP ANG LACP ACTIVE**
  - Port Group name is up to an 8 Character Name specified on the command
  - Port Group managed by the LACP protocol
  - Exclusive LAG is the default mode of operation.
- **SET PORT GROUP ANG JOIN *7210.P01 7220.P00***
  - Configure the OSA adapters that match the LAG configuration of the partner switch.

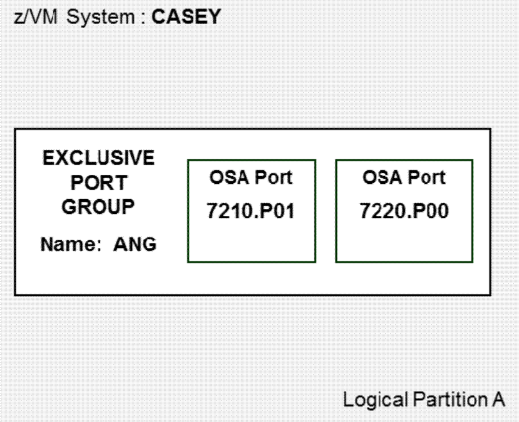


Figure 45. Exclusive Port Group

- **Verify port group ANG by issuing a QUERY PORT GROUP ANG DETAILS command**
  - Figure 46 on page 121 shows the query response, which indicates that on z/VM system CASEY, port group ANG is of scope EXCLUSIVE. The target devices are configured and when activated the port group will commence in active LACP communications.
  - At this point port group ANG is not configured as an uplink port of any VSwitch.

```
Q PORT GROUP ANG DETAILS

 Group: ANG         Inactive LACP Mode: Active   Exclusive
  VSWITCH <none>                                 ifIndex:  2112
  Load Balancing: Independent     Interval: 300
  RDEV: 7210.P01   Adapter ID: 296400081657.01C8
  RDEV: 7220.P00   Adapter ID: 296400081657.01F8
Ready; T=0.01/0.01 15:18:44
```

Figure 46. Query of Exclusive Port Group ANG

Configure the Exclusive Port Group to a VSwitch

For our configuration the following command is issued on z/VM system CASEY to configure the Uplink option on VSwitch PEGGY:

- If the VSwitch has not been previously defined, then issue a **DEFINE VSWITCH PEGGY ETHERNET**

  Note: The **ETHERNET** operand is required for VSwitches configured to a LAG configuration.
- **SET VSWITCH PEGGY UPLINK GROUP ANG**

Verify Exclusive LAG Configuration

Issue the QUERY PORT GROUP command to verify the local VSwitch port group connectivity. Figure 47 on page 122 indicates the following information about the configuration:

- VSwitch **PEGGY** has an active Uplink port connection to the LAG through exclusive port group **ANG**.

- Status is displayed for the two OSA-Express port connections for VSwitch PEGGY as far as their LACP communications with the partner switch. Each port (OSA-Express adapter) in the port group is actively participating in LACP communications along with its role of transferring production network traffic.

```
Q PORT GROUP ANG

 Group: ANG       Active    LACP Mode: Active   Exclusive
  VSWITCH SYSTEM PEGGY                          ifIndex:  2112
  Load Balancing: Independent     Interval: 300
  RDEV: 7210.P01 VDEV: 7210 Controller: DTCVSW1  ACTIVE
        Adapter ID: 296400081657.01C8
    Uplink Port Connection:
      MAC address: 02-33-90-00-00-12
      RX Packets: 0          Discarded: 29         Errors: 0
      TX Packets: 9          Discarded: 0          Errors: 0
      RX Bytes: 0                    TX Bytes: 1404
      Device: 7210 Unit: 000    Role: DATA       Post: 2049
      Partner Switch Capabilities: No_Reflective_Relay
  RDEV: 7220.P00 VDEV: 7220 Controller: DTCVSW2  ACTIVE
        Adapter ID: 296400081657.01F8
    Uplink Port Connection:
      MAC address: 02-33-90-00-00-13
      RX Packets: 0          Discarded: 20         Errors: 0
      TX Packets: 4          Discarded: 0          Errors: 0
      RX Bytes: 0                    TX Bytes: 624
      Device: 7220 Unit: 000    Role: DATA       Post: 2050
      Partner Switch Capabilities: No_Reflective_Relay
Ready; T=0.01/0.01 15:20:56
```

Figure 47. Query of Active Exclusive Port Group ANG

For a more extensive view of VSwitch PEGGY port group connectivity, issue the QUERY PORT GROUP command with the **DETAILS** option:

```
Q PORT GROUP ANG DETAILS

 Group: ANG       Active    LACP Mode: Active   Exclusive
  VSWITCH SYSTEM PEGGY                          ifIndex:  2112
  Load Balancing: Independent     Interval: 300
  GROUP Information:
    PORT Information - Total Frames per Interval:
      Device  Status    Previous      Current
       7210   Active    61            4
       7220   Active    26            12
    ROUTING Information - Frame Distribution per Interval:
      MAC     Device    Previous      Current
       0      7210      0             0
       1      7220      0             0
       2      7210      0             0
       3      7220      0             0
       4      7210      0             0
       5      7220      0             0
       6      7210      0             0
       7      7220      0             0
  RDEV: 7210.P01 VDEV: 7210 Controller: DTCVSW1  ACTIVE
        Adapter ID: 296400081657.01C8
    Uplink Port Connection:
      MAC address: 02-33-90-00-00-12
      RX Packets: 0          Discarded: 29         Errors: 0
      TX Packets: 9          Discarded: 0          Errors: 0
      RX Bytes: 0                    TX Bytes: 1404
      Device: 7210 Unit: 000    Role: DATA       Post: 2049
      Partner Switch Capabilities: No_Reflective_Relay
    PROTOCOL Counters:
      LACP RX: 32            Marker RX: 0
      LCAP TX: 5             Marker TX: 4          Timeouts: 0
    ACTOR Information:
      System ID: 32768,02-33-90-00-00-11        Oper  Key: 2
      Port Priority: 32768   Port: 2049         Group Key: 2
      State: 3D - LACP_Active   Slow AGG SYNC DIST COLL
    PARTNER Information:
      System ID: 32768,74-99-75-35-DF-00        Oper  Key: 32
      Port Priority: 32768   Port: 0026         Group Key: 32
      State: 3D - LACP_Active   Slow AGG SYNC DIST COLL
  RDEV: 7220.P00 VDEV: 7220 Controller: DTCVSW2  ACTIVE
        Adapter ID: 296400081657.01F8
    Uplink Port Connection:
      MAC address: 02-33-90-00-00-13
      RX Packets: 0          Discarded: 23         Errors: 0
      TX Packets: 4          Discarded: 0          Errors: 0
      RX Bytes: 0                    TX Bytes: 624
      Device: 7220 Unit: 000    Role: DATA       Post: 2050
      Partner Switch Capabilities: No_Reflective_Relay
    PROTOCOL Counters:
      LACP RX: 15            Marker RX: 0
      LCAP TX: 4             Marker TX: 0          Timeouts: 0
    ACTOR Information:
      System ID: 32768,02-33-90-00-00-11        Oper  Key: 2
      Port Priority: 32768   Port: 2050         Group Key: 2
      State: 3D - LACP_Active   Slow AGG SYNC DIST COLL
    PARTNER Information:
      System ID: 32768,74-99-75-35-DF-00        Oper  Key: 32
      Port Priority: 32768   Port: 0017         Group Key: 32
      State: 3D - LACP_Active   Slow AGG SYNC DIST COLL
Ready; T=0.01/0.01 15:21:00
```

Figure 48. Display of Query Port Group ANG Details

Multi-VSwitch LAG Configuration

This support provides a Link Aggregation Control Protocol (LACP) virtualization where two or more virtual switches are made to appear to the partner switch as a single virtual switch with a single link aggregation control plane. Multi-VSwitch LAG support is totally transparent to its physical partner switches that comprise the LAG.

A multi-VSwitch LAG configuration includes elements that an exclusive LAG configuration does not require. A Multi-VSwitch LAG configuration includes the following elements:

- LACP enhancements

- Automatic OSA device allocation
- IVL domain/network configuration
- Inter-VSwitch Link (IVL) – Provides the required control and data plan communications that connect the z/VM system to the domain
- Global virtual switch definitions – required to deploy shared port groups
- Shared port group configuration

The IVL and Global VSwitch technologies and their configuration are explained in more detail in Chapter 9, “Global Virtual Switch,” on page 135.

The following topics describe elements of Multi-VSwitch LAGs and provide an example of building a Multi-VSwitch LAG and sharing a LAG within the same z/VM system.

Link Aggregation Control Protocol Enhancements

Multi-VSwitch LAG expands the LACP management function (LAG Port Controller) allowing multiple virtual switches to participate in LACP operations. No single VSwitch is totally responsible for managing the physical LAG. LACP management can be distributed across any sharing virtual switch, but a specific port within the group is only managed by one virtual switch at any point in time. It is this virtualization which makes it appear to the partner switch that there is only a single switch on the other side of his LAG. As a result, a LAG Port Controller can operate in one of two possible modes. When the virtual switch network connection is providing LACP management for the physical port it is the *active* LAG Port Controller, otherwise it's a *standby* LAG Port Controller. There is only one active LAG Port Controller per OSA-Express adapter in the port group, at any given point in time.

A standby LAG Port Controller is prepared at any point to become an active LAG Port Controller. If for any reason the active LAG Port Controller's network connection becomes inoperable, one of the standby LAG Port Controllers on the same OSA-Express adapter will be selected by the OSA-Express adapter to take-over as the active LAG Port Controller for the physical port. The ability to distribute the active LAG Port Controller across any of the VSwitches sharing the LAG prevents any one virtual switch from being a single point of failure. If the network connection or the virtual switch currently providing LACP management goes down, LACP management for a physical port will be picked up automatically by another virtual switch.

The QUERY PORT GROUP or QUERY VSWITCH command displays the current operation mode of each LAG Port Controller in the port group as shown in Figure 52 on page 128.

Automatic OSA Device Allocation

Whenever a shared port group is created, an instance of the port group is created on the z/VM system the command was issued and also created (propagated) in each active IVL member (instance 0). By specifying a device number for a shared port group you are actually selecting an OSA adapter; not a specific device. To select an OSA adapter, specify a single device number configured for the target OSA adapter (PCHID) and the port number (for an OSA-Express multiport adapter). The device number that is selected by z/VM might be different from the device number that you specified on the SET PORT GROUP JOIN command.

The scope of OSA device allocation for a shared port group is as follows:

- When an OSA adapter is added to a SHARED Port Group, all devices on the same PCHID will be RESERVED for system use (SYSTEM device) and will be propagated to every z/VM system in the IVL domain.
- z/VM will allocate a device from the SYSTEM list when adding (JOIN) a device to a SHARED Port Group or an additional instance of a group.
- All device numbers (subchannels) used within a SHARED Port Group are allocated by z/VM and not necessarily the ones specified on a SET PORT GROUP JOIN command.
- A SYSTEM device cannot be attached or dedicated to a virtual machine.

- All devices on the same OSA adapter (PCHID) must be removed from all instances of a SHARED Port Group before they can be attached to a virtual machine.

Building a Multi-VSwitch LAG Configuration

The following topics provide an example of configuring a multi-VSwitch LAG.

Note: The example uses an OSA-Express 1G multi-port adapter. The configuration for all OSA-Express and Network Express adapters is similar.

The instructions in the following topics yield the configuration that is pictured in Figure 49 on page 125. The shared port group **ANG** provides the external LAG connectivity for the global virtual switch **PEGGY**.

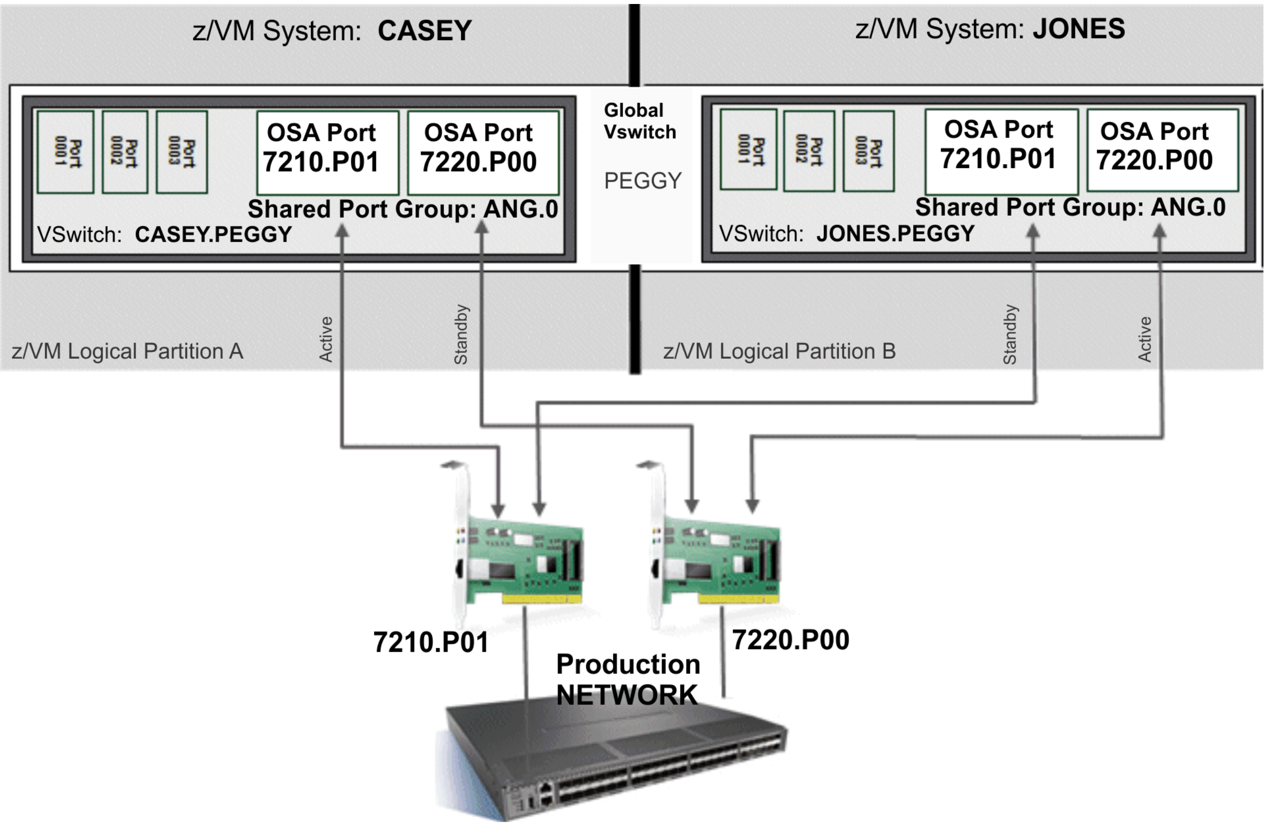


Figure 49. Global VSwitches that Share a LAG Port Group

Configure an IVL Domain

The IVL Domain is required in order to deploy a Multi-VSwitch LAG. This network must be configured prior to constructing a shared LAG port group configuration. See Chapter 9, “Global Virtual Switch,” on page 135 for how this is done.

Configure the Partner Switch

Once the IVL Domain is configured, the next step is to configure a LAG on the target physical partner switch. Each switch manufacturer has its own unique method of configuring a LACP LAG configuration.

Define Global VSwitch(s)

Once you have decided the required virtual switches for your LAG configuration and are ready to define them, remember that these virtual switch(s) must be defined with a GLOBAL scope to be configured with a shared port group as its Uplink port. Configure the global VSwitch(s) prior to constructing a shared LAG port group configuration. See Chapter 9, “Global Virtual Switch,” on page 135 for how this is done.

Determine what z/VM systems you want the global VSwitch(s) to exist and then manually define them in each system.

Note: Note; if you are migrating from an *exclusive* LAG configuration and you plan to use the same VSwitch names, then you must first delete the existing VSwitches from all z/VM systems in the IVL domain and then define the new global VSwitches.

Create a Shared Port Group

All z/VM systems sharing the same physical LAG must be members of the same IVL Domain. This provides the ability for the shared port group to be created from any one of these z/VM systems.

When sharing a port group across more than one z/VM system, you need to insure that each system that is to share the group has sufficient OSA-Express adapters for the instantiation of the shared port group.

As shown in Figure 50 on page 126, when a shared port group ANG is defined in z/VM system CASEY, the shared port group will be propagated to all z/VM systems in the IVL domain which in this case includes z/VM system JONES.

Note: The example uses an OSA-Express 1G multi-port adapter. The configuration for all OSA-Express and Network Express adapters is similar.

- **SET PORT GROUP ANG LACP ACTIVE SHARED**
  - To participate in a Multi-VSwitch LAG Configuration, the SET PORT GROUP command must be created with a global scope by specifying: **LACP ACTIVE SHARED**.
  - The z/VM System will automatically propagate an instance of Shared Port Group ANG to all active IVL Members in the same IVL domain.
- **SET PORT GROUP ANG JOIN *7210.P01 7220.P00***
  - Identify the OSA-Express adapters comprising the physical LAG.
  - Choose a single device on each OSA-Express adapter that will be used to identify the feature and reserve all devices for use by the port group.
  - The z/VM System will select the actual device numbers to be used on the target feature.
  - The z/VM System will automatically add the OSA-Express adapters to the physical LAG.

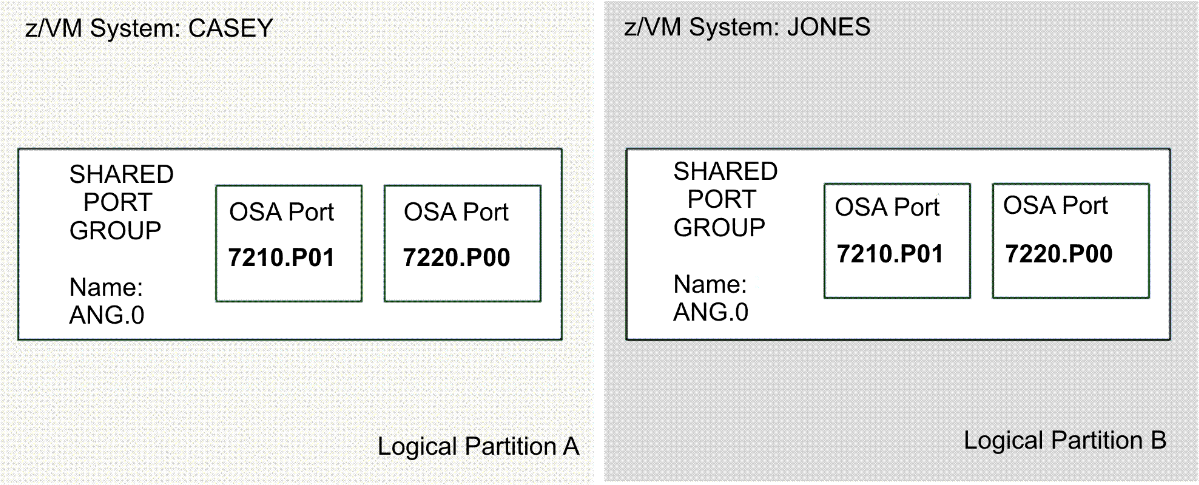


Figure 50. Shared Port Group ANG

- **Verify port group ANG by issuing a QUERY PORT GROUP ANG DET command**
  - We can ascertain from Figure 51 on page 127 that the port group ANG is successfully shared because the scope is synchronized on both the z/VM systems CASEY and JONES.
  - At this point the port group has not been configured as an Uplink port of a global VSwitch as shown by a Mode of "Inactive" on both the z/VM systems CASEY and JONES. If shared port group ANG was configured to a virtual switch, the Mode would be connected.

- The member section of the query output lists the current active member systems in the IVL domain. Remember that a shared port group was either defined or propagated to all member systems in the IVL domain.

```
Q PORT GROUP ANG DETAILS

  Group: ANG.0      Inactive LACP Mode: Active   Shared
    VSWITCH <none>                               ifIndex: 2112
    Load Balancing: Collaborative    Interval: 300
    RDEV: 7210.P01   Adapter ID: 29640007E386.0250
    RDEV: 7220.P00   Adapter ID: 29640007E386.0250
   Member:  CASEY                    (IVL Member Section)
        Scope: Synchronized
        LAG Synchronization token: CE07E6D8FD203468
        Mode:  Inactive
   Member:  JONES
        Scope: Synchronized
        LAG Synchronization token: CE07E6D8FD203468
        Mode:  Inactive
Ready; T=0.01/0.01 10:03:50
```

Figure 51. Query Port Group Display for ANG

Configure the Shared Port Group to Global VSwitches

- If the global VSwitch has not been previously defined, then issue a **DEFINE VSWITCH PEGGY GLOBAL ETHERNET** on each system.

  Note: The **ETHERNET** operand is required for VSwitches configured for a LAG configuration.

The SET VSWITCH command must be issued for each member of the Global VSwitch that is intended to share port group ANG.

For our configuration the following command is issued on z/VM systems CASEY and JONES to configure the Uplink option on the existing global VSwitch PEGGY:

- **SET VSWITCH PEGGY UPLINK GROUP ANG**

  Note: When the IVL, global VSwitches and shared port groups are defined in the system configuration file, the actual instantiation of the global VSwitches and shared port groups will be deferred until the IVL is fully operational.

Verify Shared LAG Configuration

Issue the QUERY PORT GROUP command to verify the local VSwitch port group connectivity. Even though this query display is specifically for VSwitch CASEY.PEGGY, the query display for VSwitch JONES.PEGGY would present similar information. The following observations may be drawn from Figure 52 on page 128:

- The query response shows that VSwitch **CASEY.PEGGY** has an active Uplink port connection to the LAG through shared port group ANG.
- Status is displayed for the two OSA-Express port connections for VSwitch PEGGY as far as their LACP communications with the partner switch. Each port (OSA-Express adapter) in the port group also has a LAG Port Controller along with its role of transferring production network traffic.
  - Device 7210.P01 is designated by the OSA-Express adapter as the ***ACTIVE*** LAG Port Controller. This indicates that this port is actively participating in LACP control communications with the partner switch.
  - Device 7220.P00 is designated by the OSA-Express adapter as the ***STANDBY*** LAG Port Controller. This port is not actively communicating with the partner switch but is passively monitoring all inbound LACP communication. This device will be ready in the event the OSA-Express adapter changes it to the active LAG Port Controller. This may occur during failover recovery operations.

```
Q PORT GROUP ANG

  Group: ANG.0        Active    LACP Mode: Active   Shared
    VSWITCH CASEY.PEGGY                             ifIndex: 2112
    Load Balancing: Collaborative    Interval: 300
    RDEV: 7210.P01  VDEV: 7210 Controller: DTCVSW1  ACTIVE
          Adapter ID: 29640007E386.0250
      Uplink Port Connections:
        MAC address: 02-11-11-00-00-03
        RX Packets: 0          Discarded: 29          Errors: 0
        TX Packets: 32         Discarded: 0           Errors: 0
        RX Bytes: 0                    TX Bytes: 4992
        Device: 7210  Unit: 000   Role: DATA       Port: 2049
        Partner Switch Capabilities: No_Reflective_Relay
        LAG Port Controller: Active             <-----------(LAG Port Controller
    RDEV: 7220.P00  VDEV: 7220 Controller: DTCVSW2  ACTIVE             Status)
          Adapter ID: 29640007E386.0251
      Uplink Port Connections:
        MAC address: 02-11-11-00-00-04
        RX Packets: 0          Discarded: 700         Errors: 0
        TX Packets: 27         Discarded: 0           Errors: 0
        RX Bytes: 0                    TX Bytes: 4212
        Device: 7220  Unit: 000   Role: DATA       Port: 2050
        Partner Switch Capabilities: No_Reflective_Relay
        LAG Port Controller: Standby            <-----------(LAG Port Controller
   Ready; T=0.01/0.01 10:37:10                                       Status)
```

Figure 52. VSwitch Connections to a Shared Port Group

For a more extensive view of VSwitch CASEY.PEGGY port group connectivity, issue the QUERY PORT GROUP command with the DETAILS option:

- Towards the bottom of this query display, Figure 53 on page 129, we observe that the Mode for port group ANG on each z/VM system has changed from Inactive to Connected because Global VSwitch PEGGY has been connected to the shared port group ANG.

```
Q PORT GROUP ANG DETAILS

 Group: ANG.0      Active    LACP Mode: Active   Shared
  VSWITCH CASEY.PEGGY                            ifIndex: 2112
  Load Balancing: Collaborative   Interval: 300
  GROUP Information:
    PORT Information - Total Frames per Interval:
      Device  Status     Previous      Current
       7210   Active     91            145
       7220   Active     383           424
    ROUTING Information - Frame Distribution per Interval:
      MAC     Device     Previous      Current
       0      7210       0             0
       1      7220       0             0
       2      7210       0             0
       3      7220       0             0
       4      7210       0             0
       5      7220       0             0
       6      7210       0             0
       7      7220       0             0
  RDEV: 7210.P01 VDEV: 7210 Controller: DTCVSW1  ACTIVE
        Adapter ID: 296400081657.0250
    Uplink Port Connection:
      MAC address: 02-11-11-00-00-03
      RX Packets: 0          Discarded: 29         Errors: 0
      TX Packets: 50         Discarded: 0          Errors: 0
      RX Bytes: 0                    TX Bytes: 7800
      Device: 7210 Unit: 000    Role: DATA       Post: 2049
      Partner Switch Capabilities: No_Reflective_Relay
      LAG Port Controller: Active
    PROTOCOL Counters:
      LACP RX: 288           Marker RX: 0
      LCAP TX: 46            Marker TX: 4          Timeouts: 0
    ACTOR Information:
      System ID: 32768,02-11-11-00-00-02        Oper  Key: 2
      Port Priority: 32768   Port: 2049         Group Key: 2
      State: 3D - LACP_Active   Slow AGG SYNC DIST COLL
    PARTNER Information:
      System ID: 32768,74-99-75-35-DF-00        Oper  Key: 30
      Port Priority: 32768   Port: 0026         Group Key: 30
      State: 3D - LACP_Active   Slow AGG SYNC DIST COLL
  RDEV: 7220.P00 VDEV: 7220 Controller: DTCVSW2  ACTIVE
        Adapter ID: 296400081657.0251
    Uplink Port Connection:
      MAC address: 02-11-11-00-00-04
      RX Packets: 0          Discarded: 1213       Errors: 0
      TX Packets: 45         Discarded: 0          Errors: 0
      RX Bytes: 0                    TX Bytes: 7020
      Device: 7220 Unit: 000    Role: DATA       Post: 2050
      Partner Switch Capabilities: No_Reflective_Relay
      LAG Port Controller: Active
    PROTOCOL Counters:
      LACP RX: 256           Marker RX: 0
      LCAP TX: 45            Marker TX: 0          Timeouts: 0
    ACTOR Information:
      System ID: 32768,02-11-11-00-00-02        Oper  Key: 2
      Port Priority: 32768   Port: 2050         Group Key: 2
      State: 3D - LACP_Active   Slow AGG SYNC DIST COLL
    PARTNER Information:
      System ID: 32768,74-99-75-35-DF-00        Oper  Key: 30
      Port Priority: 32768   Port: 0005         Group Key: 30
      State: 3D - LACP_Active   Slow AGG SYNC DIST COLL
 Member:  CASEY
      Scope: Synchronized
      LAG Synchronization token: CE07E6D8FD203468
      Mode:  Connected
 Member:  JONES
      Scope: Synchronized
      LAG Synchronization token: CE07E6D8FD203468
      Mode:  Connected
Ready; T=0.01/0.01 10:49:22
```

Figure 53. Display of Query Shared Port Group ANG Details

Example of Sharing a LAG within the same z/VM System

It is possible for up to four different global VSwitches to share the same port group within a single z/VM system.

Note: The example uses an OSA-Express 1G multi-port adapter. The configuration for all OSA-Express and Network Express adapters is similar.

- **DEFINE VSWITCH RICK GLOBAL ETHERNET GROUP ANG**

- As depicted in Figure 54 on page 130, another instance of a Shared Port Group LAG is created (ANG.1) when it is configured to virtual switch RICK.
- z/VM will automatically allocate an OSA-Express triplet for each feature within in the group from the available devices in the z/VM system.
- Port Group name is increased up to 10 characters.
  - 8 Character Name specified by the system administrator
  - 2 Character Instance ID assigned by z/VM
    - Default: name.0
    - Instance Number not used on SET PORT GROUP command
- ANG.0 is the base instance of a Shared Port Group and is the only instance propagated to other IVL Members within the same domain.
- Instance one to three (non-base) remain local to the system where it is being used.
- The only difference between the base and its other instances are the device numbers allocated for each OSA-Express adapter within the LAG.
- Global VSwitch RICK is only configured on system CASEY.

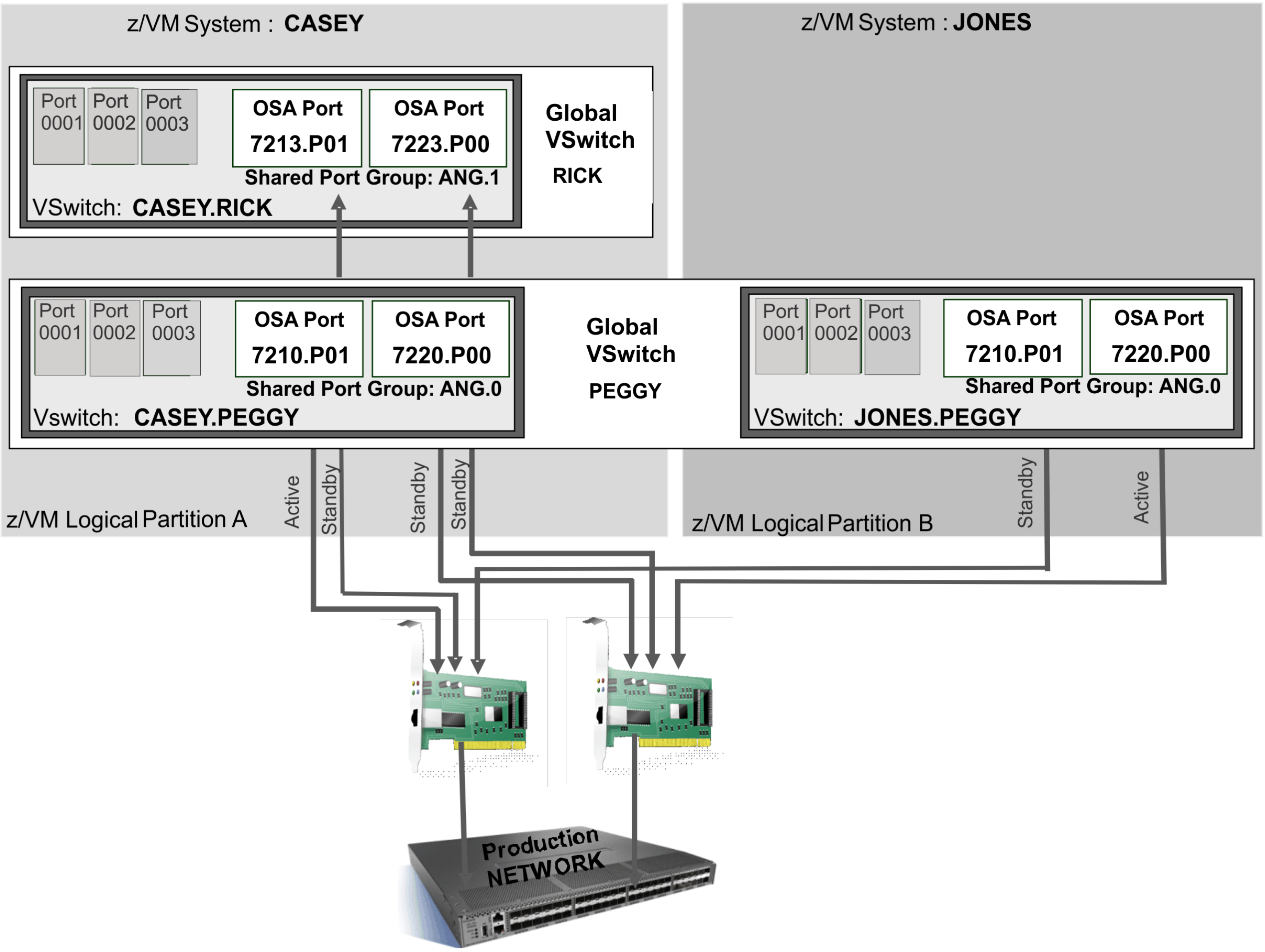


Figure 54. Sharing a Port Group within the same z/VM System

We see in Figure 54 on page 130, each OSA-Express adapter has one active LAG Port Controller and one or more standby LAG Port Controllers. This is a valid LAG configuration because each OSA-Express adapter must have one active LAG Port Controller for LACP communications with the partner switch. So, active LAG Port Controller assignments are made on a per OSA-Express adapter basis not a port group basis. The LAG Port Controller assignments are shown in the QUERY PORT GROUP display as shown in Figure 52 on page 128.

When a global VSwitch with an active LAG Port Controller is disconnected from the network, another global VSwitch with a standby LAG Port Controller will be selected by the OSA-Express adapter to take on the role as an active LAG Port Controller.

Take-Over and Take-Back Recovery for a Multi-VSwitch LAG Configuration

Multi-VSwitch LAG provides special recovery operations when a global VSwitch loses connectivity to an OSA-Express adapter in the LAG group. This recovery is for an unexpected loss of connectivity between a **specific** global VSwitch and the OSA-Express adapter. The OSA-Express adapter is still communicating with the partner switch and the remaining sharing global VSwitch connections. The recovery operations for this event are twofold:

1. If this global VSwitch connection is the active LAG Port Controller, the OSA-Express adapter selects a standby LAG Port Controller to be the active LAG Port Controller. When a standby LAG Port Controller incurs this type of error, its LAG Port Controller role is unchanged and will resume as a standby LAG Port Controller when the connection between the global VSwitch and the OSA-Express adapter is recovered.
2. The partner switch will continue to send traffic to the z/VM guests that are connected to the global VSwitch which no longer has a connection to the OSA-Express adapter. Likewise the z/VM guests will continue to send traffic to MAC destinations serviced by the physical switch. The IVL will be used to transport this network traffic through a functional connection of the same OSA-Express adapter until the original connection has been recovered.

See "Take-Over and Take-Back Operations" on page 143, for more details on the specific operational IVL recovery actions that are performed.

Load Balancing within the Virtual Switch

All active OSA-Express ports within a virtual switch port group are used in the transmission of data between the z/VM virtual switch and the connected physical switch. The virtual switch logic will attempt to distribute the load across all the ports within the group. The actual load balancing achieved will depend on the frame rate of the different conversations taking place across the OSA-Express ports and the load balance interval specified by the SET PORT GROUP INTERVAL command as to how well the load is balanced.

From the perspective of the virtual switch, a conversation is based on the source and destination MAC addresses of a frame being transmitted. All frames from a virtual NIC destined to the same destination MAC are considered the same conversation. All the outbound frames associated with this conversation can be routed automatically to any one of the active OSA-Express ports within the group. This is the smallest unit of work which can be moved from one port to another to balance the workload.

The frequency in which a load balance operation will move a conversation is based on the interval specified by the SET PORT GROUP INTERVAL command. The default load balance interval will look at the number of frames received and transmitted across each active port within the last five minutes. If there is more than a ten percent frame rate difference between the most and least utilized OSA-Express port, an attempt will be made to move a conversation from a virtual NIC currently routed to the most utilized OSA-Express port to the least utilized port. The conversation selected will be the one that will make the frame rates between these two ports as equal as possible. Of course this also assumes the frame rate remains relatively the same in the next load balance interval. After moving work load on the first pair of devices, the logic will then select the next most and least utilized ports and perform the load balance operation. This is then repeated for all the remaining OSA-Express features in the port group.

Lowering the SET PORT GROUP INTERVAL value increases the ability to balance the load across the active OSA-Express ports. However, decreasing the interval time does increase the number of times the network workload must be examined by CP, thus increasing the CPU overhead to manage the virtual switch. Selecting the largest interval value to meet your network load is the best way to go. There is also the ability to turn off load balancing for a specific port group anytime by issuing the SET PORT GROUP INTERVAL OFF command. Likewise, it can also be turned back on to any allowable interval at anytime.

The number of frames transmitted for a specific OSA-Express port in both the current and previous load balance interval can be seen at anytime by issuing a QUERY PORT GROUP DETAILS command. The “PORT Information” section of the response displays a list of the active ports within the group in addition to the load balance frame counts. Each frame count includes the sum of both inbound and outbound frames transmitted by the OSA-Express port within the load balance interval. In addition, the “ROUTING Information” section of the same response displays the current routing of outbound frames for the ports within the group. Each row displayed corresponds to the least three significant bits of the destination MAC for outbound frames. The “Device” column indicates the OSA-Express port currently assigned to handle frames with the same last three destination MAC bits and the number of frames sent out across that link in both the previous and current load balance interval. The following is an example of the QUERY PORT GROUP DETAILS response:

```
Group: LINKAG     Active       LACP Mode: Active   Exclusive
  VSWITCH SYSTEM TEST                              ifIndex: 2112
  Load Balancing: Independent       Interval: 300
  GROUP Information:
    PORT Information - Total Frames per Interval:
      Device  Status     Previous      Current
       0510   Active     10954         4
       0520   Active     10900         4
    ROUTING Information - Frame Distribution per Interval:
      MAC     Device     Previous      Current
       0      0510       5200          4
       1      0520       0             0
       2      0510       0             0
       3      0520       0             0
       4      0510       0             0
       5      0520       0             0
       6      0510       0             0
       7      0520       5303          4
  RDEV: 0510.P00 VDEV: 0510 Controller: DTCVSW1  ACTIVE
    VSWITCH Connection:
      MAC address: 02-00-00-00-00-03
      RX Packets: 4          Discarded: 0          Errors: 0
      TX Packets: 4          Discarded: 0          Errors: 0
      RX Bytes: 512                  TX Bytes: 640
      Device: 0512  Unit: 002   Role: DATA      vPort: 0001  Index: 0001
  RDEV: 0520.P00 VDEV: 0520 Controller: DTCVSW2  ACTIVE
    VSWITCH Connection:
      MAC address: 02-00-00-00-00-04
      RX Packets: 4          Discarded: 0          Errors: 0
      TX Packets: 4          Discarded: 0          Errors: 0
      RX Bytes: 512                  TX Bytes: 640
      Device: 0522  Unit: 002   Role: DATA      vPort: 0002  Index: 0002
 Backup Devices:
  RDEV: 0B00.P00 VDEV: 0B00 Controller: DTCVSW1    BACKUP
```

Management Scope of a Load Balance Operation

The options available to coordinate a load balance operation across a port group depend on whether the port group is exclusive to a VSwitch or shared by multiple VSwitches. If the port group was created as EXCLUSIVE by the SET PORT GROUP command, then only a single VSwitch can use or connect to the port group. As an exclusive port group, the only management option available is an Independent Load Balance.

With a SHARED Port Group, there are two different management options available to coordinate a load balance operation across the port group. In this configuration, multiple VSwitches can share all the OSA-Express features configured within the port group on the same or across multiple z/VM Systems. Since multiple VSwitches are involved with the same port group, the system administrator has the option to determine how and who is responsible for performing a load balance operation on a shared port group. By default, a shared port group is managed using a Collaborative Load Balance method, but it can be changed to use an Independent Load Balance. The following is a description of each load balance method:

Independent Load Balance Management

Each VSwitch using a Link Aggregated Group (LAG) to provide connectivity to an external network is responsible for balancing its own bandwidth across all the OSA-Express features within the group. No consideration is given to any other VSwitch which may also be sharing the same port group when moving a guest's outbound workload from one OSA-Express feature to another port.

A single VSwitch view is most efficient and is the only method allowed when operating the OSA-Express features within a port group in exclusive mode. While in exclusive mode, no other LPAR, VSwitch or guest can share any of OSA-Express features. Only the connected VSwitch can send or receive data from the ports within an exclusive LAG. Therefore, for an exclusive configuration it’s simple to immediately move workload from one port to another port, especially since there is only a single VSwitch using and managing workload within the port group.

When using Independent Load Balance management with a SHARED port group, the system administrator can set the load balance interval via the SET PORT GROUP INTERVAL command independently on each z/VM System. Given each VSwitch is coordinating its own load balance operation, this provides increased flexibility to manage load balance operations independently on each z/VM System.

Collaborative Load Balance Management

When multiple VSwitches are sharing the same port group as in the case of a Multi-VSwitch LAG configuration, there could be multiple VSwitches attempting to balance their user's workload to the external network. For a Multi-VSwitch LAG Configuration, having each VSwitch sharing the port group independently managing the balancing of their own workload is not as efficient when multiple VSwitches are sharing the same LAG.

With a Collaborative Load Balance management method, consideration is given to all VSwitches sharing the same port group. Instead of each VSwitch coordinating a load balance operation independently, only one of the VSwitches sharing the port group takes on this responsibility. This insures one VSwitch doesn’t undo or overdo workload changes made by any other VSwitch sharing the same port group.

The VSwitch selected as the coordinating VSwitch is always determined by z/VM. However, a system administrator can influence z/VM’s choice when multiple z/VM Systems are sharing the same port group when configuring the system’s MACPREFIX value via the VMLAN statement in the system configuration file. The z/VM System configured with the highest MACPREFIX value currently connected and active within the IVL domain selects one of its VSwitches (sharing the port group) as the coordinating VSwitch. The system selected as the coordinator is included in the “Last Load Balance” response of a QUERY Port GROUP command for a SHARED Port Group.

When using Collaborative Load Balance management, the system administrator can only set a single load balance interval via the SET PORT GROUP INTERVAL command. The interval value specified will automatically be set on every z/VM System active within the same IVL Network sharing the same port group. Since only a single VSwitch is coordinating the load balance operation, this will insure whatever VSwitch takes on the role of a coordinating VSwitch will always use the same load balance interval.

When a load balance operation is triggered, it is the coordinating VSwitch’s responsibility to collaborate and manage the entire operation. This first step includes acquiring each sharing VSwitch’s current frame usage by port and determining whether there is even a need to modify the current workload for the current load balance interval. If the coordinating VSwitch determines a change is warranted, it will first attempt to make the change on his VSwitch. If the coordinating VSwitch can’t balance the load from his own VSwitch, it will ask each sharing VSwitch one at time to see whether the requested amount of workload can be moved on their VSwitch.

Although you can configure a port group for either Independent Load Balance or Collaborative Load Balance management at any time, Collaborative Load Balance management can only be exploited when all currently active IVL Members support Collaborative Load Balance management. If a down level z/VM System is actively connected to the IVL domain, then all port groups configured for Collaborative Load Balance management with be forced to use Independent Load Balance Management automatically. If this condition occurs, a QUERY PORT GROUP command will change Load Balancing from “Collaborative” to “Forced Independent”. When there is no longer a down level system actively connected to the IVL

Network, then all Forced Independent port groups will be changed to Collaborative automatically. At this point, a Collaborative Load Balance will be performed on the next load balance interval.

Chapter 9. Global Virtual Switch

A global virtual switch is a logical grouping of virtual switches across multiple z/VM systems. The virtual switches share identical networking characteristics, including name, type, and transport. These switches communicate via an Inter-VSwitch Link (IVL) network to form a single logical switch with centralized uplink port management.

A global virtual switch allows multiple virtual machines to connect to a shared virtual network, similar to how physical machines connect to a physical Ethernet switch. It supports VLANs, Quality of service (QoS), port isolation, and MAC address learning, just like a physical switch.

Global virtual switches are the only virtual switches that can be configured with a shared port group in a LAG configuration (known as Multi-VSwitch LAG). The IVL connectivity allows all sharing global virtual switches to both manage and share the same physical LAG by exchanging LACP information, status, control, and load balancing information with each other. See Chapter 8, “Virtual Switch Link Aggregation,” on page 113 for more information on Multi-VSwitch LAG.

Benefits of global virtual switches

Global virtual switches provide the following benefits:

- **Enhanced network performance**

  Global virtual switches operate at memory speeds, which enables faster data transfer between virtual machines and external networks. Latency is reduced because there is no requirement to route data through the z/VM TCP/IP stack.

- **Simplified network management**

  Network administrators can manage virtual networking centrally, similar to how physical switches are managed. IEEE 802.1Q VLAN tagging is supported, which enables logical separation of network traffic for security and organization.

- **Improved scalability**

  Global virtual switches can handle thousands of vNICs (virtual network interface cards). Virtual NICs can be added or removed without disrupting the system, which supports dynamic configuration for dynamic workloads.

- **Security and isolation**

  Global virtual switches can enforce MAC address policies to prevent spoofing. VLANs and other features help isolate traffic between different tenants or workloads.

- **Integration with external networks**

  Global virtual switches can connect virtual networks to external LANs, which enables seamless integration with enterprise infrastructure. Global virtual switches work well with external switches and routers, which supports hybrid environments.

- **Monitoring and diagnostics**

  Global virtual switches support tools for monitoring and analyzing network traffic, which aid in performance tuning and troubleshooting. Global virtual switches support SNMP, which enables integration with enterprise network management systems.

Requirements

The instantiation of a global virtual switch is performed by specifying the GLOBAL operand on the DEFINE VSWITCH Command or Configuration Statement. The GLOBAL operand is used to include a virtual switch as a member of the global virtual switch. The same virtual switch name must be assigned by a system administrator to the collection of virtual switches which share the same network characteristics as documented earlier in this section. All virtual switches defined with the GLOBAL operand and same virtual switch name are members of the global virtual switch. The eight byte virtual switch name defines a specific member within the global virtual switch.

For example: virtual switch PEGGY is a global virtual switch. The name that identifies the switch member of PEGGY on the z/VM system named CASEY is CASEY.PEGGY. Likewise the switch member that identifies PEGGY on the z/VM system named JONES is JONES.PEGGY.

The topics that follow explain how to construct a Global Virtual Switch.

- IVL (Inter-VSwitch Link) Overview
  - IVL Domain
  - IVL Virtual Switch
- Deploying an IVL Network
  - Defining an IVL Domain
- Define a Global VSwitch
- Take-Over and Take-Back Operations

IVL (Inter-VSwitch Link) Overview

The IVL is a control plane link between one global VSwitch to another whether in the same z/VM system or spanning multiple z/VM systems. The IVL minimizes administrative involvement with virtual switch management providing more autonomous operations. This link provides the ability for z/VM systems to communicate and synchronize virtual networking activities.

An IVL configuration spanning multiple images of z/VM is as transparent as possible to a system administrator. But it will still involve some administrative configuration and management specifically for private Ethernet connectivity between multiple z/VM images.

IVL Domain

The IVL Domain is a group of up to 16 z/VM images connected by an Ethernet IVL LAN segment. All the active members within an IVL Domain provide the control plane operations that support the instantiation and management of a Global VSwitch and a Shared Port Group. The control plane is comprised of both a management and data plane.

- Management plane - All IVL communications providing discovery, instantiation, synchronization and serialization.
- Data plane – IVL communications for the transport of client payload during temporary takeover error recovery events.

The system administrator must DEFINE and configure a single IVL virtual switch on each z/VM image in order to join a specific IVL Domain. See "IVL Virtual Switch" on page 137 for more details and information on the IVL virtual switch. The Uplink Port on each z/VM image's IVL VSwitch must be connected via one or more OSA adapters to the same layer 2 LAN segment. The port sharing capabilities provided by an OSA adapter are ideal for z/VM System to z/VM System connectivity. However, for high availability (fail-over), distinct OSA adapters for primary and backup should be configured.

Multiple IVL Domains are supported for a given IVL LAN segment, but a z/VM image can only belong to a single IVL Domain at any point in time. This provides the ability for a system administrator to isolate a subset of his z/VM images into two or more IVL Domains.

Note: The IVL LAN is a layer 2 (Ethernet) network. IP (Internet Protocol) is not used to forward traffic between IVL members (z/VM systems). As we will see later, z/VM will assign a unicast MAC address for each IVL member and a multicast MAC address for each IVL domain that is deployed.

One IVL Domain can be isolated from another by one of the following two methods:

1. By the assignment of an IVL virtual switch (that is a z/VM system) to a particular IVL Domain. This isolation method supports the configuration of up to eight IVL domains sharing the IVL LAN segment. Each domain is assigned a unique multicast MAC address limiting control plane broadcast to the actual

members of a particular domain. A Multicast MAC address is assigned by z/VM from its reserved pool of MAC addresses based on the “domain” configured for the z/VM system’s IVL VSwitch:

Domain	Multicast MAC Address
A	03-FF-FF-FF-FF-FF-01
B	03-FF-FF-FF-FF-FF-02
C	03-FF-FF-FF-FF-FF-03
D	03-FF-FF-FF-FF-FF-04
E	03-FF-FF-FF-FF-FF-05
F	03-FF-FF-FF-FF-FF-06
G	03-FF-FF-FF-FF-FF-07
H	03-FF-FF-FF-FF-FF-08

These assigned multicast MAC address are displayed through the QUERY VMLAN command. See Figure 58 on page 141.

2. By the assignment of a unique VLAN ID to each IVL domain sharing the IVL LAN segment. Up to 8 domains can be configured per VLAN. This number of IVL domains supported in this method of isolation is limited only by the number of distinct VLANs configured for the IVL LAN segment. To configure a VLAN ID for a z/VM system’s IVL domain connectivity, use the SET VSWITCH IVL IVLPORT command. See “Additional IVL Controls and Diagnostic Aids” on page 138.

IVL Virtual Switch

A special type of virtual switch that provides the networking infrastructure used to create a LAN segment providing private Ethernet communication between z/VM systems.

- An IVL VSwitch is created with DEFINE VSWITCH TYPE IVL either in the System CONFIG file for instantiation at IPL or with the CP command. Once an IVL VSwitch is created, it should remain active for the entire time the z/VM image is operational.
- An IVL VSwitch will be deleted at z/VM SHUTDOWN or by using the DETACH VSWITCH command. The IVL VSwitch cannot be deleted if there are Global VSwitches or Shared Port Groups defined on the z/VM system.
- Only one IVL VSwitch may be configured per z/VM system.
- The IVL VSwitch has only an IVL Port for the z/VM system which is created automatically and does not support the coupling of traditional guest simulated NICs.
- The IVL VSwitch may be connected to OSA adapters that are shared with other operating systems.
- To ensure IVL Network stability, it is highly recommended that an OSA-Express adapter is defined with PQ_ON on either a DEFINE CHPID command or via the Hardware Configuration Definition (HCD). The IVL virtual switch exploits OSA adapters' priority queuing capabilities to ensure that IVL management transmissions are delivered to the IVL Network ahead of any error recovery production traffic. For a Network Express adapter, priority queuing is always enabled, so no additional configuration is required.

Figure 55 on page 139, presents highly available IVL connectivity by sharing OSA adapters.

For networking configurations where a global VSwitch will span across multiple z/VM systems, the IVL virtual switch in each z/VM system must be connected together via one or more OSA adapters to the same layer 2 LAN segment. There is no problem with sharing the OSA adapters with other operating systems or other z/VM systems. The number of OSA adapters that are required for the IVL is solely based on the high availability and bandwidth requirements of the client.

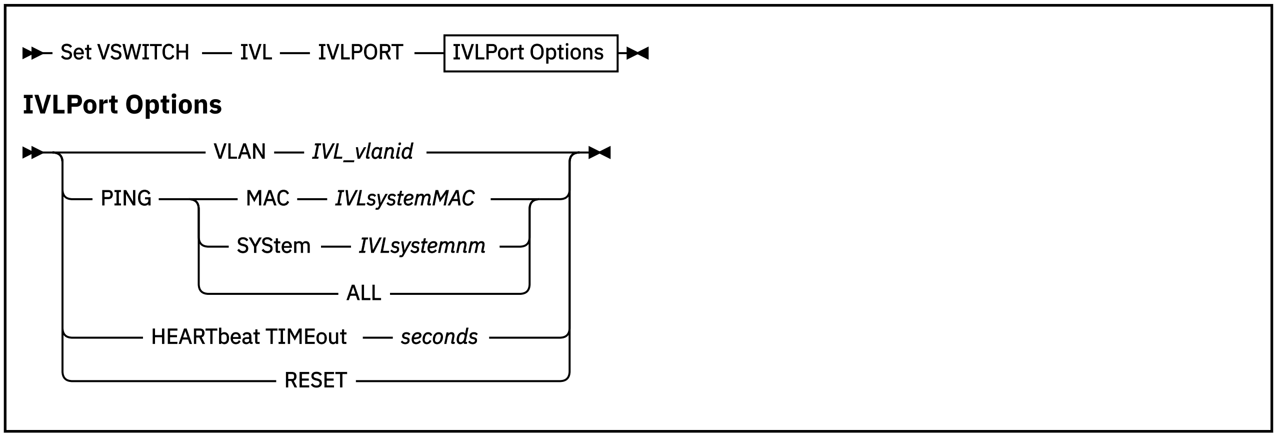
Operationally the IVL virtual switch has the following restrictions:

- Only a PORTBASED ETHERNET VSwitch is allowed

    - USERBASED not supported on DEFINE VSwitch for TYPE IVL
- VLAN AWARE and VLAN UNAWARE are supported but PORTTYPE TRUNK is not
    - Native VLAN recommendations:
        - If untagged traffic is desired then set NATIVE 1
        - If untagged traffic is not desired then set NATIVE NONE
- An IVL VSwitch cannot be defined as a GLOBAL VSwitch
- The Uplink Port is configured and managed by the system administrator
    - ACTIVE/BACKUP port configuration is supported
    - Exclusive Link Aggregation is supported
    - Multi-VSwitch Link Aggregation is not supported
    - NIC Uplink Port is not supported
- Connecting a NIC via a COUPLE command is not supported
- Uncoupling the IVL port NIC via a DETACH NIC is not supported
- Bridge Port is not supported

Additional IVL Controls and Diagnostic Aids

The IVL Port is the only port provided on the IVL VSwitch and is managed by this command:



VLAN

Sets the VLAN ID to be associated with the IVL Port and assigns the VLAN for the IVL domain. Requires IVL VSwitch to be configured as VLAN AWARE.

PING

Test connectivity between z/VM systems in the same IVL domain.

```
SET VSWITCH IVL IVLPORT PING ALL
```

HEARTBEAT TIMEOUT

Adjusts the frequency that the local z/VM system confirms with z/VM systems in the same IVL domain.

- Default is 30 seconds
- Reducing this time out will make the local heartbeat processing more sensitive to z/VM systems joining and leaving the IVL domain. The more aggressive (lower) the time out setting the more overhead is incurred by the local system.

RESET

A diagnostic tool which terminates and recreates an IVL Port connection. This option could be used if the local z/VM system is out of synchronization with the rest of the IVL domain.

Deploying an IVL Network

The following steps are the means for creating a functional IVL network.

Note: The example uses an OSA-Express 1G multi-port adapter. The configuration for all OSA-Express and Network Express adapters is similar.

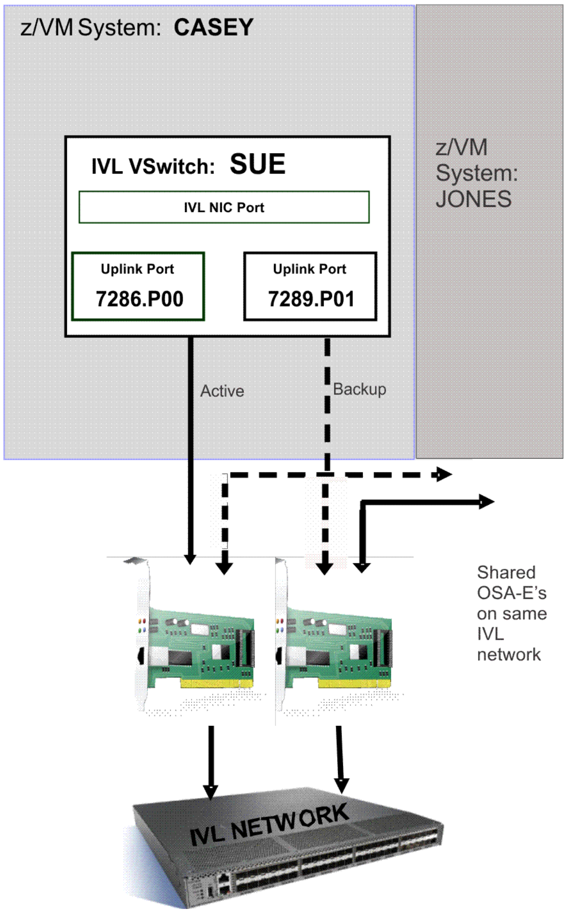


Figure 55. IVL VSwitch SUE on System CASEY

Connectivity to the IVL Network

DEFINE an IVL Virtual Switch:

- **DEFINE VSWITCH SUE TYPE IVL DOMAIN B VLAN 8 NATIVE NONE UPLINK RDEV 7286.P00 7289.P01**
  - Only one IVL VSwitch allowed per z/VM System
  - Default Domain A is selected if a domain specification is omitted
  - IVL VSwitch name may be the same or different on each system
  - VLAN 8 is assigned to the IVL Domain in this example
  - Conventional Uplink Port with backups or an Exclusive LAG
  - Recommend, but not required that two OSA-Express adapters are configured to eliminate a single point of failure
- Repeat the same setup for z/VM System JONES, but alternate the RDEVs
  - **DEFINE VSWITCH BRUCE TYPE IVL DOMAIN B VLAN 8 NATIVE NONE UPLINK RDEV 7289.P01 7286.P00**
  - **SET VSWITCH BRUCE SWITCHOVER may be used to alternate OSA-Express Uplink Ports**

IVL Network Configuration Domain B VLAN 8

As shown in Figure 56 on page 140, z/VM systems CASEY and JONES are members of IVL Domain B. They share a pair of OSA-Express adapters that maximizes utilization of the devices while also providing failover support in the event of a network failure. It is the choice of the installation to decide whether these OSA-Express adapters are allocated solely for the IVL or are allocated from an actively shared pool of devices. In support of the IVL network on actively shared OSA-features, like this example, VLANs may be deployed to isolate IVL traffic. System CASEY and System JONES may be defined on the same CEC or separate CECs.

Note: Multi-VSwitch LAG is only supported within a single CEC.

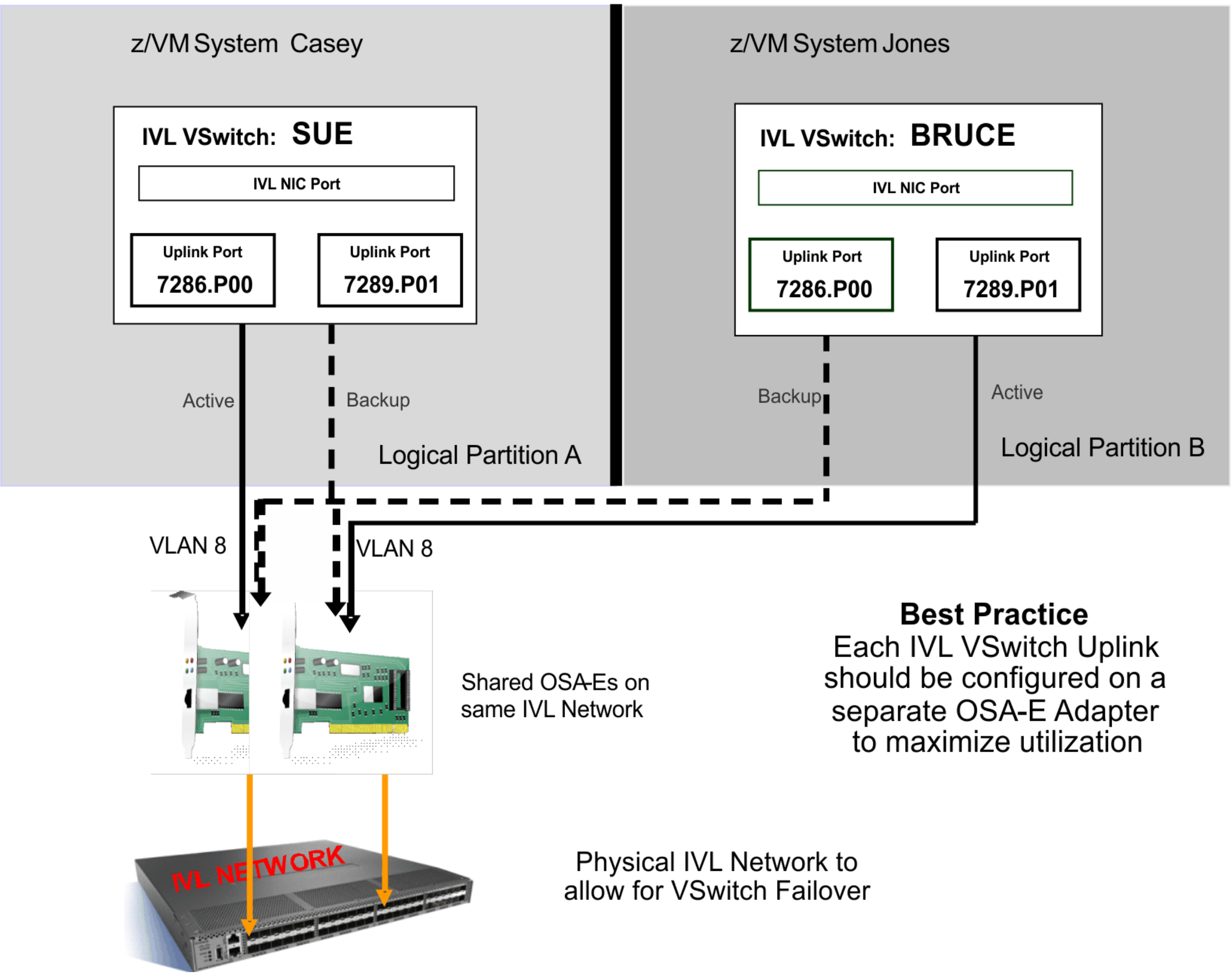


Figure 56. IVL Domain B Network

Verification of IVL Network Connectivity

Through use of the QUERY VSWITCH command in Figure 57 on page 141, we can see that IVL VSWITCH SUE has an active IVL Port. We can also see that system CASEY has been assigned a unicast MAC address of 02-11-11-00-00-00 for IVL communications with other systems. We can also determine that system CASEY is a member of IVL DOMAIN B, denoted by the multicast MAC address 03-FF-FF-FF-FF-02.

```
QUERY VSWITCH SUE

 VSWITCH SYSTEM SUE       TYPE: IVL     Connected: 0   Maxconn: INFINITE
   PERSISTENT RESTRICTED    ETHERNET                 Accounting: OFF
   PORTBASED LOCAL
   VLAN Aware  Default VLAN: 0008    Default Porttype: Access GVRP: Enabled
               Native  VLAN: 0001    VLAN Counters: OFF
   MAC address: 02-11-11-00-00-02    MAC Protection: Unspecified
   IPTimeout: 5         QueueStorage: 8
   Isolation Status: OFF        VEPA Status: OFF
   Unicast IP address count: 0
  Uplink Port:
   State: Ready
   PMTUD setting: EXTERNAL   PMTUD value: 8992     Trace Pages: 8
   RDEV: 7286.P00 VDEV: 7286 Controller: DTCVSW2  ACTIVE
         Adapter ID: 29640007E386.0370
   RDEV: 7289.P01 VDEV: 7289 Controller: DTCVSW1  BACKUP
         Adapter ID: 29640007E386.0370
  IVL Port:       (IVL Port Section only on an IVL VSwitch)
     Adapter Owner: SYSTEM   NIC: FFFD.P00 Name: UNASSIGNED  Type: QDIO
       Porttype: Access
       RX Packets: 1040       Discarded: 27         Errors: 0
       TX Packets: 538        Discarded: 2          Errors: 0
       RX Bytes: 118222               TX Bytes: 61900
       Device: FFFD  Unit: 000   Role: DATA       Port: 2100
       VLAN: 0008
       Options: Ethernet Broadcast
         Unicast MAC Addresses:
           02-11-11-00-00-00  (Unicast MAC Address assigned to this system)
         Multicast MAC Addresses:
           03-FF-FF-FF-FF-02  (Multicast MAC Address for IVL Domain B)
 Ready; T=0.01/0.01 10:02:07
```

Figure 57. Query VSwitch SUE display

Through use of the QUERY VMLAN as shown in Figure 58 on page 141 we can see that both the CASEY and JONES systems are active members of IVL Domain B.

```
QUERY VMLAN

  VMLAN maintenance level:
    Latest Service: VM65918
  VMLAN MAC address assignment:
    System MAC Protection: OFF
    MACADDR Prefix: 021111 USER Prefix: 021111
    MACIDRANGE SYSTEM: 000001-FFFFFF
               USER:   000000-000000
    VMLAN default accounting status:
    SYSTEM Accounting: OFF       USER Accounting: OFF
  VMLAN general activity:
    PERSISTENT Limit: INFINITE   Current: 2
    TRANSIENT  Limit: INFINITE   Current: 0
  Trace Pages: 8
  VMLAN Directory Network Authorization: ENABLED
                                    (IVL Domain Section)
  IVL Domain: B   MAC address: 03-FF-FF-FF-FF-02  VLAN: 8
  IVL Domain Heartbeat Timeout: 30
  IVL Domain Capability: 8000000000000000
     Member: CASEY    MAC address: 02-11-11-00-00-00
       State: Active
       Port Group Status: Synchronized
       Global VSWITCH Status: Synchronized
       Heartbeat Count: <local>
       Member Capability: C000000000000000 Maintenance Level: V642

     Member: JONES    MAC address: 02-22-22-00-00-00
       State: Active
       Port Group Status: Synchronized
       Global VSWITCH Status: Synchronized
       Heartbeat Count: 2
       Member Capability: 8000000000000000 Maintenance Level: V632
 Ready; T=0.01/0.01 10:33:02
```

Figure 58. Query VMLAN with IVL Domain B

Define a Global VSwitch

The definition of a Global VSwitch is required when GLOBAL objects (i.e. Shared Port Group) are to be deployed in your virtual network configuration. Figure 59 on page 142, presents a successfully configured global VSwitch PEGGY.

Note: The example uses an OSA-Express 1G multi-port adapter. The configuration for all OSA-Express and Network Express adapters is similar.

- **DEFINE VSWITCH PEGGY GLOBAL ETHERNET**
  - A Global VSwitch is a virtual switch which can span multiple z/VM Systems through the IVL network Domain B.
    - A Global VSwitch connects to the IVL VSwitch's special IVL Port for Global VSwitch to VSwitch Communications.
    - Permits the Global VSwitch to manage and share the same physical LAG, share LACP and load balancing information.
  - A Global VSwitch with the same 8 byte name needs to be defined in each z/VM System that is part of the Global VSwitch.
  - A **Global ID** (*System Name.VSwitch*) is generated by the z/VM System.
  - A Global VSwitch is created when its first Global VSwitch member is *defined* and is deleted only when the last Global VSwitch member is *detached*.
  - Multiple Global VSwitches can be defined per z/VM System.
  - Only a Global VSwitch can share a LAG configuration.
- Repeat the same setup for VSwitch PEGGY on z/VM System JONES.

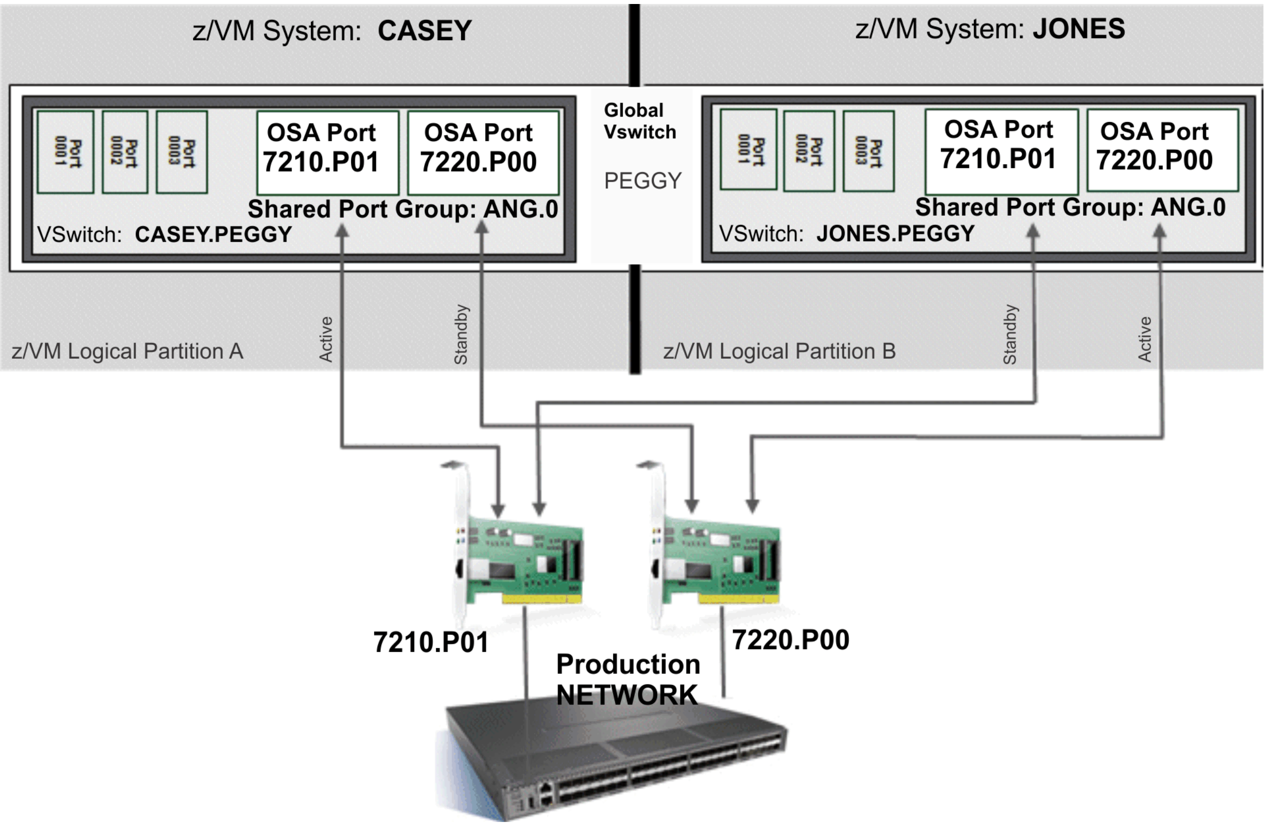


Figure 59. Global VSwitch PEGGY with shared port group ANG

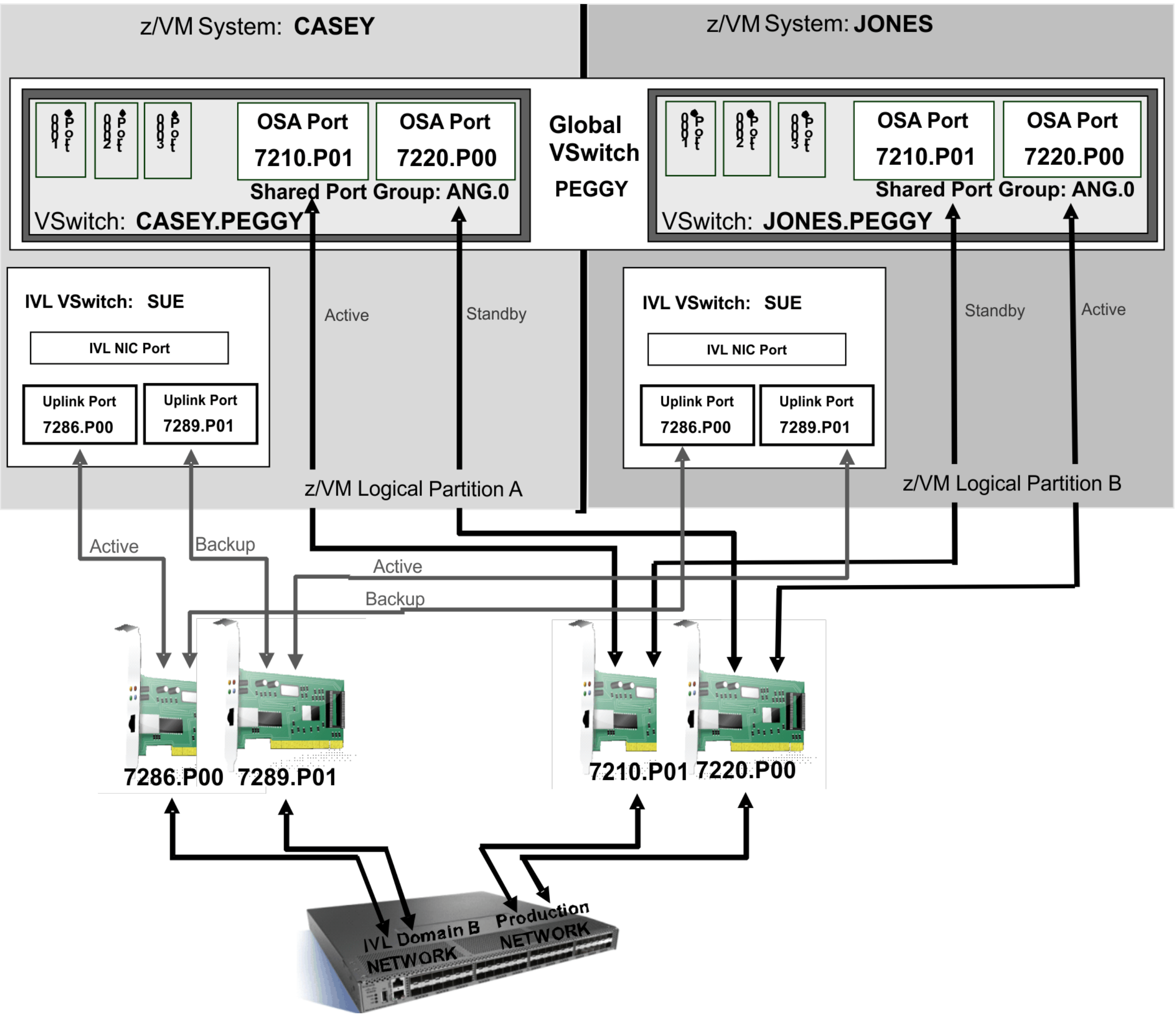


Figure 60. Example of a Multi-VSwitch LAG Configuration

Take-Over and Take-Back Operations

Besides providing communication among Global VSwitches, the IVL supports the transmission of production work load traffic. This use is required by the Multi-VSwitch LAG in support of a MAC Address Take-Over recovery operation. In brief, a take-over operation is a condition where one VSwitch takes over the transferring and receiving of production data with the partner physical switch on behalf of another VSwitch configured to share the same OSA adapter. The operational role of the IVL in a take-over operation is presented in Figure 61 on page 144, and referenced further in the following description.

MAC Address Take-Over is initiated by the OSA adapters whenever a network connection sharing a specific LAG port becomes inoperable. In the case of a network failure, one of the remaining active network connections will take over production data transfer operations for the failing VSwitch member through the IVL data plane. The following use case presents a take-over operations focusing on the IVL Manager in concert with the OSA adapter.

Once the global VSwitch network connection to the OSA adapters is restored, a MAC Address Take-Back operation restores all guest simulated NIC MAC addresses to the owning global VSwitch and production traffic resumes on the OSA adapters in the LAG.

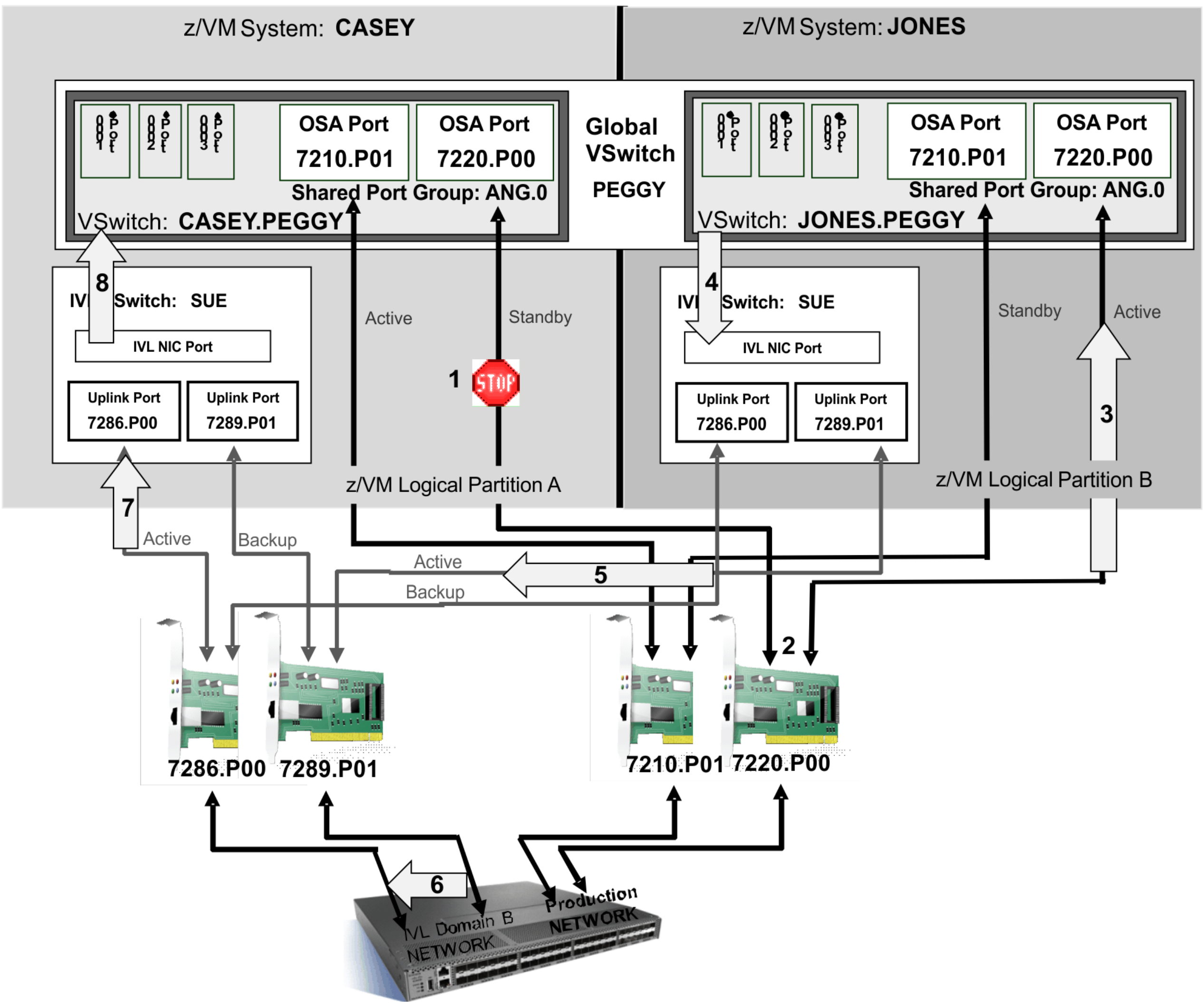


Figure 61. LAG Port Controller Take-Over Sequence

Note: The example uses an OSA-Express 1G multi-port adapter. The configuration for all OSA-Express and Network Express adapters is similar.

Take-Over Sequence

The following take over sequence references the configuration depicted in Figure 61 on page 144.

Sequence	Description
1	An unexpected connectivity outage developed between global VSwitch CASEY.PEGGY's *standby* LAG Port Controller (VSwitch Uplink Port) and its OSA-Express adapter. Being there is still a functional sharing LAG Port Controller connection to this OSA-Express adapter, the OSA-Express adapter will not drop the light to the physical partner switch. In this case, the physical partner switch will continue to send inbound data destined for VSwitch CASEY.PEGGY's non-operational LAG Port Controller through the OSA feature identified by 7220.P00. This will result in those frames being dropped by the OSA-Express adapter if another network connection doesn't take-over.
2	To mitigate this loss of connectivity the OSA-Express adapter selects another sharing VSwitch LAG Port Controller to “Take-Over” for VSwitch CASEY.PEGGY's non-operational LAG Port Controller.

Sequence	Description
3	The OSA-Express adapter selects the sharing LAG Port Controller on VSwitch JONES.PEGGY and sends the MAC addresses of the simulated NICs on VSwitch CASEY.PEGGY to JONES.PEGGY: • VSwitch JONES.PEGGY updates its LAN hash table with the new take-over MACs which represent global VSwitch CASEY.PEGGY's simulated NICs • Ethernet frames targeted for global VSwitch CASEY.PEGGY are now sent by the OSA-Express adapter to VSwitch JONES.PEGGY to forward
4	VSwitch JONES.PEGGY resolves a destination MAC address of a frame received from (inbound) the OSA-Express adapter, determines that the destination MAC is not local, but is a “take-over” MAC address. The production payload is encapsulated into an IVL protocol Ethernet frame and is sent on the IVL to VSwitch CASEY.PEGGY.
5	IVL VSwitch SUE will send the encapsulated production payload on the IVL network via its active Uplink port.
6	The Ethernet payload may be directly received and delivered by the active OSA-Express adapter of z/VM System CASEY's IVL VSwitch or may go through a single hop on the wire to reach another active OSA-Express adapter. The payload will be sent to the active uplink port of IVL VSwitch SUE on system CASEY via the external switch.
7	Once received, IVL VSwitch SUE on system CASEY, determines which global VSwitch to send the Ethernet payload. In this case the target is global VSwitch CASEY.PEGGY.
8	The IVL VSwitch removes the IVL protocol encapsulation and places the production payload into the input queue into VSwitch CASEY.PEGGY native Uplink Port as if it was received from the OSA-Express adapter. The production payload is now ready to be delivered to its target guest simulated NIC.

Chapter 10. Troubleshooting a Virtual Switch or Guest LAN

This chapter focuses on troubleshooting virtual switches and guest LAN connectivity issues in a z/VM environment. It explains two primary methods for capturing and analyzing network traffic:

1. Use promiscuous mode on guest virtual machines with tools like tcpdump.
2. Use the TRSOURCE TYPE LAN command for detailed tracing of virtual switch or guest LAN traffic.

The information details how to authorize and activate promiscuous mode securely, and emphasizes VLAN boundary enforcement and administrative controls to maintain network security. It also covers interpreting tcpdump output and processing TRSOURCE trace data with tools like TRACERED and IPFORMAT to diagnose problems such as VLAN mismatches or dropped packets. Overall, it provides practical guidance for system administrators to effectively monitor, capture, and analyze virtual network traffic within z/VM systems while maintaining strict security controls.

While virtual switches and guest LANs provide an efficient, secure method for transferring data among guest virtual machines, these memory to memory transfers do present challenges with respect to troubleshooting connectivity problems. Unlike a traditional hardware network, the deployment of a hardware LAN Sniffer can only reveal the network traffic that is entering or leaving the guest LAN segment not the actual traffic traversing the guest LAN segment. The details collected from the hardware sniffer could be used to help determine if the network problems are on the physical or virtual LAN segment. Once it has been determined that the problem resides in the virtual network, further analysis of the actual guest LAN traffic is required. This chapter outlines two methods that support the capture of virtual switch and guest LAN traffic.

See the following topics for more information on troubleshooting guest LANs and virtual switches:

Task	Source
Control Program (CP) command information	*z/VM: CP Commands and Utilities Reference*
CP messages information	*z/VM: CP Messages and Codes*
CP command: TRSOURCE	TRSOURCE in *z/VM: CP Commands and Utilities Reference*
IPFORMAT Packet Trace Formatting Tool	Using IPFORMAT Packet Trace Formatting Tool in *z/VM: TCP/IP Diagnosis Guide*
IPFORMAT Messages	*z/VM: TCP/IP Messages and Codes*

Methods of Guest LAN Sniffing

Promiscuous Mode with a Guest Virtual Machine

One method to troubleshoot a potential networking problem is to make use of the tools already available on the guest virtual machine. For example, a virtual machine running Linux may already have networking diagnostic tools available such as **tcpdump** or **ethereal**. In cases where the networking problem appears to be reduced to a limited number of guests on the virtual switch or guest LAN, the Guest promiscuous mode option can be employed. Once the Guest Virtual Machine is in promiscuous mode, **tcpdump** or **ethereal** can collect and analyze the LAN traffic sent to the Guest. Promiscuous mode is a mode of operation where the network adapter intercepts all data flowing over the network regardless of destination MAC or IP address. Promiscuous mode in a guest virtual machine environment provides the guest's virtual NIC with a copy of all data flowing on its configured LAN or VLANs.

Note: At present, there is no application programming interface (API) to deliver promiscuous data to a sniffer application for a TCP/IP stack running on z/VM.

TRSOURCE TYPE LAN Option

When more control is needed to troubleshoot a connectivity problem or when dealing with a HiperSockets LAN, the best option is to use a TRSOURCE trace with type LAN. This is a privilege class C command that allows the entire LAN segment or VSWITCH to be traced and the packets recorded. This option allows filtering of the packets based on certain criteria such as VLAN ID, USERID/NIC pairings, dropped packets, and external traffic. The debug information can then be processed like a normal TRSOURCE trace and run through TRACERED. The IPFORMAT tool can be used to readily format packets and perform data interpretation, as well as to export captured data to PCAP format. More information on IPFORMAT can be found in the *z/VM: TCP/IP Diagnosis Guide*.

Keeping the Virtual Network Secure

One of the main advantages of deploying a guest LAN or virtual switch is the added security that z/VM has to offer. For example, a guest LAN is completely isolated from the hardware network unless a router is specifically configured. A virtual switch has restricted access which permits only authorized users to couple and send data. VLAN membership controls are present in the VM environment. External Security Managers (ESMs) such as RACF/VM also help in securing guest LANs and virtual switches. The administrator controls that are in place ensure guests who couple to the virtual network receive only traffic intended for them. These safe guards ensure that the system administrator controls who is sniffing the network.

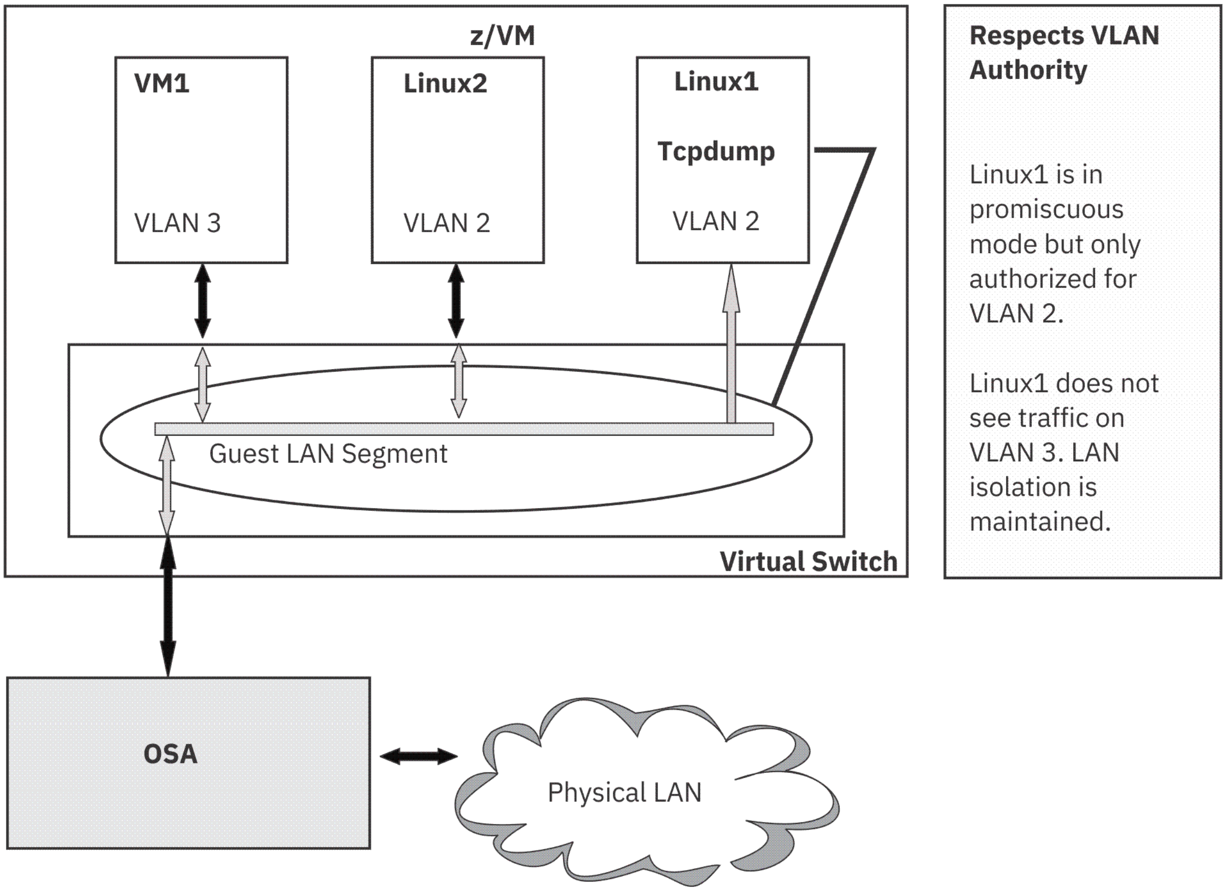

z/VM
VM1
VLAN 3
Linux2
VLAN 2
Linux1
Tcpdump
VLAN 2
Guest LAN Segment
Virtual Switch
Respects VLAN Authority
Linux1 is in promiscuous mode but only authorized for VLAN 2.
Linux1 does not see traffic on VLAN 3. LAN isolation is maintained.
OSA
Physical LAN

Figure 62. VLAN Authority Honored

The controls that manage who can couple to a guest LAN or virtual switch control which guests can use the guest LAN sniffing function. These controls include the SET commands and MODIFY statements for guest LAN and VSWITCH as well as ESM controls and user class privileges. guest LAN sniffing is also under the existing VLAN authorization. It will not violate VLAN boundaries set for a specific guest. Figure 62 on page 148 shows the VLAN isolation that is still present even when promiscuous mode is active.

Security mechanisms for guest LAN sniffing:

- Authorization for guest promiscuous mode is enforced through the use of the SET command, the MODIFY statement, or an optional ESM.
- TRSOURCE with the TYPE LAN option requires a privilege class of C in order to issue the commands.

These safe guards ensure that the system administrator knows who is ‘sniffing’ the network. This information can be queried with Query LAN, Query VSWITCH, or Query TRSOURCE. Monitor records record the number of users running in promiscuous mode and the number of active TRSOURCE TYPE LAN traces on the virtual switch.

Using Promiscuous Mode with Guest Virtual Machine

Deploying promiscuous mode for a guest virtual machine provides the ability to make use of the network diagnostic tools that are already available such as **tcpdump** or **ethereal**. This method is real-time debugging where the application receives the data immediately as it appears on the LAN. When a virtual NIC is placed into promiscuous mode, all traffic on the guest LAN segment or VLAN segment will be picked up by the virtual NIC. This operating mode is analogous to a hardware LAN sniffer. The virtual NIC may be placed into this mode programmatically by the guest device driver (if support is available) or manually through the CP SET NIC command. See Figure 63 on page 149.

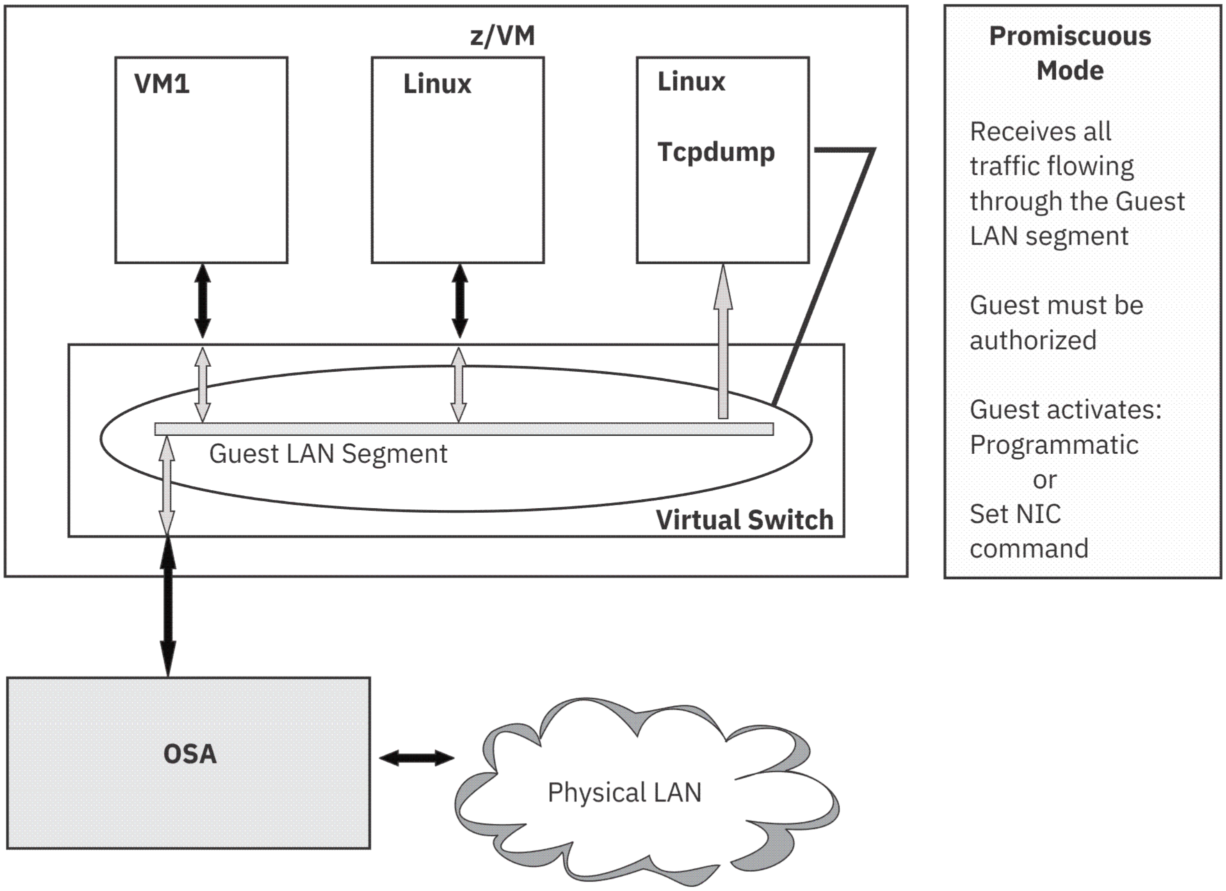


Figure 63. Guest Promiscuous Mode

Authorizing a Guest for Promiscuous Mode

By default, guests NICs are NOT authorized for promiscuous mode. To temporarily authorize a guest for promiscuous mode on a restricted network, use the SET VSWITCH command to GRANT (or for a specific NIC via a set PORTNUMBER) and PROMISCUOUS. To make a more persistent change:

- Add the PROMISCUOUS option to the MODIFY statement (SYSTEM CONFIG), or
- Add the PROMISCUOUS option to the NICDEF statement (USER DIRECT).
- This authorization may also be established by an ESM.

Using the CP SET or MODIFY Commands for Promiscuous Mode Authorization

The syntax of the SET LAN command is:

```
SET LAN lanname GRANT userid {options}
```

The following options can be specified:

NOPROmiscuous
Denies authority for promiscuous mode for the given user ID.

PROmiscuous
Grants authority for promiscuous mode for the given user ID.

The procedure is similar for SET VSWITCH GRANT, SET VSWITCH PORTNUMBER, MODIFY LAN, MODIFY VSWITCH GRANT and MODIFY VSWITCH PORTNUMBER. See *z/VM: CP Commands and Utilities Reference* or *z/VM: CP Planning and Administration* for complete syntax. When a persistent configuration is necessary, the PROMISCUOUS option can be added to a specific NICDEF user directory statement. These mechanisms allow VLAN authorization as well. When both VLANs and promiscuous mode are authorized, only data on the authorized VLANs is delivered to the guest. Promiscuous Mode does NOT cross VLAN boundaries. See Figure 64 on page 150 for an example. When the virtual switch is in isolation mode, the promiscuous guest will still receive copies of the internal guest to guest traffic. Isolation mode does not affect promiscuous mode behavior.

```
To grant Promiscuous Mode authority:
1. DEF LAN sample RESTricted TYPE QDIO
2. SET LAN sample GRANT linux1 PRO

To revoke Promiscuous Mode authority once it has been granted:
1. SET LAN sample REVOKE linux1
2. SET LAN sample GRANT linux1
   (or)
   SET LAN sample GRANT linux1 NOPRO
```

Figure 64. Example of Granting and Revoking Promiscuous Mode Authority on a Guest LAN

To view the guest promiscuous authorization, issue Q VSWITCH or QUERY LAN with the promiscuous option. See QUERY VSWITCH or QUERY LAN in *z/VM: CP Commands and Utilities Reference* for complete syntax. In Figure 65 on page 150, LINUX1 and LINUX3 are authorized for promiscuous mode on VSWITCH WADP.

```
Q VSWITCH PRO

VSWITCH SYSTEM WADP     Type: VSWITCH Connected: 3    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    PRIROUTER                 Accounting: OFF
  VLAN Aware  Default VLAN: 0001    Default Porttype: Trunk
  State: Ready
  IPTimeout: 5         QueueStorage: 8
  RDEV: 1000.P00 VDEV: F800 Controller: DTCVSW1  ACTIVE
    Authorized promiscuous userids:
      LINUX1    LINUX3
```

Figure 65. Query VSWITCH PRO output

Promiscuous mode is still bound by VLAN authority. Therefore, if LINUX1 was authorized for promiscuous mode and VLAN 2, LINUX1 would only receive VLAN 2 traffic. To see the CP VLAN authorized list, issue Q VSWITCH ACC. Use ESM interfaces to display the list if the virtual switch is protected by an

External Security Manager. In a guest LAN environment, where VLANs are done on a NIC level, a guest in promiscuous mode would receive traffic on any VLAN id (see Figure 66 on page 151).

```
Q LAN PRO

LAN SYSTEM GLOBLAN      Type: QDIO    Connected: 0    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    IP                        Accounting: OFF
    Authorized promiscuous userids:
      TEST2
```

Figure 66. Query LAN PRO output

Once a guest has been authorized, the guest's network interface card needs to be put into promiscuous mode. This can be done in two ways:

- CP SET NIC command
- IOCTL sent from the guest device driver (programmatically)

Activating Promiscuous Mode

Activating promiscuous mode for an authorized virtual NIC may be accomplished programmatically through the virtual machine's OS device driver or manually through the use of the CP SET NIC command. An example of a programmatic activation would be the deployment of the **tcpdump** program for Linux on IBM Z. With the required Linux QETH driver support, the **tcpdump** program will initiate the device driver to activate promiscuous mode on the target virtual NIC that LAN sniffing is to commence. The absence of device driver support necessitates the use of the manual method.

CP SET NIC Command

The SET NIC command activates promiscuous mode on the specified authorized NIC. The command is designed to be used when the OS device driver does not support z/VM promiscuous mode. The data device of the network adaptor which will be using promiscuous mode must be known. As well the NIC owner issuing the command must be authorized for promiscuous mode on the LAN (assuming the NIC is already coupled).

The complete syntax for the SET NIC command can be found in the *z/VM: CP Commands and Utilities Reference*.

```
CP SET NIC vdev {options}
```

vdev

the virtual data device of the NIC. A Query NIC DETails will reveal which device is the data device.

```
CP Q NIC DET
Adapter 3D00  Type: QDIO      Name: OSA03FRE    Devices: 3
  Port 0 MAC: 02-00-FF-00-00-03  VSWITCH: SYSTEM WADP
      RX Packets: 0          Discarded: 0          Errors: 0
      TX Packets: 10         Discarded: 0          Errors: 0
      RX Bytes: 0                    TX Bytes: 704
  Connection Name: z/VM0000  State: Session Established
      Device: 3D00  Unit: 000   Role: CTL-READ
      Device: 3D01  Unit: 001   Role: CTL-WRITE
      Device: 3D02  Unit: 002   Role: DATA
      VLAN: 0001
      Options: Broadcast Multicast IPv6 IPv4 VLAN
        Unicast IP Addresses:
          10.6.3.3             MAC: 02-00-FF-00-00-03
          FE80::200:FF00:100:3 MAC: 02-00-FF-00-00-03 Local
        Multicast IP Addresses:
          224.0.0.1            MAC: 01-00-5E-00-00-01
          FF02::1              MAC: 33-33-00-00-00-01 Local
          FF02::1:FF00:3       MAC: 33-33-FF-00-00-03 Local
```

Figure 67. Query NIC DETails

In Figure 67 on page 152, the data device is 3D02.

OPTIONS

PROmiscuous
Specifies that promiscuous mode should be activated on the NIC.

NOPROmiscuous
Specifies that promiscuous mode should be deactivated on the NIC.

To put LINUX1’s 3D02 into promiscuous mode, see the example outlined in Figure 68 on page 152.

```
From Linux1 issue:
    CP SET NIC 3D02 PRO
```

Figure 68. Activating Promiscuous Mode

To turn promiscuous mode off for LINUX1 3D02, see Figure 69 on page 152.

```
CP SET NIC 3D02 NOPRO
```

Figure 69. Deactivating Promiscuous Mode

Once promiscuous mode has been activated, the options line on a Query NIC DETails, will show either promiscuous or Promiscuous_Denied (see Figure 70 on page 153).

```
CP Q NIC DET
Adapter 3D00  Type: QDIO      Name: OSA03FRE    Devices: 3
  Port 0 MAC: 02-00-FF-00-00-03  VSWITCH: SYSTEM WADP
      RX Packets: 0          Discarded: 0          Errors: 0
      TX Packets: 10         Discarded: 0          Errors: 0
      RX Bytes: 0                    TX Bytes: 704
  Connection Name: z/VM0000  State: Session Established
      Device: 3D00  Unit: 000   Role: CTL-READ
      Device: 3D01  Unit: 001   Role: CTL-WRITE
      Device: 3D02  Unit: 002   Role: DATA
      VLAN: 0001
      Options: Broadcast Multicast IPv6 IPv4 VLAN Promiscuous
        Unicast IP Addresses:
          10.6.3.3             MAC: 02-00-FF-00-00-03
          FE80::200:FF00:100:3 MAC: 02-00-FF-00-00-03 Local
        Multicast IP Addresses:
          224.0.0.1            MAC: 01-00-5E-00-00-01
          FF02::1              MAC: 33-33-00-00-00-01 Local
          FF02::1:FF00:3       MAC: 33-33-FF-00-00-03 Local

          FF02::1              MAC: 33-33-00-00-00-01 Local
          FF02::1:FF00:3       MAC: 33-33-FF-00-00-03 Local
```

Figure 70. The Options Line from a Query NIC DETails

If the Promiscuous_Denied response is displayed, it means that NIC attempted to go into promiscuous mode but failed. Connectivity will continue as before but no sniffing will occur on the specified NIC. A failure could occur for any of the reasons listed in Table 6 on page 153.

Notes:

1. Promiscuous mode can only be set for a QDIO or EQDIO NIC and is not supported on HiperSockets. For assistance debugging a HiperSockets NIC, see “Using TRSOURCE to TRACE a Guest LAN or Virtual Switch” on page 154.
2. The example uses a QDIO device. The same commands can be used for EQDIO devices.

Table 6. Promiscuous_Denied Keyword Reasons and Solutions

Possible Reasons	**Possible Solutions**
The NIC owner NOT authorized for promiscuous mode on the LAN segment it is currently coupled.	1. Uncouple and couple to LAN, where the owner has promiscuous authority and is using an ESM. 2. The system administrator issues SET LAN lanname GRANT owner PRO for the owner. For virtual switches, promiscuous is specified on the SET VSWITCH GRANT or SET VSWITCH PORTNUMBER command.
The NIC data device is not the first data device on that NIC to activate promiscuous mode.	Only one promiscuous mode connection is allowed per NIC. Turn promiscuous mode off for the conflicting data device using the SET NIC command. Reissue the SET NIC command for the data device requiring promiscuous mode.

Interpreting the tcpdump output

Once the authorization and initialization of promiscuous mode has been completed, the Linux guest can now use **tcpdump** to collect data about the network traffic. Figure 71 on page 154 shows output from **tcpdump** running on a Linux guest in promiscuous mode.

```
linuxj:~ #
tcpdump -i eth0
tcpdump -i eth0
13:13:48.437836 9.60.28.55.router > RIP2-ROUTERS.MCAST.NET.router:  RIPv2
13:13:48.438364 9.60.28.23 > 9.60.28.55: icmp: RIP2-ROUTERS.MCAST.NET udp
13:13:54.947195 vmt1.endicott.ibm.com.router > RIP2-ROUTERS.MCAST.NET.rou
13:13:58.313192 :: > ff02::1:ff00:11: icmp6: neighbor sol: who has fe80::
13:13:59.313573 fe80::26f:5a00:100:11 > ipv6-allrouters: icmp6: router so
13:14:03.313200 fe80::26f:5a00:100:11 > ipv6-allrouters: icmp6: router so
13:14:05.179268 :: > ff02::1:ff00:14: icmp6: neighbor sol: who has fe80::
13:14:06.179453 fe80::26f:5a00:100:14 > ipv6-allrouters: icmp6: router so
13:14:07.314061 fe80::26f:5a00:100:11 > ipv6-allrouters: icmp6: router so
13:14:10.179770 fe80::26f:5a00:100:14 > ipv6-allrouters: icmp6: router so
13:14:14.180029 fe80::26f:5a00:100:14 > ipv6-allrouters: icmp6: router so
13:14:18.473315 9.60.28.55.router > RIP2-ROUTERS.MCAST.NET.router:  RIPv2
13:14:18.473950 9.60.28.23 > 9.60.28.55: icmp: RIP2-ROUTERS.MCAST.NET udp
13:14:20.628769 9.60.28.64.filenet-tms > btvdns01.srv.ibm.com.domain:  49
13:14:24.982405 vmt1.endicott.ibm.com.router > RIP2-ROUTERS.MCAST.NET.rou
13:14:25.629439 9.60.28.64.filenet-rpc > pokdns02.srv.ibm.com.domain:  49
13:14:25.673228 9.60.28.64.filenet-rpc > btvdns01.srv.ibm.com.domain:  49
13:14:30.638587 9.60.28.70 > 9.60.28.72: icmp: echo request (DF)"
13:14:30.689090 9.60.28.72 > 9.60.28.70: icmp: echo reply"
13:14:31.646919 9.60.28.70 > 9.60.28.72: icmp: echo request (DF)"
13:14:31.647087 9.60.28.72 > 9.60.28.70: icmp: echo reply"
13:14:32.646379 9.60.28.70 > 9.60.28.72: icmp: echo request (DF)"
13:14:32.646610 9.60.28.72 > 9.60.28.70: icmp: echo reply"
13:14:32.765191 9.60.28.64.filenet-rpc > btvdns01.srv.ibm.com.domain:  24
13:14:32.800933 9.60.28.64.filenet-rpc > btvdns01.srv.ibm.com.domain:  24
13:14:48.510304 9.60.28.55.router > RIP2-ROUTERS.MCAST.NET.router:  RIPv2
13:14:48.511094 9.60.28.23 > 9.60.28.55: icmp: RIP2-ROUTERS.MCAST.NET udp
13:14:55.018217 vmt1.endicott.ibm.com.router > RIP2-ROUTERS.MCAST.NET.rou
[1]+  Done                    tcpdump -i eth0 -w linuxj.tcpdump"

linuxj:~ #
```

Figure 71. Using ***tcpdump*** *with Promiscuous Mode on the Linux Guest*

In this example, LINUXJ is configured with IP address 9.60.28.71. Pings are captured in the echo request and echo reply for IP addresses 9.60.28.70 and 9.60.28.72. This allows a guest to verify that packets are being received by another guest. If there was a problem in the network, the **tcpdump** output may show something similar to Figure 72 on page 154.

```
13:14:30.638587 9.60.28.70 > 9.60.28.72: icmp: echo request (DF)"
13:14:31.646919 9.60.28.70 > 9.60.28.72: icmp: echo request (DF)"
13:14:32.646379 9.60.28.70 > 9.60.28.72: icmp: echo request (DF)"
```

Figure 72. Possible Network Problem Revealed by Promiscuous Mode and ***tcpdump***

In Figure 72 on page 154, echo requests are entering the network but no replies are being captured.

Using TRSOURCE to TRACE a Guest LAN or Virtual Switch

This method of capturing LAN traffic data employs the native CP tracing facilities. Data is captured and stored in a file for formatting and viewing by the IPFORMAT tool. This is not real-time debugging but can prove to be an effective method of troubleshooting LAN problems. Deploying TRSOURCE provides granularity in what is being traced. This method provides the capability to narrow the tracing to a particular virtual NIC, VLAN, or switch trunk port. In addition, all VLAN traffic is traceable with this method. See Figure 73 on page 155.

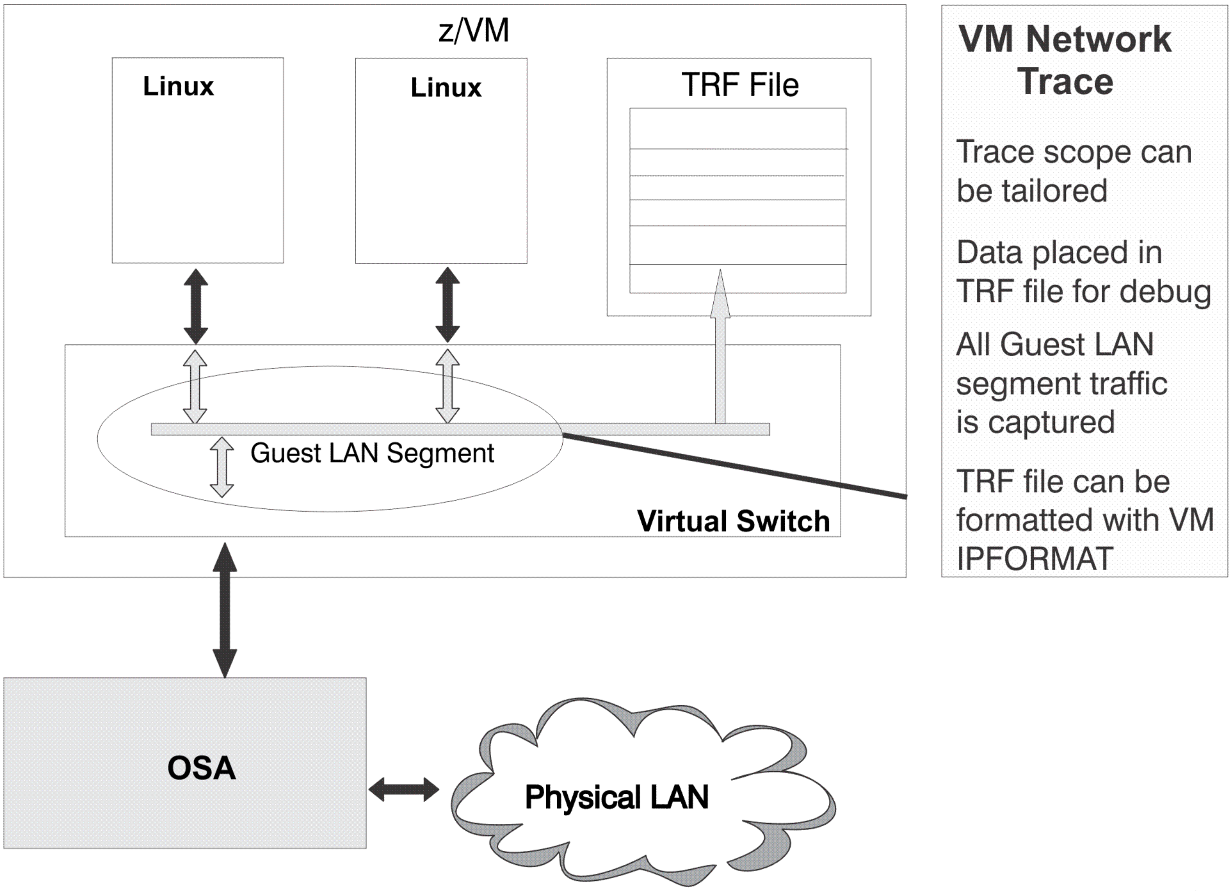

z/VM
Linux
Linux
TRF File
Guest LAN Segment
Virtual Switch
VM Network Trace
Trace scope can be tailored
Data placed in TRF file for debug
All Guest LAN segment traffic is captured
TRF file can be formatted with VM IPFORMAT
OSA
Physical LAN

Figure 73. Sniffer Mode using TRSOURCE

CP TRSOURCE TYPE LAN Command

The syntax for the TRSOURCE TYPE LAN command is:

```
CP TRSOURCE ID traceid SET traceset TYPE LAN {options}
```

where:

ID *traceid*
the identifier for the trace. Trace ID is a user-defined, alphanumeric, 1- to 8-character string.

SET *traceset*
is a user-defined, alphanumeric, 1- to 8-character string; it identifies the trace set that includes the specified trace ID.

TYPE LAN
specifies this is a trace for a virtual switch or guest LAN.

OPTIONS

OWNER *ownerid*
the owner of the guest LAN or virtual switch. If the *ownerid* is SYSTEM, then it is owned by the system and not an virtual machine.

LANNAME *lanname*
the name of the guest LAN or virtual switch to be traced.

VLAN ALL
all VLAN ids will be traced. Since this trace is initialized from a class C user, no VLAN authorization is needed in order to see all VLAN traffic.

VLAN *vlan*
up to 4 specific VLAN ids can be specified. Only the packets for the VLAN ids specified will be captured.

LENGTH 512
512 is the default. When no length option is specified, 512 bytes of data will be captured and recorded.

LENGTH FULL
2048 bytes of data will be captured and recorded. This is the maximum amount of data that can be traced per packet.

LENGTH *#ofbytes*
a number between 64 and 2048 can be specified. It represents the amount of data to be captured and recorded.

NIC *userid vdev*
userid is the user ID of the guest virtual machine to be traced. *vdev* is the virtual device belonging to the guest virtual machine being traced. Only traffic flowing inbound or outbound on the specified NIC (user ID *vdev* combination) will be captured and recorded.

TRUNK
only packets which have passed in or out of the OSA RDEV related to the VSWITCH will be recorded. If TRUNK is specified on a guest LAN, no data will be captured.

DROPPED
only packets that have been dropped or discarded will be captured. When this option is not specified, all packets regardless of successful delivery are recorded.

Note: Only one of the following options can be specified per TRSOURCE id:

- NIC *userid vdev*
- TRUNK
- DROPPED

If it is necessary to capture data on the same guest LAN or virtual switch using combinations of these options, define multiple TRSOURCE trace ids for the same OWNER LANNAME combination.

For a complete syntax diagram, see *z/VM: CP Commands and Utilities Reference*.

Figure 74 on page 157 is an example using the TRSOURCE TYPE LAN feature to capture only dropped packets on VLAN 3 or 4 for the virtual switch owned by SYSTEM named testvsw.

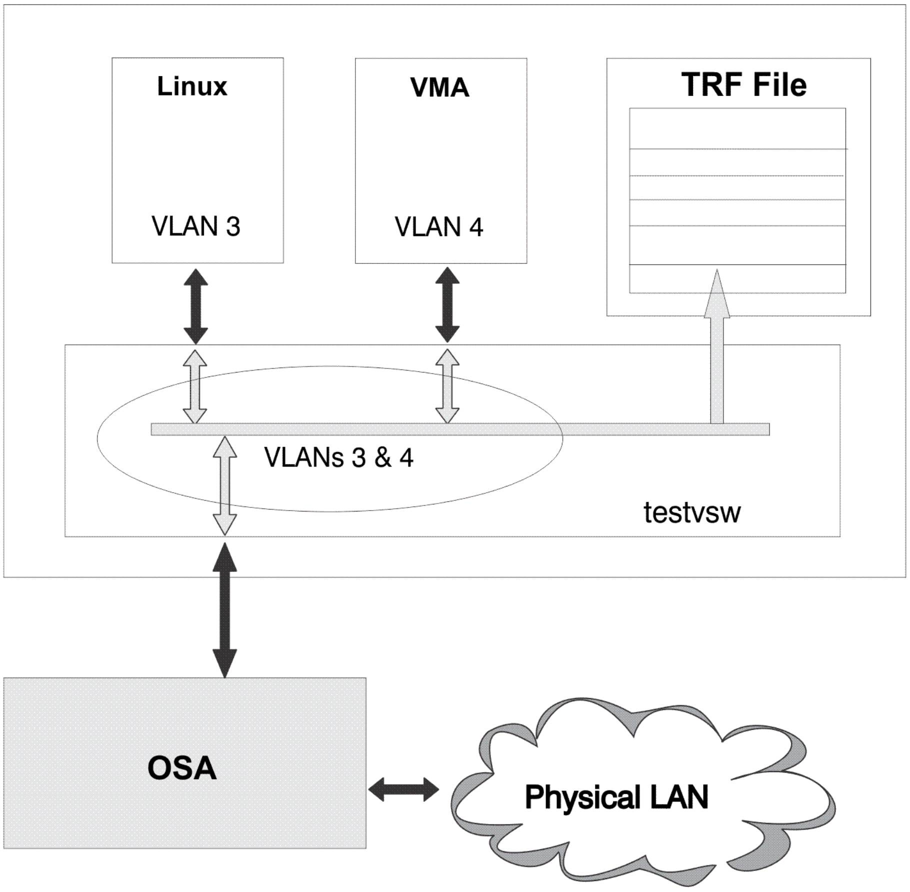


Figure 74. Example of z/VM Virtual Network

The following scenario is based on Figure 74 on page 157 and presents an example on how to deploy CP's LAN tracing function.

```
CP TRSOURCE ID testid TYPE LAN OWNER SYSTEM LANNAME testvsw VLAN 3 4 DROPPED
```

Figure 75. Example of TRSOURCE with TYPE LAN option

In this example we have two guests coupled to VSWITCH testvsw. Each guest has been configured in separate virtual LAN segments (VLANs). Based on the specified TRSOURCE command the following will be included in the trace:

- LAN segment defined by VSWITCH testvsw
- Traffic for both VLANs 3 and 4
- All packets dropped by this VSWITCH

Note: For this example, the **VLAN ALL** option may also have been used with the same results.

```
CP TRSOURCE ENABLE ID testid
```

Figure 76. Enabling a TRSOURCE trace

In order to begin tracing, the trace id must be enabled. Once enabled, the trace data will be captured and written to a TRF file.

```
CP TRSOURCE DISABLE ID testid
```

Figure 77. Disabling a TRSOURCE trace

When tracing is to be stopped, the trace id must be disabled. Once disabled, the captured data will be in a TRF file which can now be processed by TRACERED. The trace id must be disabled before TRACERED can process the TRF file.

Once the trace has been disabled, the data can be processed by TRACERED. The spool id is needed for input to TRACERED. The spool id is obtained through the Query TRF ALL. Figure 78 on page 158 shows the query and TRACERED command needed to process the TRSOURCE trace and write the results to a CMS file named TESTVSW TRCDATA.

```
Q TRF ALL

OWNERID  FILE TYPE CL RECS DATE  TIME            FILENAME FILETYPE ORIGINID
OPERATOR 0008 TRF  A  0002 06/08 12:37:09 5      testvsw  LAN      SYSTEM
Ready;

TRACERED 8 CMS TESTVSW TRCDATA (ALL
```

Figure 78. Processing a TRSOURCE trace using TRACERED

For more information on TRACERED, see *z/VM: CP Commands and Utilities Reference*.

The resulting file can be viewed in the original hex form (see Figure 79 on page 158) or formatted using the IPFORMAT tool. For detailed information on the IPFORMAT tool, see *z/VM: TCP/IP Diagnosis Guide*.

```
-------------------------- 06/08/05 12:37:14.130495 -------------------

.130495        0250BD21 BABEBE43 F76AFFFF 02000000 0134E2E8 SPID 0008
               E2E3C5D4 4040E5E2 E6E3D9C1 C3E8E3C3 D7C9D740
               40402D02 00010000 00FFE400 00000000 00000106
               00000000 00000114 00000001 00000000 00000000
               00000000 00000A06 03024500 01140002 00003C01
               63D90A06 03010A06 03020800 32200102 00000F32
               BD73238A 4810BBDB 7D501FB6 26CC8C83 FDB3182A
               E8564A84 F66F7585 1934D926 A1207713 7D7CB819
               30B0BC4D 944B83FD 52D3BE58 1D4E73DD B18D5384
               8F3BDBAD 75A92DCC 9F57ABD2 C042595A BFF0F098
               323CC102 9BE150BC 65C9ACC4 9C138FE2 6FE63A4D
               67F90F1D 65DEEB75 5E11496C D5C418C0 1B685739
               49FB76DE FA146DDB C9B0992D 8EBA5913 575C6C35
               28080991 7345BB29 563A036E 97FCAC6A 5F97ECB1
               B9DE3F05 CAF303CC CE95D432 D72CA51E 4496A3D6
               02AFCA71 85A24300 85F3847B 7948266D D4FA855F
               025F1A06 21D5972C 808FC75E D81F245A 4CC73537
               227FAE52 3FC646FC 642956E5 2B943647 BF5C57D2
               5E08205B 90E30000 00000000 00000000 00000000
               00000000 00000000 00000000 00000000 00000000
               00000000 00000000 00000000 00000000 00000000
               00000000 00000000 00000000 00000000 00000000
               00000000 00000000 00000000 00000000 00000000
               00000000 00000000 00000000 00000000 00000000
               00000000 00000000 00000000 00000000 00000000
               00000000 00000000 00000000 00000000 00000000
               00000000 00000000 00000000 00000000 00000000
               00000000 00000000 00000000 00000000 00000000
               00000000 00000000 0000
```

Figure 79. Raw Hex Trace File Processed by TRACERED

Interpreting the TRSOURCE Output

The file in Figure 79 on page 158 can be processed by IPFORMAT. However there is still useful information that can be read out of the file itself. The TRACERED file produced is similar in format to the file produced for a TYPE GT user. The output is broken down in Figure 80 on page 159. The TYPE LAN specific trace information starts at byte 10. In Figure 79 on page 158, the length of the trace file is X'0250' and the TOD value is X'BD21BABEBE43F76A'. The next halfword signifies that this is a TYPE LAN trace. For full decoding of the trace record, use IPFORMAT.

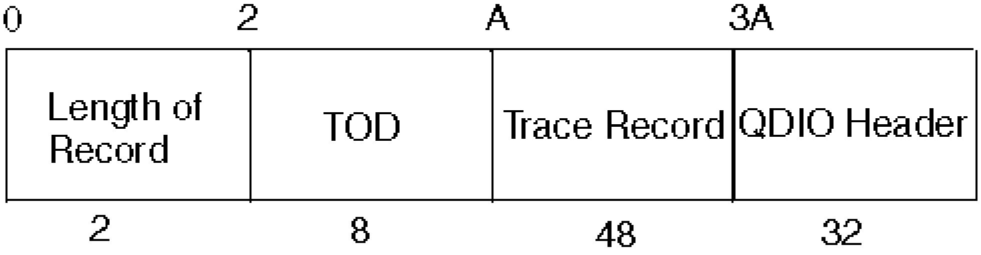


Figure 80. Layout of Raw Hex Data

Hex Displacement	Decimal Length	Description
0	2	Length of Trace Record
2	8	Time of Day Stamp
A	48	Trace Record
3A	32	QDIO Header
5A	Variable	Beginning of Payload

A Troubleshooting Example Using TRSOURCE

Assume there is a connectivity problem between two z/VM guests. Neither can ping to the other. A valid way for a network or system administrator to troubleshoot this problem would be to use a TRSOURCE TYPE LAN trace. The administrator (with Class C or higher privileges) could issue the commands in Figure 81 on page 159 to collect ping data from both guests.

```
TRSOURCE ID debug TYPE LAN OWNER SYSTEM LANNAME VSW1
TRSOURCE ENABLE ALL

(ping from both affected guests)

TRSOURCE DISABLE ALL
Q TRF ALL
TRACERED spoolid CMS debug file a (ALL
```

Figure 81. Collecting Data Using TRSOURCE

Now the TRSOURCE record is in a CMS file called DEBUG FILE A. Figure 82 on page 160 shows the file.

```
------------------------- 08/12/05 10:49:42.861852 -------------------------

       .861852           0250BD73 5BF14B81 C9A8FFFF 02000000 0134E2E8 SPID 0015
                         E2E3C5D4 4040E5E2 E6F14040 4040E3C3 D7C9D740
                         40402D02 00020004 00FFE400 00000000 00000106
                         00000000 00000114 00010002 00000000 00000000
                         00000000 00000A06 03034500 01140001 00003C01
                         63D90A06 03010A06 03030800 DEEF0103 0000B004
                         357829DD 115F38E6 D864555D 3D4E4C84 9E84DC0C
                         015B1140 D02061F2 55E48926 F30A257F 2A91126D
                         48228414 F0D9E6FB 4CD47579 127C9BE0 38960C66
                         7608A2D7 806CF9FD 0056CACB 5565084C 0E0CBC4E
                         241EA1CD FB1B7F21 173C48E7 9E40F3B1 92F51C6D
                         CA855896 B586591C CEBFC714 DEBBD920 F510B59C
                         266C8D8D 0564754A E2FD06E8 C2F6B567 21DBF09D
                         6ACBFEE6 32049489 383336E9 1E7E5470 E44D203B
                         28DC3085 1F82D8D5 52BE45E7 1906595C D8504A20
                         FC5CFA9F B5C814A7 71D2EA1B 1CEA115D 18FFBA79
                         FE86CF12 124AD10D 88F774FF 7964F757 A4A3454B
                         E5CACE84 75802288 E74752EB DE8B0635 8F6E5E92
                         FA518A3C E7220000 00000000 00000000 00000000
                         00000000 00000000 00000000 00000000 00000000
                         00000000 00000000 00000000 00000000 00000000
                         00000000 00000000 00000000 00000000 00000000
                         00000000 00000000 00000000 00000000 00000000
                         00000000 00000000 00000000 00000000 00000000
                         00000000 00000000 00000000 00000000 00000000
                         00000000 00000000 00000000 00000000 00000000
                         00000000 00000000 00000000 00000000 00000000
                         00000000 00000000 00000000 00000000 00000000
                         00000000 00000000 0000
```

Figure 82. TRSOURCE Data from DEBUG FILE A (Part 1 of 2)

```
------------------------- 08/12/05 10:52:04.672529 -------------------------

      .672529           0250BD73 5C788941 12AAFFFF 02000000 0134E2E8 SPID 0015
                        E2E3C5D4 4040E5E2 E6F14040 4040E3C3 D7C9D7E7
                        40403D02 00030004 00FFE400 00000000 00000106
                        00000000 00000114 00010003 00000000 00000000
                        00000000 00000A06 03014500 01140000 00003C01
                        63DA0A06 03030A06 03010800 DD740104 0000B035
                        F4129F16 A571EDA2 A7C9D563 61EEE0A3 CC44DDB7
                        CDE6A110 55997FFF 3B73864B D44B7D75 58DF9FC6
                        F4C8171D AE1E3A7A FDEB8649 24505E6F 52798316
                        7B425239 F4A6C0EE D1EC1576 FE906B06 20F2E653
                        29C1B0E0 BD25BB07 48AF36C9 DC01CC55 CB3C9F59
                        092E070C 3E390BF8 AE019B88 C9DD9356 91FB7031
                        B051D0EF 5547ACF9 D225548F F73E2EDF 2CDA2DCA
                        50E41371 55586781 986B3643 01778963 DFADA97B
                        80791567 93B44048 3E8C0B54 FF330564 444A643B
                        C80CC397 B33A99FA 71CA7D35 4E4F1CB9 D627F69D
                        EF66DB6F B28629BA 0ECBC25B 1B77CF84 3FB2388D
                        0338DF7E B3C09F5D B7300738 B482D6F1 627436CC
                        2CF2B9BA 89440000 00000000 00000000 00000000
                        00000000 00000000 00000000 00000000 00000000
                        00000000 00000000 00000000 00000000 00000000
                        00000000 00000000 00000000 00000000 00000000
                        00000000 00000000 00000000 00000000 00000000
                        00000000 00000000 00000000 00000000 00000000
                        00000000 00000000 00000000 00000000 00000000
                        00000000 00000000 00000000 00000000 00000000
                        00000000 00000000 00000000 00000000 00000000
                        00000000 00000000 00000000 00000000 00000000
                        00000000 00000000 0000
```

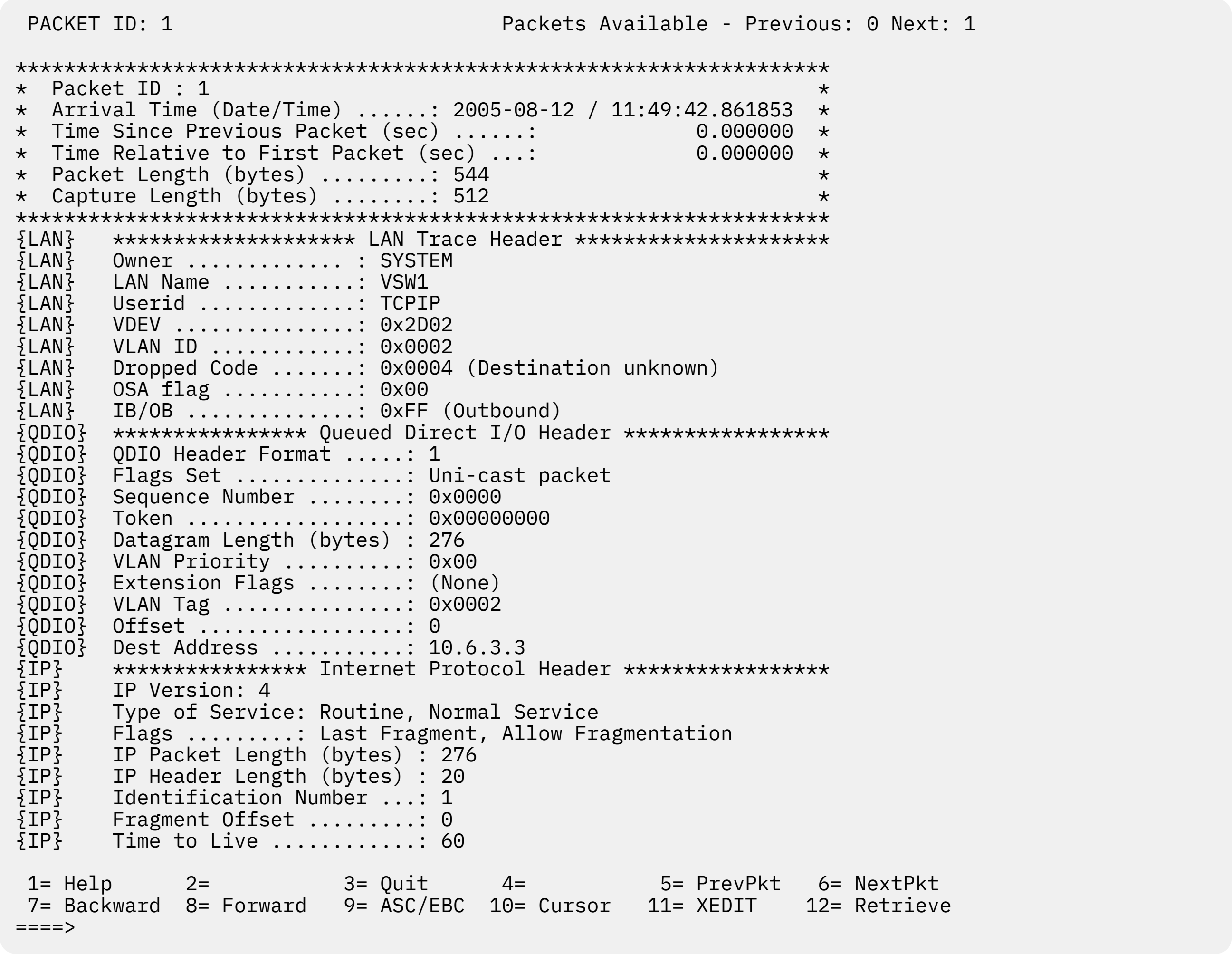
Figure 83. TRSOURCE Data from DEBUG FILE A (Part 2 of 2)

It can then be fed as input to the IPFORMAT tool as shown in Figure 84 on page 161 . Complete syntax and more information about IPFORMAT is available in the *z/VM: TCP/IP Diagnosis Guide*.

```
IPFORMAT debug file
```

Figure 84. Using IPFORMAT to decode the TRSOURCE Data

From the VLAN field in the IPFORMAT screen shots in figures Figure 85 on page 161 and Figure 86 on page 162, it can be concluded that there is a VLAN mismatch occurring. One guest is using VLAN 2 and the other is using VLAN3 . In order to solve this problem, both guests must be authorized for at least one common VLAN id and using it to communicate between each other. Depending on the guest, check to ensure that each has correct authorization. A change to authorization may require the use of an ESM (if one is controlling the LAN segment) or the use of the SET commands (GRANT, REVOKE and PORTNUMBER). The problem could also stem from a guest configuration error. An uncouple and couple back to the LAN segment is also necessary for authorization changes to take place when you are using an ESM. Ensure the guest is using the correct interface to ping from, the correct global VLAN id set, or both.

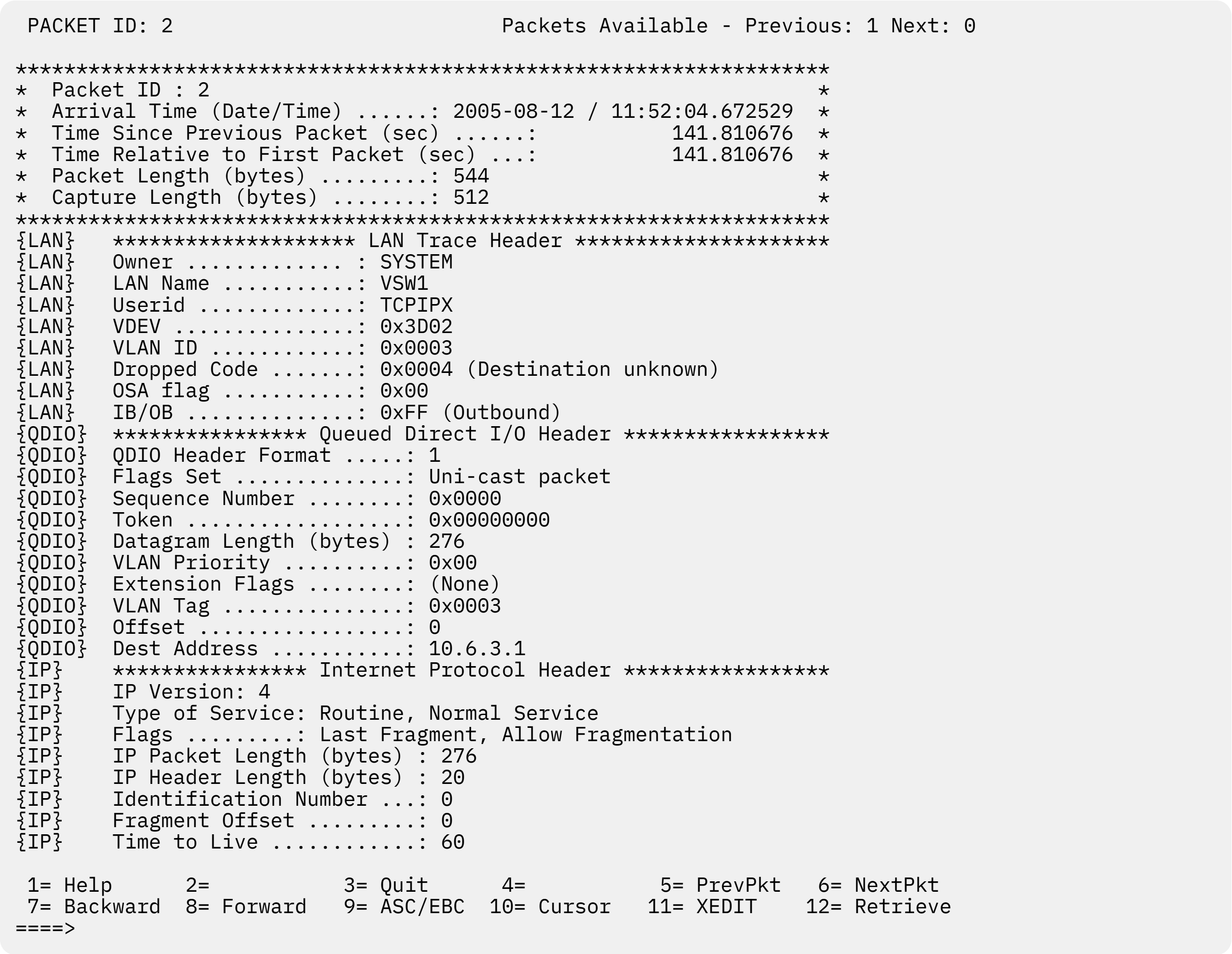


```
 PACKET ID: 1                           Packets Available - Previous: 0 Next: 1

*******************************************************************
*  Packet ID : 1                                                  *
*  Arrival Time (Date/Time) ......: 2005-08-12 / 11:49:42.861853  *
*  Time Since Previous Packet (sec) ......:            0.000000  *
*  Time Relative to First Packet (sec) ...:            0.000000  *
*  Packet Length (bytes) .........: 544                           *
*  Capture Length (bytes) ........: 512                           *
*******************************************************************
{LAN}   ******************** LAN Trace Header *********************
{LAN}   Owner ............. : SYSTEM
{LAN}   LAN Name ...........: VSW1
{LAN}   Userid .............: TCPIP
{LAN}   VDEV ...............: 0x2D02
{LAN}   VLAN ID ............: 0x0002
{LAN}   Dropped Code .......: 0x0004 (Destination unknown)
{LAN}   OSA flag ...........: 0x00
{LAN}   IB/OB ..............: 0xFF (Outbound)
{QDIO}  **************** Queued Direct I/O Header *****************
{QDIO}  QDIO Header Format .....: 1
{QDIO}  Flags Set ..............: Uni-cast packet
{QDIO}  Sequence Number ........: 0x0000
{QDIO}  Token ..................: 0x00000000
{QDIO}  Datagram Length (bytes) : 276
{QDIO}  VLAN Priority ..........: 0x00
{QDIO}  Extension Flags ........: (None)
{QDIO}  VLAN Tag ...............: 0x0002
{QDIO}  Offset .................: 0
{QDIO}  Dest Address ...........: 10.6.3.3
{IP}    **************** Internet Protocol Header *****************
{IP}    IP Version: 4
{IP}    Type of Service: Routine, Normal Service
{IP}    Flags .........: Last Fragment, Allow Fragmentation
{IP}    IP Packet Length (bytes) : 276
{IP}    IP Header Length (bytes) : 20
{IP}    Identification Number ...: 1
{IP}    Fragment Offset .........: 0
{IP}    Time to Live ............: 60

 1= Help      2=          3= Quit      4=          5= PrevPkt   6= NextPkt
 7= Backward  8= Forward   9= ASC/EBC  10= Cursor   11= XEDIT    12= Retrieve
====>
```

Figure 85. IPFORMAT Decoding of First Trace Packet



```
 PACKET ID: 2                           Packets Available - Previous: 1 Next: 0

*******************************************************************
*  Packet ID : 2                                                  *
*  Arrival Time (Date/Time) ......: 2005-08-12 / 11:52:04.672529  *
*  Time Since Previous Packet (sec) ......:           141.810676  *
*  Time Relative to First Packet (sec) ...:           141.810676  *
*  Packet Length (bytes) .........: 544                           *
*  Capture Length (bytes) ........: 512                           *
*******************************************************************
{LAN}   ******************** LAN Trace Header *********************
{LAN}   Owner ............. : SYSTEM
{LAN}   LAN Name ...........: VSW1
{LAN}   Userid .............: TCPIPX
{LAN}   VDEV ...............: 0x3D02
{LAN}   VLAN ID ............: 0x0003
{LAN}   Dropped Code .......: 0x0004 (Destination unknown)
{LAN}   OSA flag ...........: 0x00
{LAN}   IB/OB ..............: 0xFF (Outbound)
{QDIO}  **************** Queued Direct I/O Header *****************
{QDIO}  QDIO Header Format .....: 1
{QDIO}  Flags Set ..............: Uni-cast packet
{QDIO}  Sequence Number ........: 0x0000
{QDIO}  Token ..................: 0x00000000
{QDIO}  Datagram Length (bytes) : 276
{QDIO}  VLAN Priority ..........: 0x00
{QDIO}  Extension Flags ........: (None)
{QDIO}  VLAN Tag ...............: 0x0003
{QDIO}  Offset .................: 0
{QDIO}  Dest Address ...........: 10.6.3.1
{IP}    **************** Internet Protocol Header *****************
{IP}    IP Version: 4
{IP}    Type of Service: Routine, Normal Service
{IP}    Flags .........: Last Fragment, Allow Fragmentation
{IP}    IP Packet Length (bytes) : 276
{IP}    IP Header Length (bytes) : 20
{IP}    Identification Number ...: 0
{IP}    Fragment Offset .........: 0
{IP}    Time to Live ............: 60

 1= Help      2=           3= Quit      4=           5= PrevPkt   6= NextPkt
 7= Backward  8= Forward   9= ASC/EBC  10= Cursor   11= XEDIT    12= Retrieve
====>
```

Figure 86. IPFORMAT Decoding of Second Trace Packet

Part 3. APPC Server and Requester Virtual Machines

This part of the document describes how to set up server and requester virtual machines. Chapter 11, “Setting Up Server and Requester Virtual Machines,” on page 165 describes how to set up these virtual machines, how to identify resources, and how to identify the resource that is the target of the connection request.

This part describes the tasks that you must complete before programs can manage resources or request access to a resource. The information helps you complete the following tasks:

- Understand the difference between local, global, system, and private resources.
- Understand how user programs identify their communication partner, the target resource.
- Set up virtual machines to manage resources.
- Set up virtual machines to connect to resources.
- Set up CMS communications directories.

The remainder of this book does not contain TCP/IP related information. For information on TCP/IP, refer to Part 2, “Planning Virtual Networks,” on page 37.

Related Information: See the following sources for more information on the tasks in this part:

Task	Source
Writing resource manager and user programs	*z/VM: CP Programming Services* *z/VM: CMS Application Development Guide*
Establishing required virtual machine environments	Chapter 12, “Preparing to Use the TSAF Virtual Machine,” on page 187 Chapter 17, “Preparing to Use the AVS Virtual Machine,” on page 245 *z/VM: CMS Planning and Administration* *z/VM: CMS Commands and Utilities Reference*

Chapter 11. Setting Up Server and Requester Virtual Machines

This chapter focuses on configuring virtual machines (VMs) in IBM's z/VM environment to function either as servers that manage resources or as requesters that access the resources. It begins by explaining the relationship between user programs and resource manager programs, which act as communication partners. The chapter categorizes resources into four types—local, system, global, and private—each with distinct characteristics in terms of name space, accessibility, and management. Local and system resources are confined to a single system, while global and private resources can be accessed across systems in a TSAF or CS collection or through an SNA network.

Server VMs that manage local, system, or global resources must be explicitly authorized by using *IDENT and configured by using specific CP directory entries. In contrast, private resources are managed more flexibly by using a $SERVER$ NAMES file, which defines access permissions and resource manager programs. Requester VMs must identify the target resource by using LU names and transaction program names (TPNs), and can simplify this process by using symbolic destination names that are defined in CMS communications directories.

The chapter also details how to configure access security using three levels—SECURITY(NONE), SECURITY(SAME), and SECURITY(PGM)—and how to securely manage credentials by using APPCPASS directory statements. Finally, it outlines the setup of CMS communications directories at system, user, and IBM levels, which enable transparent and flexible communication across systems and networks.

Communications Partners

A user program and a resource manager program are communications partners. A user program requests access to a resource. The resource manager program, also called the resource manager, controls access to the resource. The user program and resource manager can be in the same system, in different systems in a TSAF or CS collection, or in different systems in an SNA network.

In Figure 87 on page 165, a user program, running in a requester virtual machine, has requested a connection to a resource. The resource manager, running in a server virtual machine, controls access to the resource. The resource does the task requested by the user program and the resource manager returns the results to the user program.

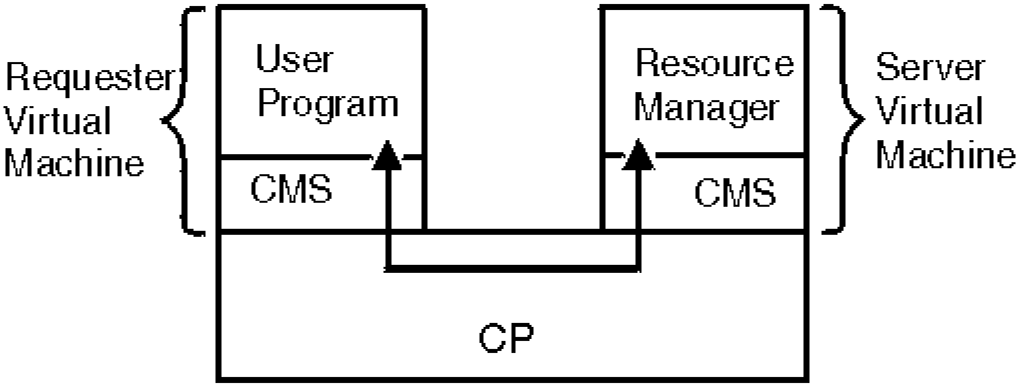


Figure 87. Communications between a User Program and a Resource Manager

The following sections describe how to set up a user's CMS virtual machine to be a server virtual machine or a requester virtual machine. The *z/VM: CMS Application Development Guide* describes how to write resource manager and user programs, and see *z/VM: CP Programming Services* for details on the *IDENT system service.

Resource Managers

The resource manager controls the availability of a local, global, system, or private resource. There are four ways to distinguish between the different types of resources:

- Name space
- Accessibility
- Environment
- Characteristics.

Resource Name Space

A resource's name space refers to the uniqueness of the resource name. A global resource name is identified to VM's Identify System Service (*IDENT). When the resource manager is authorized to manage a global resource, the resource's name must be unique in the TSAF or CS collection. Local and system resources are also identified to *IDENT. Here, when the resource manager is authorized to manage a local resource, the resource's name is unique in the local system. To identify a system resource, the virtual machine must be authorized to identify global resources. The difference between local and system resources is that a system resource is remotely accessible. Private resources are not identified to *IDENT. Their name is only unique within the server virtual machine in which they reside.

Resource names only need to be unique within their resource name space. For example, a global resource could have the same name as a local or private resource. Figure 88 on page 166 summarizes resource name spaces in a simple TSAF collection. CS collections use name spaces in a similar fashion.

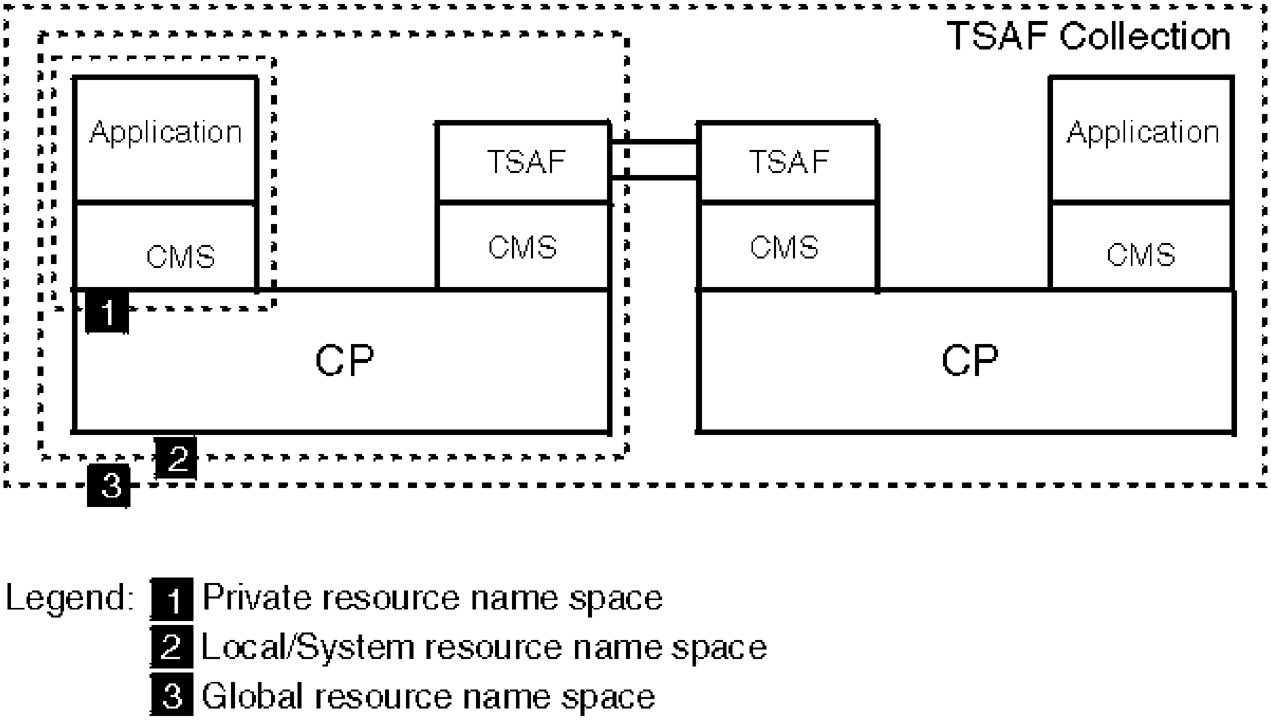

TSAF Collection
Application
CMS
TSAF
CMS
TSAF
CMS
Application
CMS
1
CP
CP
2
3
Legend:
1 Private resource name space
2 Local/System resource name space
3 Global resource name space

Figure 88. Resource Name Space in a TSAF Collection

Resource Accessibility

Just as the name space for resources differs, the accessibility of resources also differs. Local resources can only be accessed by user programs running in the same system. Global, system, and private resources can be accessed by user programs running in any system in the TSAF or CS collection or in the SNA network (by AVS and VTAM).

Virtual Machine Environment

Local and system resources are known on the system where they reside. Global resources are known throughout the TSAF or CS collection. Local, global, and system resources are identified, using *IDENT, by a server virtual machine (also called the resource manager). The server virtual machine must be logged on and the resource manager program must be running when a user program requests a connection to a local, global, or system resource. The resource manager program can run in a disconnected virtual machine. The Structured Query Language/Data System (SQL/DS) and the Shared File System (SFS) are examples of resource manager programs that run in disconnected server virtual machines. Connections to global resources are routed by the global resource name, which must be unique in the TSAF or CS collection. System resources must be unique on the system they reside, but can share a name with a system resource on another system.

Private resources are not explicitly known in TSAF or CS collections. The private resource server virtual machine does not identify itself to the TSAF or CS collection as a resource manager. It must be enabled to process private resource requests. This virtual machine owner can set up a special file, $SERVER$ NAMES, which lists the private resources known to that virtual machine. When a user program requests a connection to a private resource, the private server virtual machine does not need to be logged on and the private resource manager does not need to be running. If necessary, VM will automatically log on the server virtual machine and cause the resource manager program to start running. Connections to private resources are routed by user ID.

"Preparing Virtual Machines to Manage Resources" on page 173 describes how to set up server virtual machines to manage local, global, and private resources.

Resource Characteristics

The system administrator controls which users have access to local, global, and system resources. Because local and system resource names only need to be unique within a system, the system administrator can put a copy of the local or system resource using the same resource name on each system in the TSAF or CS collection. Users on each system can have local access to the resource. Data that does not change often and that is easily copied or a program that controls a device could be defined as a local or system resource. For example, a system administrator could have a printer at each system that is assigned the same resource name.

The system administrator can set up a resource as global when it is important that all users in the TSAF or CS collection have access to the same resource. Because the data they contain can change often, an SFS file pool or SQL/DS database may be defined as a global resource.

Server virtual machine owners control who has access to private resources without making changes to the resource manager. The virtual machine owner can make changes to the special NAMES file to authorize users. Therefore, a resource could be located in a central location, like a departmental virtual machine. Only those authorized department members could access the resource. Unauthorized department members could not access the resource even though they could access the departmental virtual machine.

As described earlier, more than one server virtual machine in the same system or in the TSAF or CS collection can manage private resources with the same name. Private resource names need only be unique within a virtual machine. For example, there is a program that lets users access files located in their virtual machines in a VM system from their workstation. This program is copied to each workstation user's virtual machine. A user program running in a workstation requests access to the private resource (the file access program) by specifying the private resource name and the user ID of the server virtual machine. The request is routed based on the user ID to the correct server virtual machine.

Programs that are not frequently used should be defined as private resources. For example, a private resource manager manages a plotter. The private server virtual machine is autologged and the plotter resource manager program is started when a user program requests to connect to the plotter. The server virtual machine would only be running when the plotter is being used. To improve performance when many connections are made to the same private resource within a short time, VM does not log off the autologged server virtual machine until it has been idle for 30 minutes.

The autologged server virtual machine is considered to be idle if:

- It has not been active since the last 30 minute check.
- It does not have a secondary user defined.
- It does not have any IUCV or APPC connections.

Different private resource managers can run serially in the same server virtual machine. The $SERVER$ NAMES file in the server virtual machine contains the list of users who are authorized to connect to each resource. This server virtual machine has a batch-like environment in that users are running programs in another virtual machine. It is not batch-like because the virtual machine owner can control who runs programs and which programs are run.

Because users of a private resource server virtual machine are generally authorized to access a few programs, the server virtual machine can be given special privileges. For example, a program (private resource) that issues privileged commands must run in a virtual machine that is authorized to issue privileged commands. System administrators can give the private server virtual machine this authorization because they can limit who accesses the program that issues privileged commands and they control which programs run in the server.

Summary of Resource Features

Table 7 on page 168 summarizes the features of local, global, system, and private resources. You should consider these features when designing resource managers programs.

Table 7. Summary of Resource Features

Feature	Resource Type			
	Local	**System**	**Global**	**Private**
Name space	Unique in the same system.	Unique in the same system.	Unique in the TSAF or CS collection.	Unique in the server virtual machine.
Accessible by user programs in the same system	Yes	Yes	Yes	Yes
Accessible by user programs in other systems in the TSAF or CS collection or in the SNA network	No	Yes	Yes	Yes
Security features	Resource manager must check, based on the access security user ID.	Resource manager must check, based on the access security user ID.	Resource manager must check, based on the access security user ID.	VM checks the access security user ID when invoking the resource manager.
Logon and invocation	Server virtual machine is explicitly logged on and resource manager is explicitly called; the resource is always ready to be used.	Server virtual machine is explicitly logged on and resource manager is explicitly called; the resource is always ready to be used.	Server virtual machine is explicitly logged on and resource manager is explicitly called; the resource is always ready to be used.	When needed, the server virtual machine is automatically logged on and the resource manager is called.
Resource manager characteristics (within a virtual machine)	One resource manager provides services.	One resource manager provides services.	One resource manager provides services.	Several resource managers can provide services serially.
Number of concurrent users	Typically many users.	Typically many users.	Typically many users.	Typically a few users.

User Programs

Users request services from resource manager programs. The resources to which the user program is asking to be connected can be located in the same system, in a different system in a TSAF collection, or in a system in the SNA network. "Preparing Virtual Machines to Request Connections to Resources" on page 177 describes how to set up the requester virtual machine in which a user program runs. See *z/VM: CMS Application Development Guide* for information on how to write a user program.

Identifying Communications Partners

A user program must identify its communications partner, the resource, and the location of the partner in its connection request. In the APPC and CPI Communications programming interfaces, the following parameters identify a communications partner:

- LU name
- Transaction program name (TPN).

LU Names

The LU name identifies the location of the transaction program. There are two types of LU names: the ***fully qualified name*** and the ***locally known LU name***. The fully qualified name is a network addressable unit (NAU), which is used for routing within the SNA network and is unique within the network. For more information on network addressable units, see the *Systems Network Architecture Transaction Programmer's Reference Manual for LU Type 6.2*.

The locally known LU name is unique to VM and identifies the location of the target resource. VM uses the locally known LU name for routing within the TSAF or CS collection. VTAM translates part of the locally known LU name to the fully qualified name so that the connection can be routed to the appropriate LU in the SNA network.

The locally known LU name, which has two parts separated by a blank, has the following format:

```
lu_name_qualifier target_lu_name
```

The first part of the locally known LU name, called the ***LU name qualifier***, identifies if the resource is located in the TSAF or CS collection or in the SNA network. The second part of the locally known LU name, called the ***target LU name***, is used by VTAM to create the fully qualified name.

The information you specify for each part of the locally known LU name varies, depending on the type and location of the resource specified on the connection request. Table 8 on page 170 summarizes how to specify this information.

Transaction Program Names

The second parameter used to identify a communications partner is the transaction program name (TPN). The LU name routes the connection to the correct LU. However, unlike the LU name, the transaction program name is not known to the SNA network. Within the specified LU, the TPN determines which transaction program is the target of the connection request.

The LU name qualifier specified on the connection request identifies the name space of the resource identified by the TPN. If the name of a global gateway is specified as the LU name qualifier, the name of a global or system resource name can be specified as the TPN. If a private gateway is specified as the LU name qualifier, the TPN indicates a private resource name.

Specifying the Locally Known LU Name and Transaction Program Name

A user program specifies the locally known LU name and the TPN (resource name) in its connection request to identify its communications partner.

To connect to a local or system resource on the same system, or a global resource located in the same TSAF or CS collection as the user program, the user program can specify *IDENT or the system gateway name of the target system (if known) as the LU name qualifier; the target LU name is omitted. The user program specifies the name of the target local, system, or global resource as the resource name (TPN). The resource must have been previously identified to the collection as a local, system, or global resource using the Identify System Service (*IDENT). Local resources are accessible by only the users on that local system.

To connect to a private resource in the same TSAF or CS collection as the user program, the user program specifies *USERID or the system gateway name of the target system (if known) as the LU name qualifier. The user ID of the private resource server virtual machine is also specified as the target LU name. The user program specifies the private resource name as the resource name (TPN).

To locate a resource in an adjacent TSAF collection from a CS collection or an adjacent CS collection from a TSAF collection, a user program specifies the system gateway of the system that is a part of both collections as the LU name qualifier. The target LU and resource name (TPN) vary, depending on the type of resource being accessed. If the user program is accessing a global resource, the target LU name is omitted and the TPN is specified as the name of the global resource. When accessing a private resource, the user ID of the private resource server virtual machine is specified as the target LU. The name of the private resource is specified as the TPN.

To connect to a resource in SNA network, the user program specifies the name of the gateway as the LU name qualifier and the name by which VM knows the remote LU as the target LU name. The user program specifies the name of the target transaction program as the resource name (TPN).

Table 8 on page 170 summarizes how to specify locally known LU names and resource names if a user program does not use symbolic destination names. The user program must identify the complete locally known LU name and target resource name.

If a CMS communications directory is created, the user program can specify a symbolic destination name to identify the locally known LU name and target resource name. For more information about the CMS communications directory, see "Setting Up a CMS Communications Directory" on page 178.

Table 8. Identifying the Target of a Connection Request

Type of Connection		Locally Known LU Name		TPN	Name Space	LU
From	To	LU Name Qualifier	Target LU Name			
A program in a CS collection or a virtual machine in the TSAF collection.	The local resource or system resource on the local system, if they exist, or the global resource in the collection, if it exists.	*IDENT	Omitted (see Note 1)	Local, Global, or System resource name	Local, Global, or System	TSAF or CS collection

Table 8. Identifying the Target of a Connection Request (continued)

Type of Connection		Locally Known LU Name		TPN	Name Space	LU
From	To	LU Name Qualifier	Target LU Name			
A program in a CS collection or virtual machine in the TSAF collection.	The *userid* on the local system, if it exists, or the *userid* in the collection, if it exists. If more than one *userid* exists in the collection, the one that is logged on (if one is) receives the connection request; otherwise the first one found in the collection will be chosen. The requester must be authorized in the user ID's $SERVER$ NAMES file.	*USERID	*userid*	Private resource name	Private	Virtual machine *userid*
A program in a CS collection or virtual machine in the TSAF collection.	The global or system resource on the system identified by *sysgate* in the collection, if it exists, or a global resource in an adjacent CS or TSAF collection, respectively, if it exists and *sysgate* is the system gateway of the node that is common to both collections.	*sysgate*	Omitted (see Note 1)	Global or System resource name	Global, System	VM system identified by *sysgate*
A program in a CS collection or virtual machine in the TSAF collection.	The *userid* on the system identified by *sysgate* in the collection or the *userid* in an adjacent CS or TSAF collection, if it exists, and the *sysgate* is the system gateway of the node that is common to both collections. The requester must be authorized in the $SERVER$ NAMES file of the *userid*.	*sysgate*	*userid*	Private resource name	Private	Virtual machine *userid*

Table 8. Identifying the Target of a Connection Request (continued)

Type of Connection		Locally Known LU Name		TPN	Name Space	LU
From	To	LU Name Qualifier	Target LU Name			
A program in a CS or TSAF collection connected to the target collection through VTAM and AVS.	The system resource on the same VM system as AVS is running in the TSAF or CS collection if it exists, otherwise the global resource in the TSAF or CS collection if it exists.	*loc_gat* AVS global gateway	*rem_gat* AVS global gateway	Global or System resource name	Global, System	TSAF or CS collection
The user ID tied to *loc_gat* in a CS or TSAF collection connected to the target collection through VTAM and AVS.	The user ID tied to *rem_gat* on the same VM system as AVS is running, if it exists, or the user ID in the TSAF or CS collection if it exists. The same search criteria applies here as for *USERID *userid*.	*loc_gat* AVS private dedicated gateway	*rem_gat* AVS private dedicated gateway	Private resource name	Private	Virtual machine user ID tied to *rem_gat*.
A program in a CS or TSAF collection connected to the target collection through VTAM and AVS.	The access user ID on the same VM system as AVS is running in the TSAF or CS collection, if it exists, or the access user ID in the TSAF or CS collection, if it exists. The same search criteria applies here as for *USERID *userid*.	*loc_gat* AVS private nondedicated gateway	*rem_gat* AVS private nondedicated gateway	Private resource name	Private	Virtual machine of the access user ID.
The user ID tied to *loc_gat* in a CS or TSAF collection connected to the target collection through VTAM and AVS.	The access user ID on the same VM system as AVS is running in the TSAF or CS collection, if it exists, or the access user ID in the TSAF or CS collection, if it exists. The same search criteria applies here as for *USERID *userid*.	*loc_gat* AVS private dedicated gateway	*rem_gat* AVS private nondedicated gateway	Private resource name	Private	Virtual machine of the access user ID.

Table 8. Identifying the Target of a Connection Request (continued)

Type of Connection		Locally Known LU Name		TPN	Name Space	LU
From	**To**	**LU Name Qualifier**	**Target LU Name**			
A program in a CS or TSAF collection connected to the target collection through VTAM and AVS.	The user ID tied to *rem_gat* on the same VM system as AVS is running, if it exists or the user ID the TSAF or CS collection, if it exists. The same search criteria applies here as for *USERID *userid*.	*loc_gat* AVS private nondedicated gateway	*rem_gat* AVS private dedicated gateway	Private resource name	Private	Virtual machine user ID tied to *rem_gat*.
Note 1: The actual value of this field depends on the interface being used. If the :luname. tag is being coded in a CMS Communications Directory file, the field is left blank. (See "Setting Up a CMS Communications Directory" on page 178.) If the CP APPC/VM interface is being used, the field is filled with binary zeros. (See *z/VM: CP Programming Services*.) If the CPI Communication interface is being used, CMSPLN, the field can be filled with either blanks or omitted. (See *Common Programming Interface Communications Reference (https://publibfp.dhe.ibm.com/epubs/pdf/c2643999.pdf)*.) If specified on the APPCPASS directory statement, the character "0" (zero) is used. (See "Using APPCPASS Directory Statements" on page 181.)						

Preparing Virtual Machines to Manage Resources

A resource manager program runs in a server virtual machine. Each server virtual machine requires CP directory entries to enable it to manage a resource. The entries in the CP directory vary, depending on the type of resource (local, global, system, or private). For private resources, the server virtual machine also requires entries to the PROFILE EXEC and NAMES files.

This section describes how to set up server virtual machines. See "Managing Local, Global, and System Resources" on page 174 for a description of local, global, or system resource server virtual machines. For information about private resource server virtual machines, see "Managing Private Resources" on page 174.

Defining Unique Resource Names

A VM resource can be located on the local system or on any other system within the TSAF or CS collection. Each global resource name within a TSAF or CS collection must be unique; TSAF or ISFC will enforce this. Each local and system resource name within a VM system must be unique. *IDENT will enforce this. For local, global, and system resources, do not specify the name to be ALLOW, ANY, SYSTEM, or a name that is the same as a user ID on the system.

For private resources, the private server virtual machine owner must ensure unique resource names within the virtual machine. If there are duplicate private resource names, connections will be made to the resource that appears first in the CMS search order.

Defining Unique User IDs

Information that a resource manager program uses for auditing and for checking authorization is called access security information. Within a TSAF or CS collection, the access security information for connections with SECURITY(SAME) is the VM user ID.

Each user ID in a TSAF or CS collection must be assigned to only one person, though the same user ID can exist on more than one system within the collection. TSAF and ISFC do not enforce this limited assignment; enforcement is the responsibility of the system administrator. Failure to make assignments unique can compromise the security of all the systems in the collection.

Connections from the SNA network are not allowed with SECURITY(SAME) unless the remote user ID has been mapped to a local access security user ID. The system administrator can map a remote user ID to a local access security user ID with the AVS command, AGW ADD USERID (see "AGW ADD USERID" on page

273). If the local access security user ID duplicates any user ID in the collection, the duplicated user ID must be assigned to the same person as the remote user ID.

For connections with SECURITY(PGM), the access security information supplied by the connecting program includes a user ID and a password that are validated by CP, so the user ID need not be unique.

With one exception, a private resource server virtual machine should have a unique user ID within a TSAF or CS collection, because the access security user ID is used to route connection requests to the server. The exception applies to private resource servers that make connections only within the same collection and only with applications that select the specific system by using the system gateway. See Figure 12 on page 34 for more details.

Managing Local, Global, and System Resources

The following sections describe the CP directory statements that enable a virtual machine to manage local, global, or system resources.

Establishing IUCV *IDENT Authorization

To authorize a server virtual machine to manage a local, global, or system resource, you must identify the resource in the server virtual machine's CP directory entry. To do so, you must add an IUCV *IDENT directory control statement. See "Authorizing Resource Management" on page 312 for the format of the IUCV *IDENT statement.

Resource managers connect to *IDENT to register the local, system, or global resource they are managing. After the virtual machine connects to *IDENT, it becomes the resource manager for the specified resource. User programs running in authorized requester virtual machines can then connect to these resources. See *z/VM: CMS Application Development Guide* for information on how to write a resource manager.

Note: You must place the IUCV statement before any CONSOLE, SPOOL, LINK, or MDISK statements that may already be in the CP directory entry for the server virtual machine.

Specifying Other CP Directory Statements

You need to add the OPTION directory control statement with the MAXCONN keyword to the CP directory entry of the local, global, or system resource server virtual machine. This OPTION statement indicates the number of IUCV and APPC/VM connections allowed for this virtual machine. Therefore, specify a number large enough to support additional IUCV and APPC/VM connections for:

- Each resource to be managed by the resource manager, and
- The maximum number of users that are expected to be concurrently connected to all of the resources managed by this virtual machine.

Though not required, you should include IUCV ALLOW in the CP directory entry for any server virtual machine. This lets programs communicate between the requester virtual machine and the server virtual machine. It also does not require each user who needs to connect to a specific resource to have directory authorization. For security reasons, however, you may want to explicitly authorize each requester virtual machine that wants to connect to a resource (see "Authorizing Virtual Machines to Connect to Resources and Gateways" on page 178).

Managing Private Resources

Setting up private resource server virtual machines is different from setting up local, global, or system resource server virtual machines. Because private resources are not explicitly identified to CP, private resource server virtual machines do not require an IUCV *IDENT directory control statement. However, you must perform the following tasks to prepare a virtual machine to manage private resources:

- Set up the server virtual machine's CP directory entry

- Establish the CMS environment for the virtual machine
- Create a $SERVER$ NAMES file.

The following sections describe how to set up the private resource server virtual machine. For information on writing a private resource manager, see *z/VM: CMS Application Development Guide*.

Setting Up the CP Directory Entry

You need to add the following directory control statements to the private resource server virtual machine's CP directory entry:

- IUCV ALLOW
- IPL CMS
- OPTION MAXCONN *nnnnn*

The IUCV ALLOW directory control statement lets programs communicate between the requester virtual machine and the private resource server virtual machine. It does not require each user who needs to connect to a specific private resource to have directory authorization. However, you may want to explicitly authorize each requester virtual machine that wants to connect to a private resource (see "Authorizing Virtual Machines to Connect to Resources and Gateways" on page 178).

The IPL CMS directory control statement causes CP to start CMS in the server virtual machine. A private resource server virtual machine need not be logged on when the private resource connection request is sent. If it is not logged on, CP will autolog the private resource server virtual machine.

The OPTION MAXCONN directory control statement indicates the number of IUCV and APPC/VM connections allowed for this virtual machine. Therefore, specify a number large enough to support additional IUCV and APPC/VM connections for the maximum number of users that are expected to be concurrently connected to this virtual machine and the number of connections outbound from this virtual machine. If you do not specify a number, MAXCONN defaults to 64 connections.

Setting Up the Private Server Virtual Machine Environment

To enable the private resource server virtual machine to process connection requests after it has been autologged, you must add the following statements to its PROFILE EXEC:

- SET SERVER ON
- SET FULLSCREEN OFF or SET FULLSCREEN SUSPEND
- SET AUTOREAD OFF

The SET SERVER command enables CMS private resource processing. The SET FULLSCREEN command ensures that session services are deactivated. The SET AUTOREAD command prevents CMS from issuing a console read immediately after command execution. This means that CMS will remain in a `Ready;` state and can start the private resource program. For more information about these CMS commands, see *z/VM: CMS Commands and Utilities Reference*.

Setting Up the $SERVER$ NAMES File

The $SERVER$ NAMES file contains information that enables the private resource server virtual machine to process connection requests. The $SERVER$ NAMES file lists the names of the private resources and the user IDs of the users who are authorized to access those resources.

The $SERVER$ NAMES file should be placed on the file mode A of the private resource server virtual machine. If it is not on file mode A, the $SERVER$ NAMES file must be placed on some other minidisk or SFS directory that is accessed in the CMS search order. Note that if the private resource server virtual machine is being autologged by CP, then the access of the disk or SFS directory containing the $SERVER$ NAMES file should be specified in its PROFILE EXEC.

Table 9 on page 176 describes the contents of the $SERVER$ NAMES file. To create or update the $SERVER$ NAMES file, you can use the SERVER operand of the CMS NAMES command or edit the file. See *z/VM: CMS Commands and Utilities Reference* for more information about the NAMES command.

Table 9. Contents of the $SERVER$ NAMES File

Entry Tag	Usage
:nick.	Specifies the eight character name of the private resource. '&TSAF' is the name reserved for TSAF virtual machines that are using APPC links for communication.
:list.	Specifies the user IDs or user ID nicknames of the users (requesters) authorized to use the private resource.
:module.	Specifies the CMS-invokable name of the resource manager (program) of the private resource specified in the nickname field. The file type of the resource manager is either MODULE or EXEC. The private resource name specified as the nickname will be passed to the resource manager as a parameter.

The $SERVER$ NAMES file contains information used in authorizing connection requests to private resources. Three authorization checks are made when a user program requests a connection to a private resource.

1. CP checks the requester virtual machine's CP directory entry for IUCV directory authorization to connect to the private resource server virtual machine.
2. CMS checks the private resource server virtual machine's $SERVER$ NAMES file for an entry for that resource. This check is NOT case sensitive (for example, TPNABC1 will equal tpnabc1).
3. CMS checks if the user ID specified by the requesting program is listed in the $SERVER$ NAMES file for the requested private resource.

The private resource server virtual machine starts the private resource if all three checks complete correctly. If any authorization check fails, the connection request is severed.

$SERVER$ NAMES Examples

The following entry in the $SERVER$ NAMES file registers the private resource PRTR1, whose private resource manager is LOCPRINT. The entry also authorizes the user IDs USER9 and USER4 to connect to PRTR1:

```
:nick.prtr1         :list.user9 user4
                    :module.locprint
```

You can also specify a nickname or a list of nicknames in a $SERVER$ NAMES entry. The nicknames are resolved by looking up the nickname in the server virtual machine's *userid* NAMES file. For example, you want to authorize a group of users named ACCTUSER to access a private resource ACCTRPT, whose resource manager is ACTRPORT. ACCTRPT resides in the virtual machine whose user ID is FINANCE. The $SERVER$ NAMES file contains the following entry for ACCTRPT:

```
:nick.acctrpt       :list.acctuser
                    :module.actrport
```

The FINANCE NAMES file for the server virtual machine contains the following entries:

```
:nick.acctuser      :list.dept66 mike

:nick.dept66        :list.usera userd user8

:nick.mike          :list.user77  :node.vm8
```

Although the FINANCE NAMES file contains node ID information, this information is not used when user ID authorization is verified.

The following $SERVER$ NAMES file entry enables any user to access the private resource ACCTRPT, whose resource manager is ACTRPORT:

```
:nick.acctrpt       :list.*
                    :module.actrport
```

The asterisk (*) indicates any user requesting a connection is authorized to connect. You should use the asterisk when you want to authorize a private resource for conversation SECURITY(NONE) connections.

Using Private Resource Defaults

The user ID of the private resource server virtual machine is assumed to be in the list field of each private resource managed by that virtual machine. If the access security user ID in the connection request matches the user ID of the private resource server virtual machine, CMS and CP treat this request as if it came from the private resource server virtual machine. CMS does not check the target's $SERVER$ NAMES file for an entry for the user ID, because the private resource server virtual machine is always allowed to request connections to itself. CP does not check the requester virtual machine's CP directory entry for IUCV directory authorization. Also, a connection request with an alternate user ID that matches the access security user ID is handled by both CMS and CP in the same manner as described above; these connections are always allowed.

If the CMS-invokable name of the private resource manager is the same as the private resource name specified in the nickname field of the $SERVER$ NAMES file, the module tag (:module.) can be omitted.

Preparing Virtual Machines to Request Connections to Resources

User programs may request connections to resources to complete tasks or retrieve information. To do so, user programs must:

- Be running in a virtual machine that is authorized to connect to the server virtual machine.
- Identify the resource that is the target of the connection.
- Specify access security information.

To explicitly authorize requester virtual machines to access resources, the system administrator can add IUCV statements to the CP directory entries of each virtual machine (see in “Authorizing Virtual Machines to Connect to Resources and Gateways” on page 178). Alternatively, the system administration can add an IUCV ALLOW statement to the CP directory entries of the TSAF and AVS virtual machines.

A user program requests a connection to a resource by specifying either the resource's locally-known LU name and resource name or a symbolic destination name. Locally-known LU names are described in “Identifying Communications Partners” on page 169. The names of local, global and system resources are identified to CP by *IDENT authorization. The $SERVER$ NAMES file defines the names of private resources.

The symbolic destination name is an alias for the locally-known LU name and the resource name. Symbolic destination names let user programs transparently communicate with resources within a TSAF or CS collection or across an SNA network. By specifying a symbolic destination name on an allocation request, a user program does not need indicate the resource type, the location, or the resource name. A symbolic destination name is defined in a CMS communications directory that is set up by the system administrator. A CMS communications directory is a special NAMES file that defines a symbolic destination name for a locally known LU name and resource name, and its related access security information. See “Setting Up a CMS Communications Directory” on page 178 for more information.

A user program must tell the resource manager the type of access security information being sent and it must specify the access security information. There are three types of access security information: SECURITY(NONE), SECURITY(SAME), SECURITY(PGM). If the user program specifies SECURITY(NONE), it does not send any access security information to the target LU and resource manager. If the user program specifies SECURITY(SAME), it does not send a user ID, but VM sends the logon user ID of the requester virtual machine to the target LU and resource manager. If the user program specifies SECURITY(PGM), it sends a user ID and password to the target LU and the user ID is sent to the target resource manager. The user ID and password sent through the gateway must be valid at the target LU.

A user program's access security information can be specified in several formats:

- The connection request data in the user program (see *z/VM: CMS Application Development Guide* for more information).
- The CMS communications directory entry for the target resource (see "Setting Up a CMS Communications Directory" on page 178 for more information).
- A CP directory statement in the requester virtual machine (see "Using APPCPASS Directory Statements" on page 181 for more information).

Authorizing Virtual Machines to Connect to Resources and Gateways

The system administrator only needs to enter explicit IUCV directory control statements for resources and gateways if the TSAF, AVS, and server virtual machines do not have IUCV ALLOW directory control statements in their CP directory entries. If the TSAF, AVS, and server virtual machines have IUCV ALLOW directory control statements specified in their CP directory entries, the following connections are possible:

- Any user can access any local or system resource server virtual machine in the same system.
- Any user in the TSAF or CS collection can connect to any global or private resource server virtual machine in the collection using the system gateway.
- Any user in the TSAF or CS collection can enter an outbound connection to resources in the SNA network through an AVS gateway.
- Any user in the SNA network can connect to any global and private resource server virtual machine in the TSAF or CS collection, and any system resource server virtual machine located on the same system where AVS is running.

In this case, the server virtual machine must validate security information. CMS provides the security validation for the private resource server virtual machine by checking the $SERVER$ NAMES file. The local, global, or system resource server virtual machine needs to have the resource manager check security.

At times, the system administrator may want to explicitly authorize each requester virtual machine to connect to a resource or use a gateway in the TSAF or CS collection. To explicitly authorize a requester virtual machine to connect to a global resource in a collection or to a local resource in the same system, add an IUCV directory control statement with the resource ID parameter (IUCV *resourceid*) to the requester virtual machine's CP directory entry. To explicitly authorize a requester virtual machine to connect to a private resource in a TSAF or CS collection, add an IUCV directory control statement with the user ID parameter (IUCV *userid*) to the requester virtual machine's CP directory entry. The user ID specified is the user ID of the private resource server virtual machine. To explicitly authorize a requester virtual machine to use a gateway to access a resource in an SNA network, add an IUCV directory control statement with the gateway ID parameter (IUCV *gatewayid*) in the requester virtual machine's CP directory entry.

For a description of these IUCV directory control statements and examples of explicitly authorized TSAF or CS collections, see Appendix A, "IUCV Directory Control Statement," on page 311.

Setting Up a CMS Communications Directory

A CMS communications directory is a special NAMES file that assigns symbolic destination names to locally known LU names and resource names. The CMS communications directory also contains APPC access security information. Using the CMS communications directory facility lets a symbolic destination name specified as the target of a connection request be transformed into a locally known LU name and a resource name that are needed to route the request.

By specifying a symbolic destination name for the target of the connection request, programs can communicate transparently within a TSAF or CS collection or across an SNA network. If a resource is moved to another system, the locally known LU name and resource name can be updated in the CMS communications directory for that symbolic destination name. The individual programs do not have to be changed if this information changes.

To prepare a system or a TSAF collection to use a CMS communications directory, the system administrator needs to:

- Set up the CMS communications directory NAMES files.
- Add statements to the requester virtual machine's CP directory entry (optional).
- Check statements in the SYSPROF EXEC for each system in the TSAF or CS collection that will have a communications directory.
- Set up the requester virtual machine's PROFILE EXEC (optional).

The detailed information for these items and tasks are described throughout the rest of this chapter.

Creating CMS Communications Directory Files

The CMS communications directory file contains information about the location of a target resource and access security information. There are three levels of CMS communications directories:

- System-level (SCOMDIR NAMES)
- User-level (UCOMDIR NAMES)
- IBM-level (ICOMDIR NAMES)

The default names for these files, SCOMDIR NAMES and UCOMDIR NAMES, are defined in the SYSPROF EXEC. However, you can specify a different file name for these files.

Table 10 on page 180 describes the contents of the SCOMDIR and UCOMDIR NAMES files. To create or update a CMS communications directory NAMES file, you can use the COMDIR operand of the CMS NAMES command or edit the file. See *z/VM: CMS Commands and Utilities Reference* for more information about the NAMES command.

The IBM-level communications directory file is provided by IBM and is installed to the CMS 190 disk (S-disk). The communications directory entries in this file are intended for use by IBM and should not be changed.

The system administrator sets up the system-level communications directory, SCOMDIR NAMES (the default name defined in the SYSPROF EXEC). The SCOMDIR NAMES file should be installed on a system-wide or collection-wide location (for example, file mode S). Requester virtual machines should also have read access to the file. The requester virtual machine must access this file mode before the commands that enable communications directory processing and define the communications directories are processed. These commands are described in "Setting Up the SYSPROF EXEC" on page 183.

The owner of a requester virtual machine can set up the user-level communications directory, UCOMDIR NAMES (the default name defined in the SYSPROF EXEC). This file is only needed if the user programs specify symbolic destination names that are not in the SCOMDIR NAMES file or if the owner of the requester virtual machine wants to override definitions in the SCOMDIR NAMES file.

When you enter SET COMDIR FILE, CMS immediately tries to find the specified file and load it into storage. For the SET to take effect, the specified file must be present at the time the SET is entered. If the specified file is not found or cannot be read, then CMS leaves the specified Communications Directory level unchanged.

If your installation has not modified your SYSPROF EXEC, then CMS issues these SET COMDIR commands:

```
SET COMDIR FILE SYSTEM SCOMDIR NAMES *
SET COMDIR FILE USER UCOMDIR NAMES *
SET COMDIR RELOAD
```

CMS attempts the system-level SET with only the S-disk accessed and the user-level SET with only the A-disk and S-disk accessed. Therefore, if you are attempting to use the user-level Communications Directory, you must either put a UCOMDIR NAMES file on your A-disk or enter the appropriate SET

COMDIR FILE command after you access the minidisk or SFS directory containing your Communications Directory file. These ACCESS and SET COMDIR FILE commands can be placed in your PROFILE EXEC if you like.

Table 10. Contents of a CMS Communications Directory File

Tag	Specified Value		
:nick.	Eight-character symbolic destination name for the target resource.		
:luname.	The locally known LU name, which identifies where the resource resides. This name consists of two fields: an LU name qualifier and a target LU name. Each field may contain up to 8 characters and must be separated by a blank. The values that can be used for each depend on the connection:		
	Connection	**LU Name Qualifier**	**Target LU Name**
	Private resource within the TSAF or CS collection	*USERID	Private resource manager's user ID
	Local or system resource, or to a global resource within the TSAF or CS collection	*IDENT or blank	blank
	Outside the TSAF or CS collection	Defined gateway name	Name of partner's LU
	Global or system resource on a specific system in the CS or TSAF collection	System gateway name of target system	blank
	Private resource on a specific system in the CS or TSAF collection	System gateway name of the target system	Private resource manager's user ID
	Resource at remote LU *rem_luname* using AVS gateway *loc_luname*	*loc_luname*	*rem_luname*
:tpn.	The transaction program name as it is known at the target LU. For a local or global resource, this is the resource name identified by the resource manager. For a private resource, this is the nickname specified in the private resource server virtual machine's $SERVER$ NAMES file. For a resource in the SNA network, this is the transaction program name. This resource ID cannot start with a period. The transaction program name '&TSAF' is reserved for TSAF virtual machines that are using APPC links.		
:modename	For connections outside your TSAF or CS collection, this field specifies the mode name for the SNA session connecting the gateway to the target LU. For connections within the TSAF or CS collection, this field specifies a mode name of either VMINT or VMBAT, or it is omitted. Only user programs running in requester virtual machines with OPTION COMSRV specified in their CP directory entry can specify connections with a mode name of VMINT or VMBAT. In the APPN environment, #INTER and #BATCH are commonly used.		
:security.	The security type of the conversation (NONE, SAME, or PGM). Security levels are described in "What Is Conversation Security?" on page 26.		
:userid.	The access security user ID, which is required for security type PGM; it is ignored for other security types.		
:password.	The access security password, which is required for security type PGM; it is ignored for other security types.		

Note: For additional security, you can specify the access security user ID and password on the APPCPASS statement in the source virtual machine's directory, rather than in this file.

CMS Communications Directory Examples

The global resource SALESRPT is located in the same TSAF or CS collection as the user program (no mode name is needed). No access security information is sent by the user program and none is presented to the resource manager (SECURITY(NONE)). To define a symbolic destination name MONSALES for SALESRPT, add the following entry in the CMS communications directory NAMES file:

```
:nick.monsales      :luname.*ident
                    :tpn.salesrpt
                    :security.none
```

The private resource PRTR1 is located in the same TSAF or CS collection as the user program (no mode name is needed). PRTR1 resides in the server virtual machine whose user ID is USER999. No access security information is being sent by the user program, but VM will send the logon user ID of the requester virtual machine to the private resource manager (SECURITY(SAME)). To define a symbolic destination name PRINTIT for PRTR1, add the following entry in the communications directory NAMES file:

```
:nick.printit       :luname.*userid user999
                    :tpn.prtr1
                    :security.same
```

The transaction program name (:tpn.) PRTR1 is the private resource name that appears in the nickname field of the $SERVER$ NAMES file in the USER999 virtual machine (see “$SERVER$ NAMES Examples” on page 176).

The resource (transaction program) RES4 resides at LU COLLECTB in the SNA network. The AVS gateway COLLECTA accesses resources located at COLLECTB. The mode name of the conversation is SECURE. The user program needs to send a user ID (REMUSER1) and a password (REMPASS1) to COLLECTB, where it will be validated and the user ID will be sent to the target resource manager (SECURITY(PGM)). To define a symbolic destination name of INVENTRY for RES4, enter the following in the communications directory NAMES file:

```
:nick.inventry      :luname.collecta collectb
                    :tpn.res4
                    :modename.secure
                    :security.pgm
                    :userid.remuser1
                    :password.rempass1
```

How CMS Communications Directory Resolution is Performed

When VM resolves a symbolic destination name, the user-level directory (if it exists) is checked first. If the user-level communications directory does not contain the specified symbolic destination name, CMS searches the system-level communications directory. If the same symbolic destination name is defined in both the UCOMDIR NAMES and SCOMDIR NAMES files, the information in the SCOMDIR NAMES file is ignored. Lastly, if the user-level and/or system-level communications directories do not contain the specified symbolic destination name, CMS searches the IBM-level communications directory.

If the resource identified in the connection request does not match a symbolic destination name defined in any of the CMS communications directories, then the connection request is processed using the specified resource ID as the name of a resource located in the same TSAF or CS collection as the user program.

Using APPCPASS Directory Statements

For security reasons, you may not want to place the access security user ID and password in a communications directory files. To avoid this situation, you can add an APPCPASS statement to the requester virtual machine's CP directory entry.

The APPCPASS directory statement specifies the locally known LU name of the resource that is the target of the connection request. It also contains the user ID and password of the requester on the target system. The APPCPASS directory statement has the following syntax:

►►─ APPCpass ── *lu_name_qualifier* ── *target_lu_name* ── *userid* ── *password* ─►◄

lu_name_qualifier
specifies the first part of the locally known LU name. See Table 8 on page 170 for a list of valid values.

target_lu_name
specifies the second part of the locally known LU name. See Table 8 on page 170 for a list of valid values.

userid
specifies a user ID that is valid on the target LU.

password
specifies the password that corresponds to the user ID and that is valid on the target LU.

You can have any number of APPCPASS directory statements. If you have multiple APPCPASS statements with the same target LU name and LU name qualifier, but with different user IDs, each can be used independently by CP to obtain passwords. If two APPCPASS statements have the same target LU name, LU qualifier, and user ID, only the first entry will be used.

See *z/VM: CP Planning and Administration* for details on proper placement of the APPCPASS statement within the directory entry.

Communications Directory and APPCPASS Examples

The following entry in the SCOMDIR NAMES file defines the symbolic destination name INVENTRY for the resource RES4, which resides at LU COLLECTB in the SNA network.

```
:nick.inventry      :luname.collecta collectb
                    :tpn.res4
                    :modename.secure
                    :security.pgm
                    :userid.remuser1
                    :password.rempass1
```

To use an APPCPASS directory statement to specify the access security user ID and password information, the following APPCPASS directory statement is added to the requester virtual machine's CP directory entry:

```
appcpass collecta collectb remuser1 rempass1
```

The access security information is then removed from the SCOMDIR NAMES file entry:

```
:nick.inventry      :luname.collecta collectb
                    :tpn.res4
                    :modename.secure
                    :security.pgm
```

When the user program requests a connection to INVENTRY, CMS resolves the symbolic destination name and asks CP to get the access security user ID and password information from the APPCPASS statement.

If the requester has several user IDs at the target LU, it can indicate which user ID should be used for access security information by specifying the user ID in the CMS communications directory entry for the resource. The requester's CP directory entry has APPCPASS statements for each user ID. For example, a requester is authorized to access some resources as USER1 and others as USER2. The following UCOMDIR NAMES entry associates user ID USER1 with the resource PRTRES5:

```
:nick.prntr         :luname.gate5 remote2
                    :tpn.prtres5
                    :modename.fast
                    :security.pgm
                    :userid.user1
```

The following APPCPASS statement specifies the password for USER1:

```
appcpass gate5    remote2  user1 pass1
```

When the user program requests a connection to PRNTR, CMS will resolve the symbolic destination name and ask CP to get the password information for USER1 from the APPCPASS statement.

Setting Up the SYSPROF EXEC

The SYSPROF EXEC shipped with your system contains SET COMDIR commands that define the names of the communications directories and enable communications directory processing. The following five SET COMDIR commands are in the SYSPROF EXEC:

```
  SET COMDIR FILE SYSTEM SCOMDIR NAMES *
  SET COMDIR ON SYSTEM
  SET COMDIR FILE USER UCOMDIR NAMES *
  SET COMDIR ON BOTH
  SET COMDIR RELOAD
```

The SET COMDIR FILE commands define the name of the system-level communications directory file as SCOMDIR NAMES and the name of the user-level communications directory file as UCOMDIR NAMES. SCOMDIR NAMES and UCOMDIR NAMES are the default names for the CMS communications directories; you can choose any file identifier. The SET COMDIR ON commands enable communications directory processing. The SET COMDIR RELOAD command is used to load the $QUEUES$ NAMES and ICOMDIR NAMES files.

CMS, when enabling communications directory processing, reads the communications directory files from disk and makes an image of them in memory. If you update the communications directories on disk, you must enter the SET COMDIR RELOAD command to load the new images into memory. Your updates do not affect existing connections; they only affect connection requests made after the new memory image of the directories has been created.

To disable communications directory processing, enter SET COMDIR OFF.

If a user IPLs CMS in a requester virtual machine without executing the SYSPROF EXEC (for example, by issuing IPL CMS PARM NOSPROF), CMS issues the following SET COMDIR commands to define and enable the system-level communications directory file SCOMDIR NAMES:

```
SET COMDIR FILE SYSTEM SCOMDIR NAMES *
```

and

```
SET COMDIR ON SYSTEM
```

If the user wishes to define and enable user-level communications directory processing the appropriate SET COMDIR commands must be issued. For example, the commands may be added to the PROFILE EXEC.

For more information on the SYSPROF EXEC, see *z/VM: CP Planning and Administration*. See *z/VM: CMS Commands and Utilities Reference* for more information on the SET COMDIR command.

Part 4. TSAF Virtual Machine

This part of the document describes how to set up and use the TSAF virtual machine. It includes the tasks that the system administrator, operator, or both must do to run the TSAF virtual machine.

- Chapter 12, "Preparing to Use the TSAF Virtual Machine," on page 187 describes how to set up and prepare to use the TSAF virtual machine (CP directory entry, servicing, and links).
- Chapter 13, "Setting Up TSAF Collections," on page 195 describes how TSAF collections form and merge, how resources are accessed, how routes are chosen, and how performance can be improved.
- Chapter 14, "Operating the TSAF Virtual Machine," on page 215 describes how to use the TSAF commands to run and maintain the TSAF virtual machine.
- Chapter 15, "Generating TSAF Accounting and Link Statistics," on page 235 describes the contents of the TSAF accounting and link statistics records.
- Chapter 16, "Collecting TSAF Problem Diagnosis Information," on page 239 gives an overview of how to diagnose TSAF virtual machine problems by using dumps and system trace data.

Related Information: See the following sources for more information on the tasks in this section:

Task	Source
Establishing virtual machine environments	Chapter 11, "Setting Up Server and Requester Virtual Machines," on page 165 *z/VM: CMS Planning and Administration* *z/VM: CP Planning and Administration* *z/VM: CMS Commands and Utilities Reference* *z/VM: CP Commands and Utilities Reference*
Installing and servicing code	*z/VM: Installation Guide* *z/VM: VMSES/E Introduction and Reference*
Improving system performance	*z/VM: Performance*

Chapter 12. Preparing to Use the TSAF Virtual Machine

This chapter outlines the necessary preparations for using the TSAF (Transport Services Access Facility) virtual machine within a z/VM environment. It details the setup process, including updating the TSAF PROFILE EXEC, configuring system directory entries, loading program materials, and enabling the message repository. The chapter explains how to define communication links—both TSAF-controlled and VTAM-controlled—and how to rebuild the RUNTSAF module after system updates. It also covers the creation of a CP directory entry with specific IUCV and OPTION statements, and the use of the ATSLINKS FILE to manage link information. Additionally, it describes how to centralize system operations using the Programmable Operator Facility and the Single Console Image Facility (SCIF), which enables efficient message handling and remote system management. Overall, the chapter provides a comprehensive guide to ensure that the TSAF virtual machine is properly configured and integrated into a z/VM system.

This chapter describes the tasks that you must complete before you can use the TSAF virtual machine:

- Set up or update TSAF PROFILE EXEC
- Update TSAF entries in the system directory
- Load the TSAF virtual machine program materials
- Enable the TSAF message repository
- Set up links for communication (this may have already been done during installation)
- Rebuild the RUNTSAF module
- Set up centralized system operations.

Setting Up the TSAF Virtual Machine

As Figure 89 on page 187 shows, TSAF is a separate component of z/VM that runs in a CMS virtual machine. TSAF provides communications by allowing logical APPC/VM paths to span between VM systems that have TSAF running in a virtual machine.

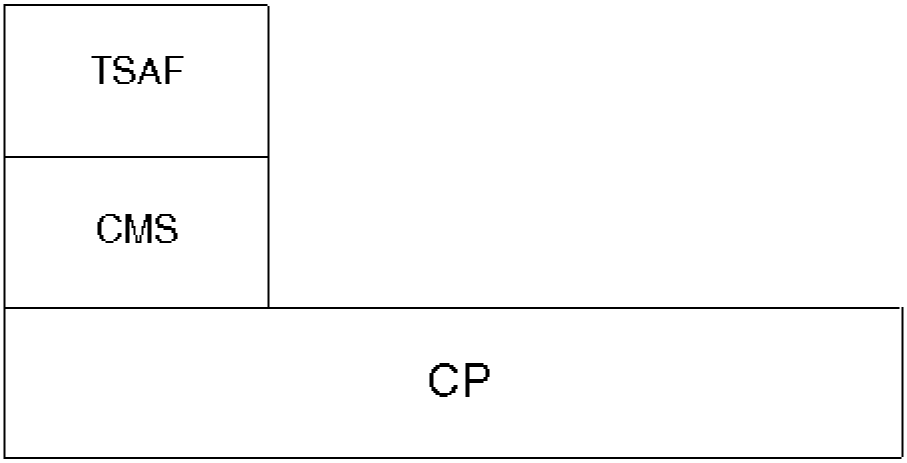


Figure 89. A TSAF Virtual Machine Running in a z/VM System

When a TSAF virtual machine communicates with a TSAF virtual machine on another system, a TSAF collection is formed. The TSAF virtual machines keep track of all the global resources and gateways within the system and within the TSAF collection.

The TSAF virtual machine dynamically configures the TSAF collection. When a link is established between two systems that have not yet joined the TSAF collection, the TSAF virtual machines:

- Exchange information about global resources and gateways on both systems.

- Dynamically configure a new TSAF collection.

Links are discussed in more detail in Chapter 13, "Setting Up TSAF Collections," on page 195.

Loading TSAF Program Materials

You can install TSAF and load the TSAF object code to either the VMPSFS:MAINT*vrm*.TSAF.OBJECT directory for SFS installations or MAINT*vrm*'s 7B2 minidisk for minidisk installations. Because TSAF code is shipped pregenerated, no more processing is needed. See *z/VM: Installation Guide* for more information on installing TSAF.

The most recent build for the TSAF program creates a load map. You can use this load map to process problem information (dumps) for the TSAF virtual machine. Chapter 16, "Collecting TSAF Problem Diagnosis Information," on page 239 has more information on servicing the TSAF virtual machine.

Rebuilding the RUNTSAF MODULE

The RUNTSAF module and the TSAF load map should be rebuilt after the system programmer has applied corrective service or built a system without a particular fix applied. Use the VMFBLD EXEC with the ATSMLOAD build list. For more information on VMFBLD EXEC, see *z/VM: VMSES/E Introduction and Reference*.

Creating the CP Directory Entry

To define the TSAF virtual machine to the z/VM system, you must create a CP directory entry. This section describes the information you must specify for the TSAF virtual machine on the following CP directory entry statements. See "Sample CP Directory Entry" on page 190 for an example of the CP directory entry provided for TSAF):

- IUCV statements
- OPTION directory control statement
- Other directory statements.

z/VM: CP Planning and Administration contains more information about the CP directory entry statements.

IUCV Statements

The following IUCV statements should be specified for the TSAF virtual machine; these statement specify how TSAF can manage resources in the TSAF collection. See Figure 90 on page 191 for examples of these statements.

Statement	Function
IUCV ALLOW	Allows any virtual machine to connect to the TSAF virtual machine. You may not want to let every requester virtual machine connect to the TSAF virtual machine. Instead, you can explicitly authorize each requester virtual machine that wants to connect to a resource or gateway. To do so, include an IUCV directory control statement for a specific resource, gateway, or user ID; see "IUCV Statement Syntax (Gateway Management)" on page 314 or an IUCV ANY statement in each requester virtual machine's CP directory entry. When you explicitly authorize each virtual machine this way, you should also give explicit directory authorization to the TSAF virtual machine residing on the same system as the global resource, private resource, or gateway by using the IUCV directory control statement for the resource or gateway or by using IUCV ANY.
IUCV ANY	Allows the TSAF virtual machine to connect to any virtual machine, resource, or gateway. See "IUCV Statement Syntax (Gateway Management)" on page 314 for more information.

Statement	Function
IUCV *CRM	Allows the TSAF virtual machine to connect to *CRM (Collection Resource Management System Service). Only one virtual machine in a system can connect to *CRM at any time.

OPTION Directory Control Statement

You can specify the following operands on the OPTION directory control statement (Figure 90 on page 191 shows an example of this statement):

OPTION Operand	Function
MAXCONN *nnnnn*	Defines the maximum number of IUCV and APPC/VM connections allowed for this virtual machine. The *nnnnn* value should be large enough to handle all planned intersystem APPC/VM paths that start from, or end at, your system. If you do not specify a number, MAXCONN defaults to 64 connections. **Note:** Message ATSIIN305W indicates that the MAXCONN value is too big for the current TSAF virtual storage. The system administrator may need to increase the TSAF virtual storage size.
COMSRV	Allows the TSAF virtual machine to act as a communications server to route connections for requester virtual machines to other servers. Servers can establish connections to other servers while handling requests for other users. With this option, the TSAF virtual machine or any other communications server can put the user ID of the virtual machine that issued the APPCVM CONNECT in the connection parameter list. When TSAF sends the connection request to the target server virtual machine, the request contains the user ID of the originating requester virtual machine. Without this option, CP would send the connect request with the communications server's user ID. A virtual machine that is authorized as a communications server can specify any SEVER or SENDERR code; CP does not verify the SEVER code. When the virtual machine specifies a SENDERR code, CP does not generate a SENDERR code but, instead, uses the one provided. For more information on APPCVM CONNECT and the SEVER and SENDERR codes, see *z/VM: CP Programming Services*.
DIAG98	Allows the TSAF virtual machine to use DIAGNOSE code X'98' to access the IBM Token Ring Local Area Network subsystem and IEEE 802.3 Local Area Network subsystem on rack-mounted processors. You must specify this option if TSAF uses these types of links. This option is not required if the TSAF virtual machine does not run on a rack-mounted processor or does not use one of these LAN subsystems for communications. Since TSAF is a 370 mode application, the DIAGNOSE code X'98' instruction will require that an RIO370 area be defined. See *z/VM: CP Programming Services* for more information about DIAGNOSE code X'98' and RIO370 area.
ACCT	Causes the TSAF virtual machine to generate accounting records.

OPTION Operand	Function
CONCEAL	Places the TSAF virtual machine in a protected application environment at logon time. A protected application environment means the following: • Multiple attention interrupts do not cause the TSAF virtual machine to drop into CP mode. • TERMINAL BRKKEY is set to NONE. • If the TSAF virtual machine changes a shared page, CP tries to resume execution in the virtual machine, before it starts an automatic re-IPL. • CP starts an automatic re-IPL when it finds errors, such as: a virtual machine disabled wait, a paging error, a PSW that is not valid, an external interrupt loop, or a translation exception. • If a TSAF or CMS ABEND occurs, OPTION CONCEAL causes CP to start an automatic re-IPL.

Other Directory Statements

The CP directory should contain the following directory statements (Figure 90 on page 191 shows examples of these statements). See *z/VM: CP Planning and Administration* for more information about specific devices or system directory entries.

Statement	Function
MACHINE XA	A MACHINE XA statement in the TSAF CP directory entry may be used to create a XA virtual machine. This will allow the use of CMS Level 12 and above. However, the TSAF virtual machine must run in 370 accommodation mode for TSAF to execute.
MDISK	Identifies a 191 minidisk for the TSAF virtual machine; this disk may contain files such as the TSAF PROFILE EXEC and link definition file, ATSLINKS FILE A1. Sufficient storage for the TSAF virtual machine should be forty 1024-byte blocks or 1-cylinder.

Sample CP Directory Entry

Figure 90 on page 191 shows the sample CP directory entry provided for the TSAF virtual machine. This CP directory entry defines the following characteristics:

- Privilege class G
- 16MB of virtual storage
- Dedicated links.

Note: The virtual storage size for the TSAF virtual machine may range from 6MB to 2047MB. Because TSAF uses 24 bit addressing, it will not use storage above the 16MB line. However, the additional storage may be used by CMS.

```
IDENTITY TSAFVM TSAFVM 16M 16M G
 INCLUDE IBMDFLT
 BUILD ON * USING SUBCONFIG TSAFVM-1
 ACCOUNT 1 TSAFVM
 OPTION MAXCONN 256 COMSRV DIAG98 ACCT CONCEAL
 MACH ESA
 IUCV ANY
 IUCV ALLOW
 IUCV *CRM
 IPL CMS PARM AUTOCR

SUBCONFIG TSAFVM-1
 LINK MAINT 193 193 RR
 MDISK 191 3390 0910 002 MO1W01 MR RTSAFOBJ WTSAFOBJ MTSAFOBJ
DEDICATE 4A0 300
```

Figure 90. Sample CP Directory Entry for the TSAF Virtual Machine

Accessing TSAF Service Updates

To access the TSAF object code and service updates for an SFS installation, you must:

- Be enrolled as a user of the VMSYS file pool.
- Be explicitly authorized to access the VMPSFS:MAINT*vrm*.TSAF subdirectories OBJECT, ALTERNATE, INTERMED, and PRODUCTION, which contain the TSAF object code and updated TSAF execs and modules.
- Be explicitly authorized to access MAINT*vrm*'s 3C4 and 3C2 minidisks which contain local modifications and samples, respectively.
- Access these directories and minidisks in the TSAF virtual machine's PROFILE EXEC.

To access the TSAF object code and service updates for a minidisk installation, you must:

- Be explicitly authorized to access MAINT*vrm*'s 7B2, 7D6, 7D4, 7D2, 3C4, and 3C2 minidisks.
- Access these minidisks in the TSAF virtual machine's PROFILE EXEC.

Creating the TSAF PROFILE EXEC

Because TSAF runs in a virtual machine, you can create a PROFILE EXEC to automatically issue certain commands each time the system is IPLed. These commands enable the TSAF virtual machine to access or define the following information:

- TSAF message repository

  To ensure access to TSAF message repository files, specify the following SET LANGUAGE statement:

  ```
  set language ameng (add ats user
  ```

- TSAF code and service updates (optional)

  To access TSAF object code and service updates, you must access the disks or SFS directories that have been defined in the CP directory entry for the TSAF virtual machine. See “Accessing TSAF Service Updates” on page 191 for additional information.

- Run time information

  To ensure that the TSAF timer functions are accurate, you can include a SET TIMER command in the PROFILE EXEC. The SET RUN ON command ensures that TSAF will continue to run if the TSAF virtual machine is in CP Read state.

  TSAF is a 370 mode application and must run with 370 accommodation mode enabled. To ensure that this is true, you must add a 'CP SET 370ACCOM ON' command to TSAF's PROFILE EXEC.

Figure 91 on page 192 is an example of a PROFILE EXEC file for the TSAF virtual machine. In this example, the TSAF virtual machine accesses the MAINT user ID's 193 disk as file mode D; the TSAF executable module and message repository are stored on this disk. The SET LANGUAGE command enables TSAF to access the TSAF message repository. Finally, the RUNTSAF command is specified to automatically start the TSAF virtual machine.

```
/*   Sample TSAF PROFILE EXEC                              */
Trace 0
'ACCESS 193 D'
'CP SET RUN ON'
'SET LANGUAGE AMENG ( ADD ATS USER'
'CP SET 370ACCOM ON'
'CP SET TIMER REAL'
'RUNTSAF'
```

Figure 91. Sample PROFILE EXEC for the TSAF Virtual Machine

Enabling the TSAF Message Repository

The source file for TSAF messages is the ATSUME*x* REPOS file (*x* is the country code for a particular language). When you use the VMFNLS EXEC to apply updates to the ATSUME*x* REPOS file, the ATSUME*x* TEXT file is generated.

You must ensure that the ATSUME*x* TEXT files are available to the TSAF virtual machine. During installation, the ATSUME*x* TEXT files are installed on MAINT*vrm*'s 7B2 disk and MAINT's 193 disk. If you are installing TSAF from an SFS directory, the text files should be placed in the VMPSFS:MAINT*vrm*.TSAF.OBJECT file pool.

For example, Figure 90 on page 191 shows that a link to the MAINT 193 disk has been defined in the CP directory entry for the TSAF virtual machine. The ACCESS command, shown in the PROFILE EXEC in Figure 91 on page 192, ensures that the TSAF virtual machine will access this disk each time it is started.

The SET LANGUAGE statement, also shown in Figure 91 on page 192, enables TSAF to use the message repository file after service has been applied. This statement loads the repository and parser tables into the TSAF virtual machine. If your system is running in a language other than American English, see *z/VM: CMS Commands and Utilities Reference* for information on this command format.

If you do not add the SET LANGUAGE entry in the PROFILE EXEC, TSAF cannot access the message repository. When TSAF tries to issue a message, you will receive the following message:

```
DMSMGM813E  ATS repository not found, message msgid cannot be retrieved
```

See *z/VM: CMS Commands and Utilities Reference* for more information about the SET LANGUAGE command. For information about the VMFNLS EXEC, see *z/VM: VMSES/E Introduction and Reference*.

Setting Up Links for Communication

This section describes how communication links are defined and identified to TSAF.

Defining Real Communication Links

The real communication links that will be used to communicate with other systems must be defined to the TSAF virtual machine. (These real communication links may have already been defined during installation.)

To define real communication links, you may need to specify RDEVICE statements in the system configuration file. The RDEVICE statements identify the real device address and the device type of the link. TSAF may use some types of communication links that do not need to be defined by RDEVICE statements (for example, VTAM-controlled links). For information about the types of links that must be defined with RDEVICE statements, see *z/VM: CP Planning and Administration*.

Dedicating Links to TSAF

After the real I/O devices are defined to the system, you must ensure that TSAF can communicate over the links. This process differs for TSAF and VTAM-controlled links.

Dedicating TSAF-Controlled Links

For TSAF-controlled links, you must dedicate and attach the links to the TSAF virtual machine. To do so, you specify DEDICATE statements in the system directory entry of the TSAF virtual machine. For example, the following DEDICATE statement (shown in Figure 90 on page 191) dedicates the real device at address 300 to TSAF as the virtual device with address 4A0.

```
dedicate 4a0 300
```

Specify the virtual address when you enter the ADD LINK command. See "ADD LINK" on page 217 for additional information.

For each system that is part of a local area network (LAN) that is dedicated to the TSAF virtual machine, you must dedicate four devices for the Continuously Executing Transfer Interface (CETI) group associated with each LAN system. A CETI group consists of four consecutive device addresses that have been defined to CP. See *z/VM: CP Planning and Administration* for more information on setting up an adapter. The real device addresses and the virtual device addresses must be consecutive. The real device addresses are defined by RDEVICE statements in the system configuration file. The use of these link types also requires that the TSAF virtual machine be allowed to use DIAGNOSE code X'98' and that an RIO370 area be defined. For more information, see *z/VM: CP Planning and Administration*.

For example, the following statements dedicate a LAN system's CETI group real addresses (500, 501, 502, and 503) to TSAF as the virtual device with addresses 300, 301, 302, and 303:

```
dedicate 300 500
dedicate 301 501
dedicate 302 502
dedicate 303 503
```

Virtual subchannel addresses do not have to be the same as the real subchannel addresses, but they must be consecutive and start within the range of X'00' and X'F8'.

To attach the links, you enter the ATTACH command, which is described in the *z/VM: CP Commands and Utilities Reference*.

Establishing VTAM-Controlled Links

This section describes the steps needed to establish an APPC link between two z/VM systems that are connected by a physical VTAM link. These steps should be taken on each system in the TSAF collection that is directly connected to another system by an APPC link. After these steps are completed, you do not have to repeat them each time you establish communications to that other system.

Perform the following steps to establish an APPC link between two systems ("Establishing an APPC Link" on page 202 shows an example of each step):

1. Enter AGW ACTIVATE GATEWAY PRIVATE to define a dedicated private gateway for the connection (see "Purpose" on page 271).
2. Enter AGW CNOS to define session limits (see "Purpose" on page 274).

   Note: You can add these commands to the AGWPROF GCS exec in the AVS virtual machine so that they take effect automatically each time the AVS virtual machine is logged on.
3. Add APPCPASS statements to the TSAF virtual machine's CP directory entry (see "Using APPCPASS Directory Statements" on page 181)
4. Create a CP directory entry on your system for the remote TSAF virtual machine (see "Creating the CP Directory Entry" on page 188)
5. Stop the TSAF virtual machine (enter STOP TSAF), and:

- Create a user-level CMS communications directory NAMES file entry in the TSAF virtual machine (see "Setting Up a CMS Communications Directory" on page 178)
- Create a $SERVER$ NAMES file entry in the TSAF virtual machine (see "Setting Up the $SERVER$ NAMES File" on page 175)
- Add entries to the TSAF virtual machine's PROFILE EXEC (see "Creating the TSAF PROFILE EXEC" on page 191).

6. Restart the TSAF virtual machine (enter RUNTSAF).

Note that multiple active links from one TSAF virtual machine to another TSAF virtual machine are not supported.

Updating the ATSLINKS FILE

TSAF stores information about the links it can use in a CMS file called ATSLINKS FILE. You will get error messages if you store the ATSLINKS FILE file on any file mode other than TSAF's file mode A.

The ATSLINKS FILE contains the virtual device address or the symbolic destination name for each link that TSAF can use. The virtual device address or the symbolic destination name can start in any column in the ATSLINKS FILE (TSAF writes the device address in columns 2 through 9).

Each time you enter the ADD LINK command to identify a link to the TSAF virtual machine, the link is automatically added to the ATSLINKS FILE. (You can also create this file and add the link information before you start TSAF; in this case, you do not need to enter ADD LINK commands after TSAF is started.) Each time TSAF is started, it checks the ATSLINKS FILE for information about the TSAF links that have been defined. Link information is only removed from this file when you enter the DELETE LINK command to remove a link from TSAF or if you manually edit the contents of the ATSLINKS FILE.

Setting Up Centralized System Operations

To centralize system operations within the TSAF collection, you can use the Programmable Operator Facility and the Single Console Image Facility. You can enable a central site to receive messages from systems in the TSAF collection that cannot be automatically handled at the distributed system.

The Programmable Operator Facility allows remote operation of systems in a distributed data processing environment. It does this by intercepting all messages and requests directed to its virtual machines and by handling them according to preprogrammed actions. These messages and requests are sent to a logical operator.

The Single Console Image Facility (SCIF) lets one operator control multiple disconnected virtual machines. The operator's user ID is defined in the service virtual machine's CP directory entry (that is, the TSAF virtual machine's CP directory entry) with the CONSOLE directory control statement so that SCIF sends the service virtual machine's console output to the operator virtual machine.

To use the Programmable Operator Facility and SCIF together, use SCIF to send messages generated by the TSAF virtual machine (which is running disconnected) to an operator's virtual machine that is using the Programmable Operator Facility. The programmed operator acts on the messages sent to it by SCIF.

To centralize system operations in the TSAF collection, you need either the Remote Spooling Communications Subsystem (RSCS) or, if you have VTAM on each system, NetView. When RSCS is installed, the Programmable Operator Facility sends messages through RSCS to a central site virtual machine where a real operator is located. This central operator can act on the messages issued by the local TSAF virtual machines. When NetView is installed, the Programmable Operator Facility will forward messages to the NetView virtual machine. NetView sends the messages to the central site's NetView virtual machine, where a real operator is located. This central operator gets messages from each system and can act on the messages issued by the local TSAF virtual machines.

For information about the Programmable Operator Facility, see *z/VM: CMS Planning and Administration*. For information about SCIF, see *z/VM: Virtual Machine Operation*.

Chapter 13. Setting Up TSAF Collections

This chapter provides a comprehensive guide on setting up and managing TSAF (Transport System Access Facility) collections in IBM's z/VM environment. A TSAF collection consists of up to eight VM systems that collaborate to share and access resources across systems. The chapter outlines the structure of TSAF collections, the process of identifying TSAF virtual machines, and the steps for connecting to various types of resources, including global, private, and shared file system resources. It also details how to establish APPC links between systems using VTAM-controlled communication. Emphasis is placed on maintaining a single, reliable TSAF collection through strategic link configuration and startup sequences. The chapter concludes with performance considerations, which highlight the impact of link types and routing strategies on communication efficiency and system responsiveness.

TSAF Collection Structure

A group of VM systems that each have the TSAF virtual machine component installed and running can form a TSAF collection. A TSAF collection can have up to eight VM systems. Each system in the TSAF collection must have two unique identifiers:

- The processor ID, which is preassigned by the processor manufacturer
- The system identifier, or node ID, which is assigned when the system is installed.

The SYSTEM NETID file, an existing CMS file, associates each processor ID with its corresponding node ID. You must ensure the processors within a TSAF collection have unique node IDs. For information on how to update the SYSTEM NETID file, see *z/VM: CMS Planning and Administration*.

TSAF Collection Examples

This section includes scenarios that describe how to set up a TSAF collection and allow resources to be accessed. The scenarios are:

- Setting up a TSAF collection (described in "Setting Up a TSAF Collection" on page 195)
- Connecting to a global resource (described in "Connecting to a Global Resource" on page 196)
- Connecting to a Shared File System (SFS) file pool (described in "Connecting to a Shared File System File Pool" on page 198)
- Connecting to a private resource (described in "Connecting to a Private Resource" on page 200)
- Establishing an APPC Link (described in "Establishing an APPC Link" on page 202).

The first scenario, "Setting Up a TSAF Collection" on page 195, describes how a TSAF collection is set up and resources are accessed. The remaining scenarios involve:

- Two systems, each with a TSAF virtual machine. The TSAF virtual machines are TSAFa and TSAFb.
- One resource manager that manages a resource.
- Two programs that want to use the resource. One program is on the local system and the other is on a remote system in the TSAF collection.
- Two systems, each with a TSAF virtual machine and connected by a physical VTAM link, that form a TSAF collection.

Setting Up a TSAF Collection

The following scenario describes how a virtual machine is identified as the TSAF virtual machine and how a TSAF collection is formed.

Step 1—TSAF Virtual Machine Identifies Itself

When the TSAFa virtual machine begins running (see Figure 92 on page 196), it requests a connection to the Collection Resource Management System Service (*CRM). Because no other local virtual machine is already connected to *CRM and TSAFa is authorized, CP accepts TSAFa as the TSAF virtual machine and starts a connection.

Through *CRM, TSAFa gets information about the resources and gateways that have been defined on the local system. If any virtual machines in the system had identified themselves as the managers of any global resources, then, on request, CP would send those resource names to TSAFa. In this example, there are no previously established resources.

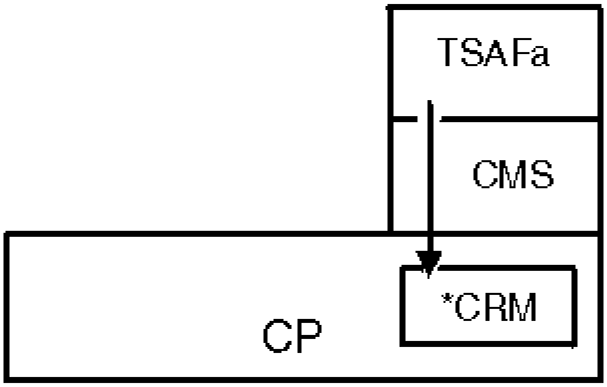


Figure 92. TSAF Virtual Machine Identifying Itself

Step 2—TSAF Virtual Machines Exchange Information

Now that TSAFa is the TSAF virtual machine for the local system, this system is a TSAF collection of one system. This single system collection tries to join itself with another TSAF collection to form a larger collection. TSAFa sends out data along each physical link that has been defined to TSAF. So TSAFa can join the TSAF collection on the other end of the link, the TSAF virtual machines, on the other end of each link, exchange the following information with TSAFa:

- Names of the resources and gateways that TSAFa knows
- Names of other resources and gateways in the TSAF collection that the remote TSAF virtual machines know.

In Figure 93 on page 196, there is only one other TSAF virtual machine in the remote TSAF collection (TSAFb) and it does not know about any resources yet. The two TSAF collections merge to form a TSAF collection of two systems.

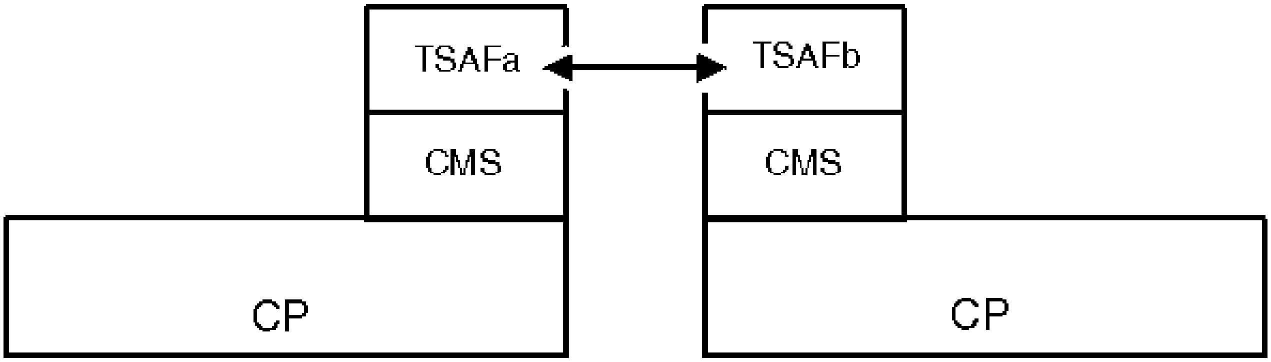


Figure 93. TSAF Virtual Machines Exchanging Information

Connecting to a Global Resource

In the following scenario, the global resource manager RESMGR manages the global resource RES1. The TSAF collection is set up as described in “Setting Up a TSAF Collection” on page 195. The programs that want to use the resource, RES1, are PGMa and PGMb.

Step 1—A Resource Manager Requests to Manage a Global Resource

When the resource manager RESMGR enters the TSAF collection (see Figure 94 on page 197), it issues an IUCV CONNECT to the Identify System Service (*IDENT). CP then recognizes RESMGR as the manager of the global resource, RES1. CP notifies TSAFa over its *CRM connection that RESMGR wants to manage the resource, RES1.

TSAFa and TSAFb then agree that RESMGR can be the manager of the resource, RES1, in the TSAF collection. TSAFa notifies the local CP, which adds the global resource, RES1, to its system resource table and accepts the connection from RESMGR to *IDENT. TSAFa and TSAFb add RES1 to their global resource tables. If this global resource had already been defined in the TSAF collection, CP would have rejected the identify request.

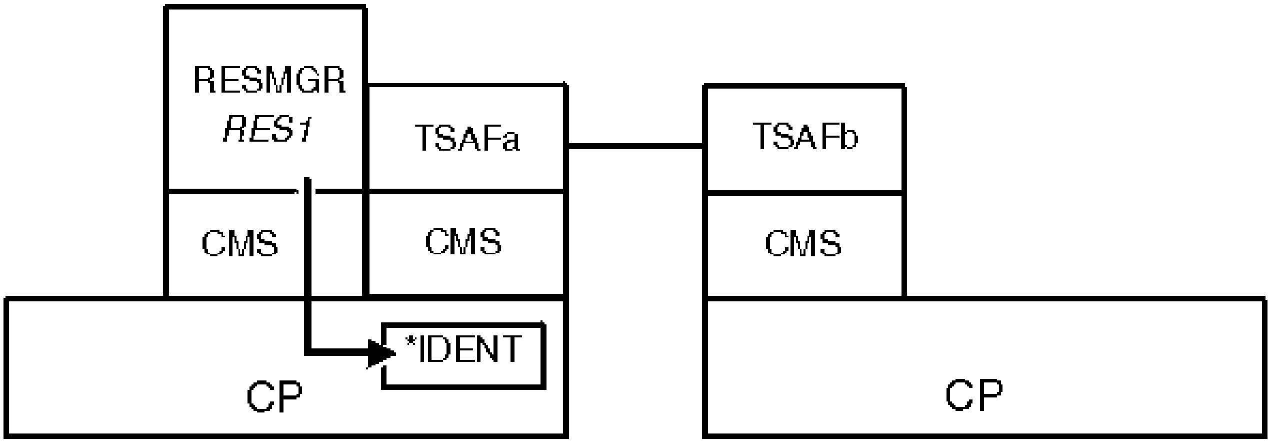


Figure 94. Resource Manager Requesting to Manage a Global Resource

Step 2—A Local Program Connects to the Resource Manager to Access the Resource

PGMa requests to connect to resource RES1. The local CP finds RES1 in its resource table and connects PGMa to RESMGR (see Figure 95 on page 197). After the connection is complete, APPC/VM communication can begin over the established path. Because the resource is on the local system, there is no need to go through the TSAF virtual machine.

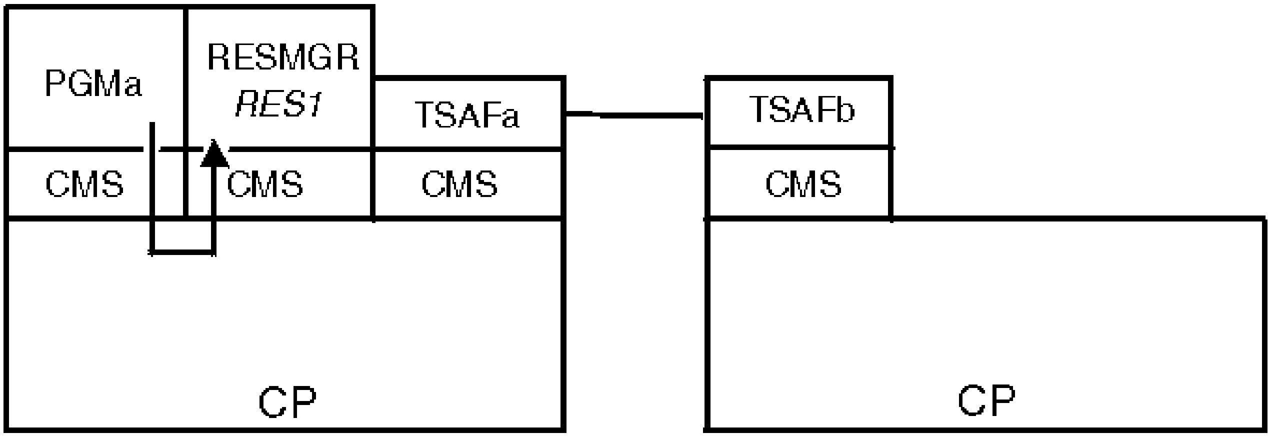


Figure 95. A Local Program Accessing a Global Resource

Step 3—A Remote Program Connects to the Resource Manager to Access the Resource

PGMb requests to connect to resource RES1. Because the local CP does not find RES1 in its system resource table, it connects the user to TSAFb. TSAFb finds RES1 in its resource table and sends the connection request to TSAFa. TSAFa issues the connection request to the resource, RES1, for PGMb. The local CP finds RES1 in its system resource table and gives the connection request to RESMGR (see Figure 96 on page 198).

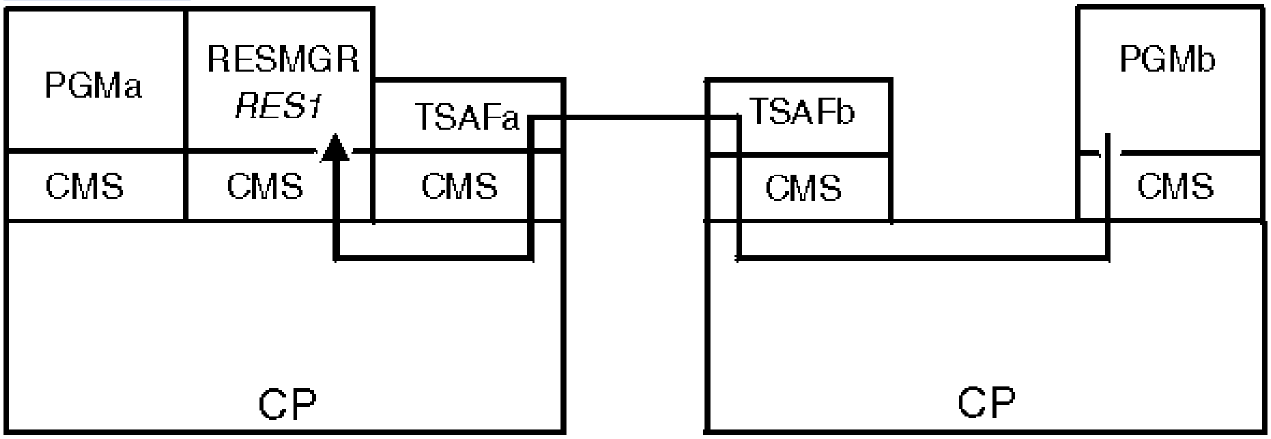


Figure 96. Remote Program Accessing a Global Resource

This connection actually consists of two APPC/VM paths (one between PGMb and TSAFb and the other between TSAFa and RESMGR) and a communications link (between the TSAFa and TSAFb virtual machines). The result is a logical APPC/VM connection between PGMb and RES1. This connection logically spans more than one system.

Step 4—Sends and Receives for Local and Remote Programs

After the connections are complete, TSAF routes the various send and receive requests between the resource manager RESMGR and the user program that wants access to the global resource RES1 (see Figure 97 on page 198).

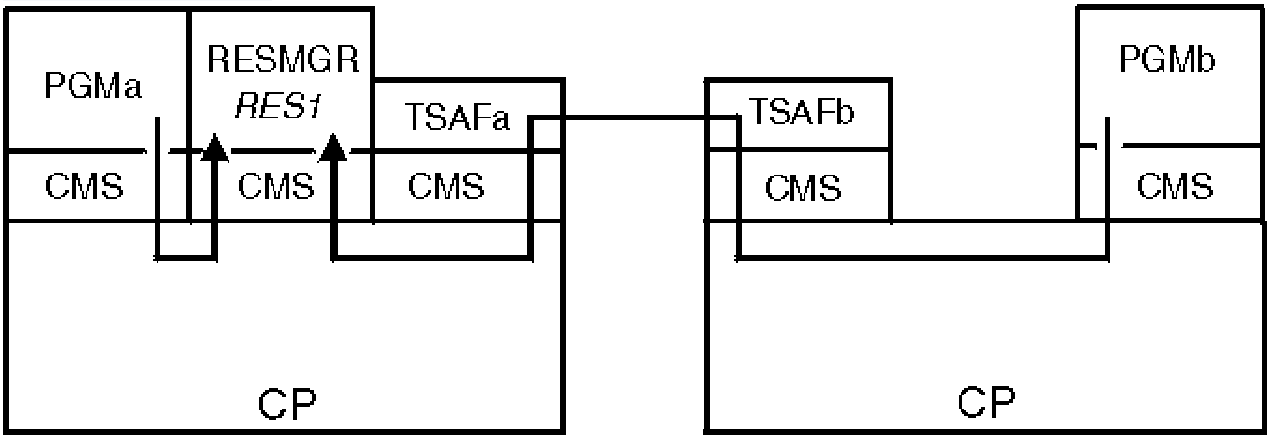


Figure 97. Sending and Receiving

Connecting to a Shared File System File Pool

In the following scenario, the Shared File System (SFS) file pool server POOLSRV manages the file pool POOL15 (a global resource). The TSAF collection is set up as described in "Setting Up a TSAF Collection" on page 195. The user virtual machines that want to access the file pool POOL15 are TONY and JANET.

See *z/VM: CMS File Pool Planning, Administration, and Operation* for more information about SFS.

Step 1—File Pool Server Requests to Manage a File Pool

The file pool server POOLSRV requests to manage the global resource file pool POOL15 by connecting to *IDENT. This request is issued internally when the file pool server administrator issues the FILESERV START command. TSAFa and TSAFb agree that POOLSRV can manage POOL15. POOL15 is then added to the resource tables.

The system administrator of POOL15 enrolls the users TONY and JANET. The top directories POOL15:TONY and POOL15:JANET are generated in POOL15.

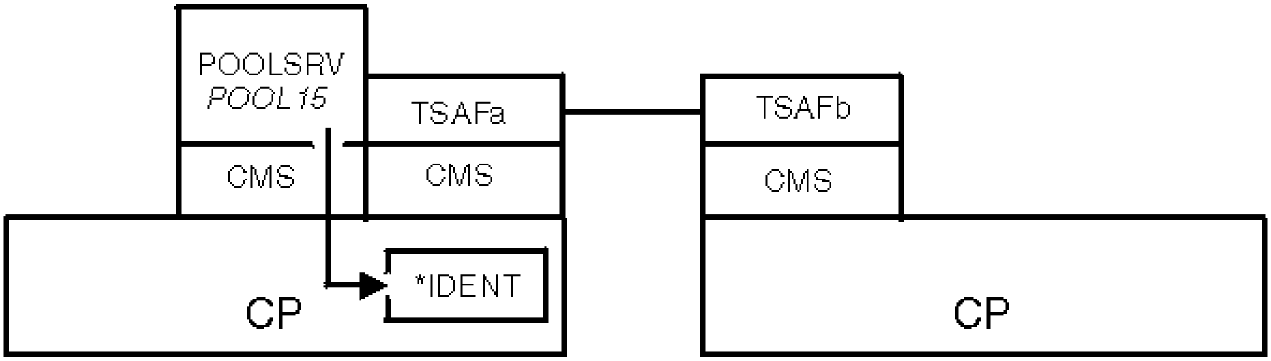


Figure 98. File Pool Server Requesting to Manage a File Pool

Step 2—A Local User Connects to the File Pool Server to Access the File Pool

The user TONY enters a CMS ACCESS command for his top directory, that is maintained in the file pool POOL15. CMS in the TONY virtual machine tries to connect to the global resource POOL15. The local CP finds POOL15 in its resource table and connects TONY to POOLSRV. The user ID TONY is presented to POOLSRV for authorization to access POOL15. Because TONY is an authorized user, TONY is connected to file pool POOL15 (see Figure 99 on page 199). Because the file pool is on the local system, there is no need to go through the TSAF virtual machine.

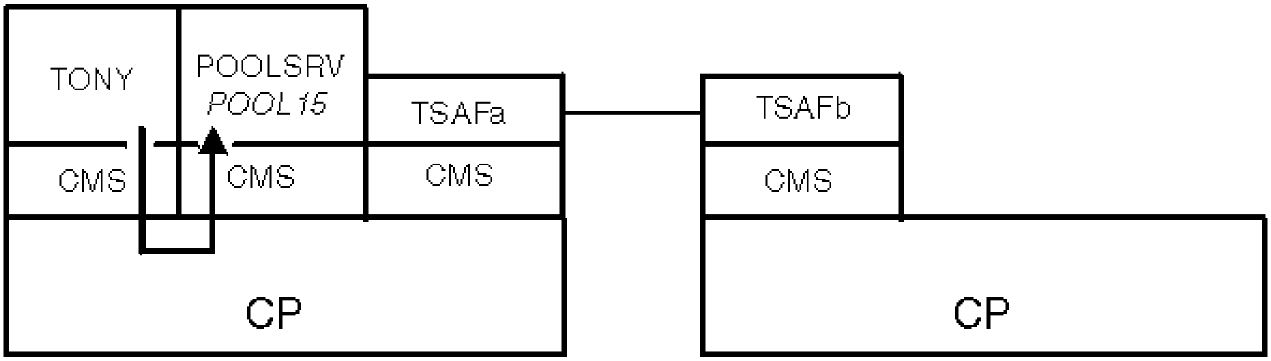


Figure 99. Local User Accessing the File Pool

Step 3—A Remote User Connects to the File Pool Server to Access the File Pool

The user JANET enters a CMS ACCESS command for her top directory, that is maintained in the file pool POOL15. CMS in the JANET virtual machine tries to connect to the global resource POOL15. Because the local CP does not find POOL15 in its system resource table, it connects the user to TSAFb. TSAFb finds POOL15 in its resource table and sends the connection request to TSAFa. TSAFa issues the connection request to the resource POOL15 for JANET. The local CP finds POOL15 in its system resource table and gives the connection request to POOLSRV.

The user ID JANET is presented to POOLSRV for authorization to access POOL15. Because JANET is an authorized user, JANET is connected to file pool POOL15 (see Figure 100 on page 200).

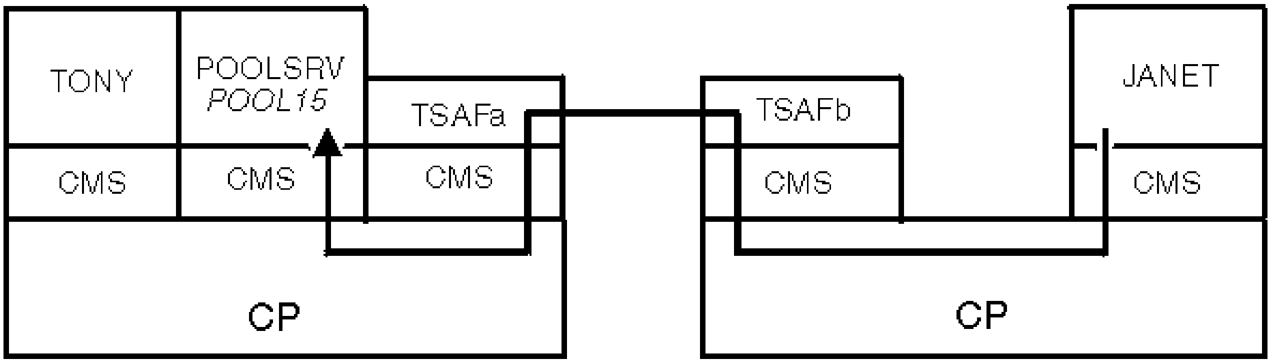


Figure 100. Remote User Accessing the File Pool

Step 4—CMS Commands from Local and Remote Users

After the connections are complete, TSAF routes the various CMS commands entered by TONY and JANET that refer to POOL15 between the file pool server POOLSRV and the user virtual machines that want access to file pool POOL15 (see Figure 101 on page 200).

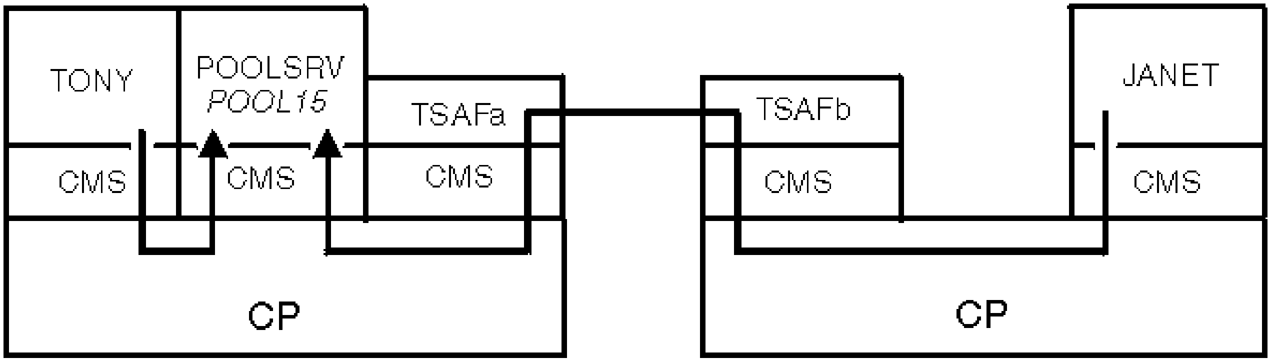


Figure 101. Processing CMS Commands

Connecting to a Private Resource

In the following scenario, the private resource manager PRIMGR2 manages the private resource RES2. PRIMGR2 resides in the virtual machine whose user ID is USR119. The TSAF collection is set up as described in "Setting Up a TSAF Collection" on page 195. The programs that want to use the resource, RES2, are PGMc and PGMd. PGMc runs in a virtual machine whose user ID is USER6, and PGMd runs in a virtual machine whose user ID is USER8.

One CMS communications directory is set up for the entire TSAF collection. This directory is stored in a file pool and all requester virtual machines are enrolled in the file pool. Each system in the TSAF collection has:

- Been enrolled in the file pool.
- Accessed the SFS directory in which the CMS communications directory resides.
- Been enabled for CMS communications directory processing.

Step 1—A Resource Manager Prepares to Manage a Private Resource

The server virtual machine owner prepares to manage the private resource RES2 by creating the following $SERVER$ NAMES file entry to authorize USER6 and USER8 to access RES2:

```
:nick.res2          :list.user6  user8
                    :module.primgr2
```

The system administrator creates the following entry for RES2 in the collection communications directory:

```
:nick.res2          :tpn.res2
                    :luname.*userid usr119
                    :security.same
```

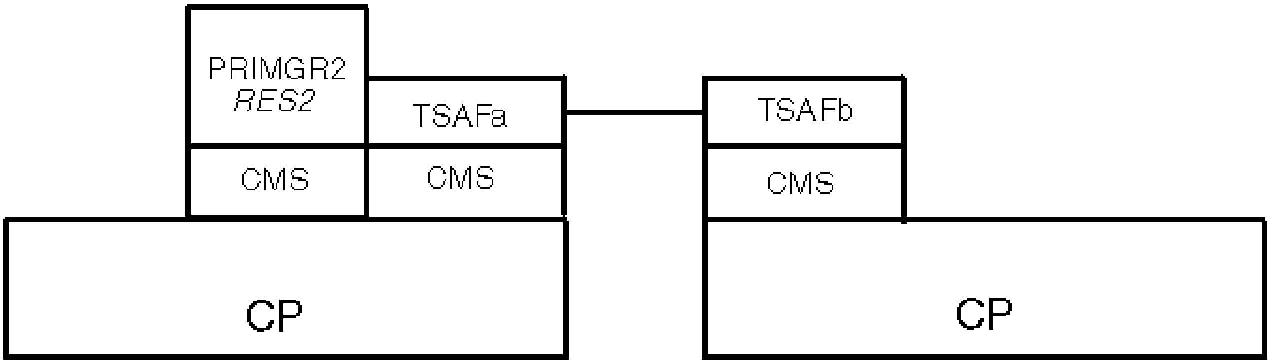


Figure 102. Resource Manager Preparing to Manage a Private Resource

Step 2—A Local Program Connects to the Resource Manager to Access the Private Resource

PGMc requests to connect to RES2. CMS translates this symbolic destination name into the locally-known LU name and the private resource name RES2. The local CP finds USR119 in its directory and connects PGMc to the USR119 virtual machine. If USR119 is not logged on, it is autologged and the resource manager PRIMGR2 is started. CP queues the connection pending interrupt for USR119's CMS virtual machine. USR119 receives the interrupt when it enables for APPC/VM interrupts. When CMS receives the interrupt, it checks its $SERVER$ NAMES file to see if USER6 is authorized. The connection to RES2 is completed and APPC/VM communications can start over the established path (see Figure 103 on page 201).

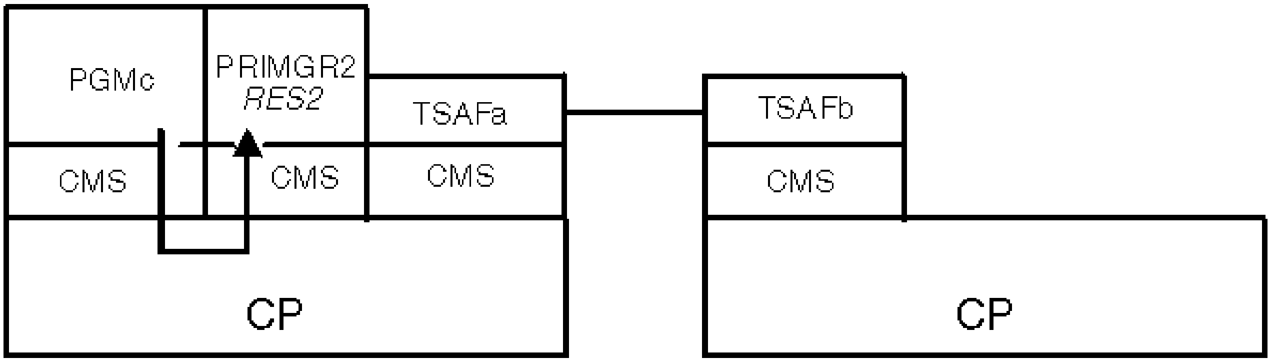


Figure 103. Local Program Accessing a Private Resource

Step 3—A Remote Program Connects to the Resource Manager to Access the Private Resource

PGMd requests to connect to RES2. CMS translates this symbolic destination name into the locally known LU name and the private resource name RES2. The local CP does not find USR119 in its directory and it requests TSAFb to determine the location of USR119. TSAFb sends a request to TSAFa to determine if USR119 is located on that system. TSAFa informs TSAFb that USR119 resides on its system. TSAFb informs its local CP that USR119 has been found and its CP routes the user to TSAFb which sends the connection request to TSAFa.

TSAFa then issues the connection request to USR119 and connects PGMd to the USR119 virtual machine. If USR119 is not logged on, it is autologged and the resource manager PRIMGR2 is started. CP queues the connection pending interrupt for USR119's CMS virtual machine. USR119 receives the interrupt when it enables for APPC/VM interrupts. When CMS receives the interrupt, it checks its $SERVER$ NAMES file to see if USER8 is authorized. The connection to RES2 is completed and APPC/VM communications can begin over the established path (see Figure 104 on page 202).

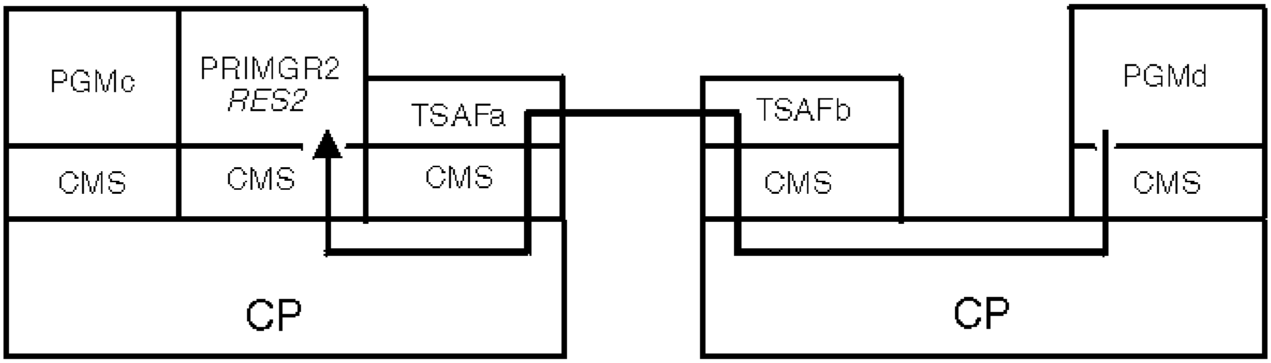


Figure 104. Remote Program Accessing a Private Resource

Step 4—Sending and Receiving

After the connections are complete, TSAF routes the send and receive requests between the resource manager PRIMGR2 and PGMd (see Figure 105 on page 202).

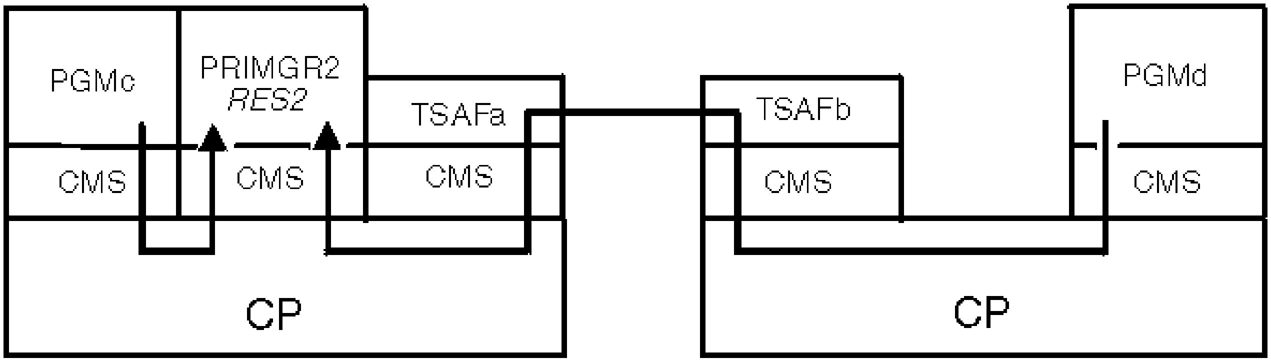


Figure 105. Sending and Receiving Data

Establishing an APPC Link

In the following scenario two z/VM systems, connected by a physical VTAM link, form a TSAF collection by establishing an APPC link. As Figure 106 on page 203 shows, this example involves:

- A z/VM system (System A) made up of:
  - A TSAF virtual machine, with the user ID TSAFa
  - An AVS virtual machine, with the user ID AVSa
  - A VTAM virtual machine, with the user ID VTAMa.
- A z/VM system (System B) made up of:
  - A TSAF virtual machine, with the user ID TSAFb
  - An AVS virtual machine, with the user ID AVSb
  - A VTAM virtual machine, with the user ID VTAMb.

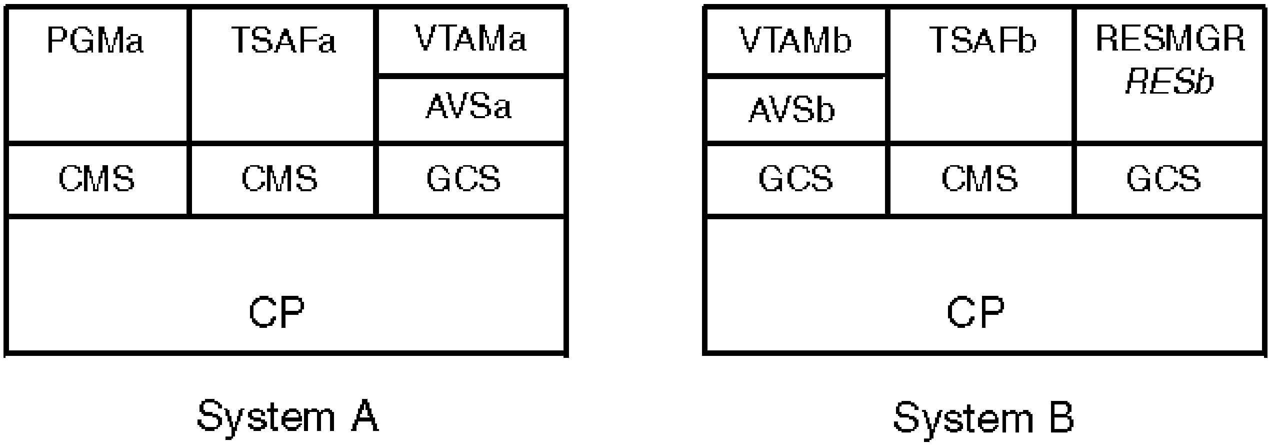


Figure 106. System A and System B

Step 1—Preparing System A

The following sections describe the steps that enable System A to add an APPC-type link to System B.

Defining the Dedicated Private Gateway and Session Limits

The AVSa virtual machine operator enters the following command to identify the private dedicated gateway on System A (`systema`).

```
agw activate gateway systema private userid tsafa
```

The AVSa virtual machine operator also enters an AGW CNOS command to specify the session limits (2) and the mode name (`vmint` in this example):

```
agw cnos systema systemb vmint 2 1 1
```

Note: The AVS virtual machine operator can add these commands to the AGWPROF GCS exec in the AVS virtual machine so that they take effect automatically each time the AVS virtual machine is logged on.

Adding an APPCPASS Statement

The system administrator adds an APPCPASS statement to the TSAFa virtual machine's CP directory entry. This statement identifies the TSAFa virtual machine's user ID (`tsafaonb`) and password (`apassonb`), which are valid on System B:

```
appcpass systema  systemb  tsafaonb apassonb
```

When the TSAFa virtual machine issues the connection request, CP on System B compares the user ID and password in the request, which is supplied by this APPCPASS statement, to the user ID and password in the CP directory entry created on System B for the TSAFa virtual machine (see "Creating a CP Directory Entry for the TSAFa Virtual Machine" on page 204).

Creating a CP Directory Entry for the TSAFb Virtual Machine

The system administrator adds the following statements to System A's CP directory. These statements contain the TSAFb virtual machine's user ID (`tsafbona`) and password (`bpassona`), which are valid on System A:

```
USER TSAFBONA BPASSONA
CONSOLE 00F
```

Creating a UCOMDIR NAMES File Entry

The TSAF virtual machine operator enters the following command to stop the TSAFa virtual machine:

```
stop tsaf
```

Next, the TSAF virtual machine operator creates an entry in the UCOMDIR NAMES file for the TSAFb virtual machine. This entry defines the symbolic destination name (`locatnb`) for the TSAFb virtual machine:

```
:nick.locatnb     :luname.systema systemb
                  :tpn.&tsaf
                  :modename.vmint
                  :security.pgm
```

Creating a $SERVER$ NAMES File

The TSAF virtual machine operator creates a $SERVER$ NAMES file entry, which contains the TSAFb virtual machine's user ID (`tsafbona`) that is valid on System A:

```
:nick.&tsaf       :list.tsafbona
```

Adding Commands to the PROFILE EXEC

The system administrator adds the following private server commands to the TSAFa virtual machine's PROFILE EXEC:

```
SET SERVER ON
SET COMDIR ON
SET COMDIR FILE USER UCOMDIR NAMES
```

After each of the preceding steps has been completed, the TSAFa virtual machine operator runs the PROFILE EXEC again. If it is not already in the PROFILE EXEC, the TSAFa virtual machine operator also enters RUNTSAF to restart the TSAFa virtual machine.

Step 2—Preparing System B

The following sections describe the steps that enable System B to add an APPC-type link to System A.

Defining the Dedicated Private Gateway and Session Limits

The AVSb virtual machine operator enters the following AVS commands to identify the private dedicated gateway and to define session limits:

```
agw activate gateway systemb private userid tsafb
agw cnos systemb systema vmint 2 1 1
```

Adding an APPCPASS Statement to the CP Directory

The system administrator adds the following APPCPASS directory statement to the TSAFb virtual machine's CP directory entry. This statement specifies the TSAFb virtual machine's user ID (`tsafbona`) and password (bpassona) that are valid on System A:

```
appcpass systemb  systema  tsafbona bpassona
```

Creating a CP Directory Entry for the TSAFa Virtual Machine

The system administrator add the following statements to System B's CP directory. These statements contain the user ID (`tsafaonb`) and password (apassonb) of the TSAFa virtual machine that are valid on System B:

```
USER TSAFAONB APASSONB
CONSOLE 00F
```

Creating a UCOMDIR NAMES File Entry

First, the TSAF virtual machine operator enters STOP TSAF to stop the TSAFb virtual machine.

Then, the TSAF virtual machine operator creates an entry in UCOMDIR NAMES file for the TSAFa virtual machine. This entry defines the symbolic destination name (`locatna`) for the TSAFa virtual machine on System A:

```
:nick.locatna      :luname.systemb systema
                   :tpn.&tsaf
                   :modename.vmint
                   :security.pgm
```

Creating a $SERVER$ NAMES File Entry

The TSAF virtual machine operator creates an entry in the $SERVER$ NAMES file for the TSAFa virtual machine. This file contains the TSAFa virtual machine's user ID (`tsafaonb`) that is valid on System B:

```
:nick.&tsaf        :list.tsafaonb
```

Adding Commands to the PROFILE EXEC

The system administrator adds the following private server commands to the TSAFb virtual machine's PROFILE EXEC:

```
SET SERVER ON
SET COMDIR ON
SET COMDIR FILE USER UCOMDIR NAMES
```

After taking the preceding steps, the TSAF virtual machine operator runs the PROFILE EXEC again. If it is not already in the PROFILE EXEC, the TSAF virtual machine operator also enters the RUNTSAF command to restart the TSAFb virtual machine.

Step 3—Requesting a Connection to the TSAFb Virtual Machine

The TSAFa virtual machine operator on System A enters an ADD LINK command with the symbolic destination name (`locatnb`) that was defined in the UCOMDIR NAMES file entry for the TSAFb virtual machine:

```
add link locatnb
```

As Figure 107 on page 206 shows, this causes the TSAFa virtual machine to request a connection to the TSAFb virtual machine. The connection request is routed to the AVSa and VTAMa virtual machines on System A and passes over a VTAM controlled communications link to System B. On System B, the request passes through the VTAMb and AVSb virtual machines. The request is then sent on to the TSAFb virtual machine.

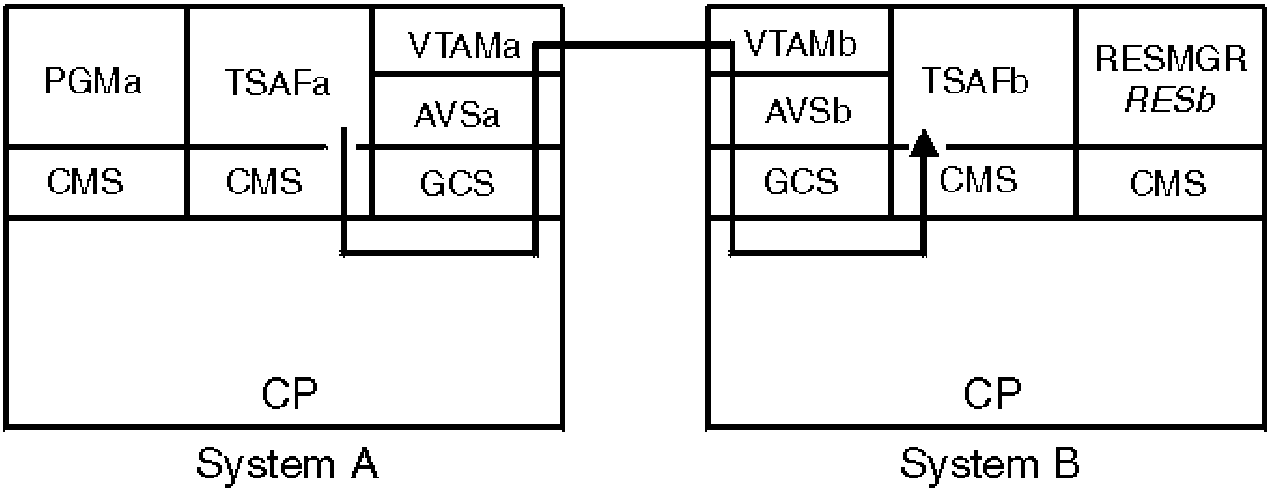


Figure 107. Requesting a Connection to the TSAFb Virtual Machine

Step 4—Requesting a Connection to the TSAFa Virtual Machine

The TSAFb virtual machine operator enters an ADD LINK command with the symbolic destination name (`locatna`) that was defined in the UCOMDIR NAMES file entry for the TSAFa virtual machine:

```
add link locatna
```

As Figure 108 on page 206 shows, this also causes the TSAFb virtual machine to request a connection to the TSAFa virtual machine. This connection request passes to the AVSb and VTAMb virtual machines. The request then passes over a VTAM controlled communications link to System A. On System A, the request passes through the VTAMa and AVSa virtual machines. The request is then sent on to the TSAFa virtual machine.

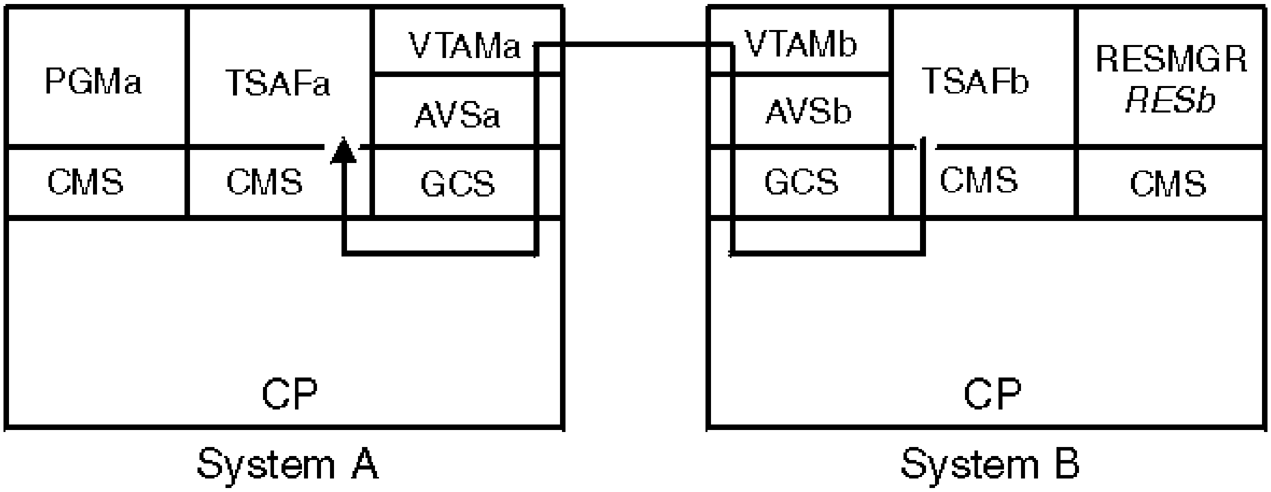


Figure 108. Requesting a Connection to the TSAFa Virtual Machine

Step 5—Creating the TSAF Collection

As Figure 109 on page 207 shows, an APPC link is established over the physical link controlled by VTAM when System A and System B accept the connection requests. The TSAF virtual machines on each system can communicate with each other and the systems form a TSAF collection.

The programs on each system can then exchange information. These programs do not know (or need to know) what type of links are used within the TSAF collection.

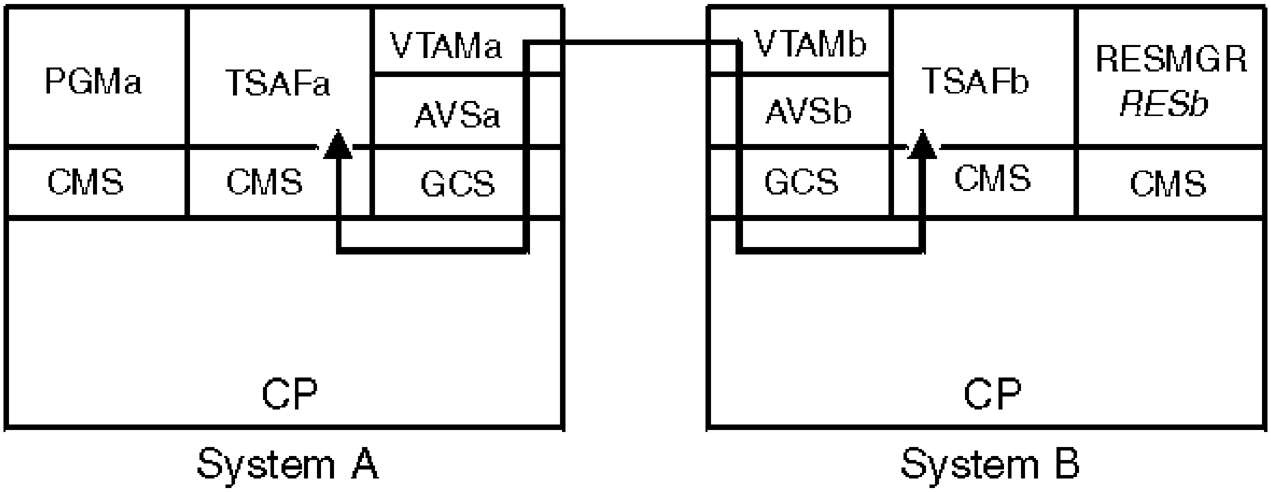


Figure 109. System A and System B Form a TSAF Collection

Because System A and System B have formed a TSAF collection, as Figure 110 on page 207 shows, the user program (PGMa) on System A can access the resource manager (RESb) on System B. These programs exchange information over the VTAM controlled communications as they would over a TSAF controlled link.

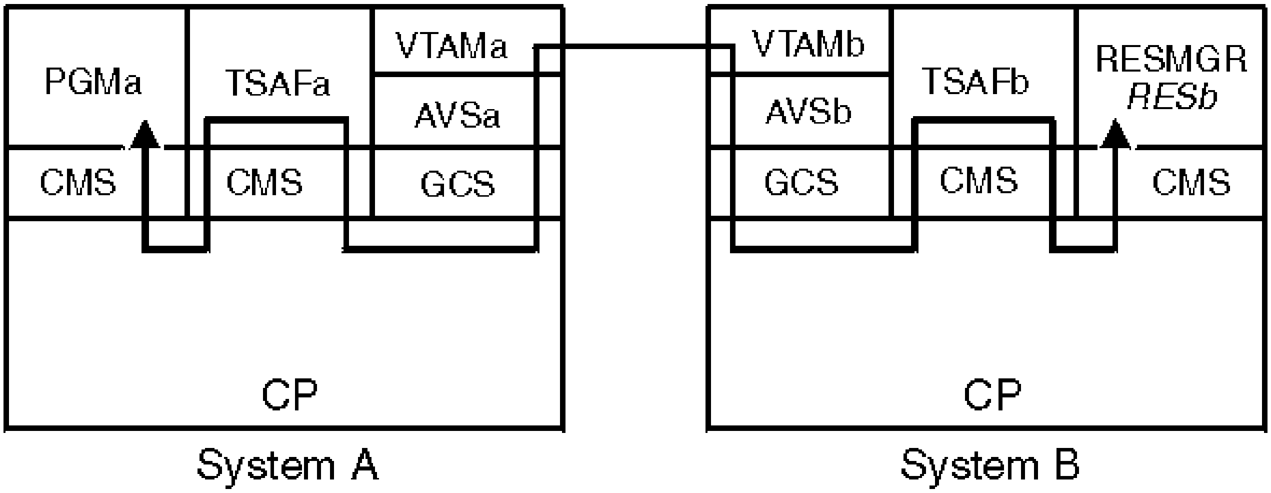


Figure 110. Accessing the Resource Manager on System B

Supported Links

The systems that make up the TSAF collection can be connected by any of the following links:

- TSAF-controlled links:
  - Channel-to-channel (CTC) links, including 3088 links
  - Binary Synchronous Communications (BSC) links
  - IEEE 802.3 Local Area Network (Ethernet or IEEE 802.3) subsystem on the IBM rack-mounted processors
  - IBM Token Ring Local Area Network subsystem on the IBM rack-mounted processors.
- VTAM-controlled links:
  - Links controlled by VTAM occur when logical APPC links and SNA sessions are established between the TSAF virtual machines.

In Figure 111 on page 208, assume each system has TSAF running. This is an example of a TSAF collection made up of eight z/VM systems. VMSYS3, VMSYS4, VMSYS5, and VMSYS6 are a local area network of rack-mounted systems. The systems are connected by the various types of links. Links labeled A can be CTC links (including 3088), BSC, or APPC links. Links labeled B are Local Area Network (LAN) links. Each LAN link has a Continuously Executing Transfer Interface (CETI) group.

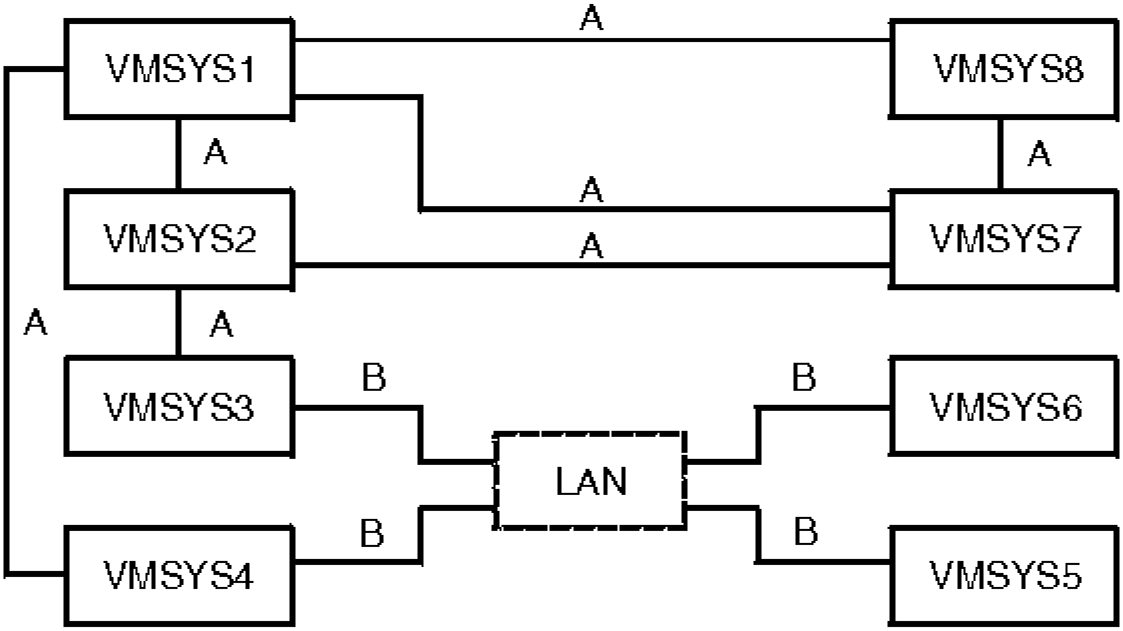


Figure 111. Sample TSAF Collection with Various Links

You should not define multiple active links between two TSAF virtual machines. If multiple active links are defined, the TSAF virtual machines may not choose the same link to communicate and unpredictable results may occur.

Setting Up Reliable Routes

When setting up a TSAF collection of more than two systems, try to assign links from each system to at least two other systems. This defines at least two fully or partially distinct physical routes through which the systems can communicate.

In Figure 112 on page 208, assume that each system has TSAF running. These four z/VM systems make up a TSAF collection. In this figure, VMSYS2 and VMSYS4 are not physically connected. However, the transaction programs on each system can communicate because a route to the other system exists through VMSYS1 or VMSYS3.

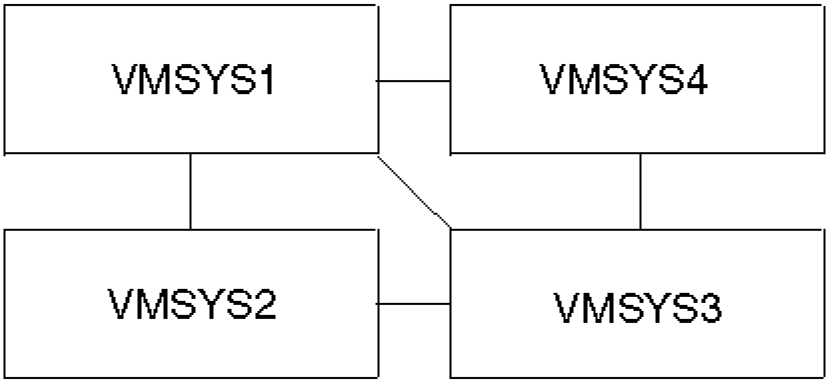


Figure 112. A Sample TSAF Collection

In Figure 113 on page 209, the systems, through the TSAF virtual machines, are connected by links VMSYS1 to VMSYS2, VMSYS2 to VMSYS3, and VMSYS3 to VMSYS4. These systems form a TSAF collection.

If the link from VMSYS2 to VMSYS3 failed, the TSAF collection would be partitioned. For example, users on VMSYS1 communicating with programs on VMSYS3 would be disconnected from those programs.

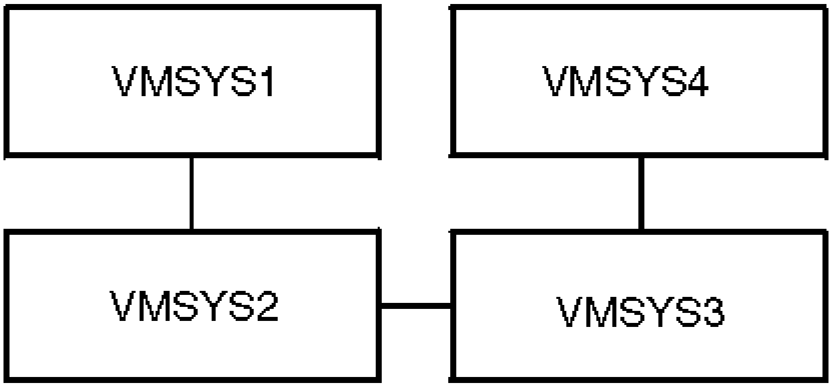


Figure 113. A TSAF Collection

If a link is added between systems VMSYS1 and VMSYS4 (see Figure 114 on page 209), the TSAF collection becomes more reliable. If the link from VMSYS2 to VMSYS3 failed, communication could continue on the path from VMSYS1 to VMSYS4 to VMSYS3.

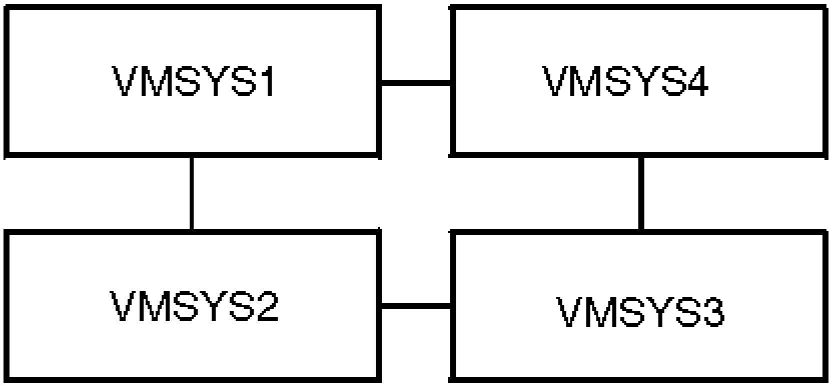


Figure 114. More Reliable TSAF Collection

TSAF Routing

When the TSAF collection configures itself, the TSAF virtual machines determine the various routes that connect each TSAF virtual machine to every other TSAF virtual machine. If more than one possible route exists between two TSAF virtual machines, TSAF chooses the route with a combination of the smallest number of intermediate systems and the fastest links.

The TSAF virtual machines reconfigure the TSAF collection if a route becomes unavailable because of an inoperative link, system, or TSAF virtual machine. The TSAF virtual machines also reconfigure the TSAF collection when a route becomes available or is added. After the TSAF collection is reconfigured, TSAF then selects a new route, if one exists.

How TSAF Dynamically Configures a TSAF Collection

A link is a physical or logical connection between two systems. When you start the TSAF virtual machine, you must give it the addresses or symbolic destination names of the links it needs to communicate with other systems. See “ADD LINK” on page 217 for information on how to add links.

The TSAF virtual machine uses this link information to dynamically configure the TSAF collection. TSAF sends out messages over each link that has been added. If an active TSAF virtual machine is connected at the other end of the link, the two TSAF virtual machines exchange information about the resources they manage. The TSAF virtual machines automatically configure the TSAF collection, based on the information they exchange. This procedure does not require an operator.

If a link is not operating or if there is not an active TSAF virtual machine on the other end of the link, TSAF does not use that link. However, TSAF periodically checks each link defined to it. TSAF dynamically reconfigures the TSAF collection when one of the following occurs:

- A link becomes operational or inoperative
- A TSAF virtual machine becomes active or inactive.

Route Failure

If APPC/VM data does not get to its target within a predefined time, the originating TSAF virtual machine tries to send the data again. It will repeat this process several times. If a problem occurs in the route the data has taken (for example, a hardware failure), the TSAF virtual machines will reconfigure the TSAF collection. After the TSAF collection and routing information are reconfigured, the originating TSAF virtual machine tries to send the data again. If the send function continues to fail (for example, because the node is no longer in the collection), the TSAF virtual machine severs the APPC/VM connection.

The TSAF virtual machines send out test messages at variable intervals over each link. If a test message indicates that a link is not operating, the TSAF virtual machines reconfigure the TSAF collection and routes.

When Two TSAF Collections Merge to Form One

Two or more TSAF collections can join to form a single collection if any of the following occurs:

- The system recovers from an abnormal end (abend)
- You bring up a TSAF virtual machine
- You make a link active.

The TSAF virtual machines joined in the collection exchange information about global resources and gateways. Only global resources and gateways are affected when TSAF collections merge, because the TSAF virtual machines do not know about local and private resources.

If this is the first time that the collections are merging and the collections were set up without consideration to unique resource and gateway names, the resulting collection may have one or more duplicate resource or gateway names. Because two systems in the same TSAF collection cannot each manage a global resource or gateway with the same name, TSAF guarantees unique names by awarding management responsibility to one of the two systems.

TSAF does not sever existing APPC/VM paths to a resource manager that loses management responsibility for the resource. However, new paths will go to the resource manager in the winning TSAF collection.

Two TSAF collections that contain a total of nine or more systems may try to merge. Because a TSAF collection may contain only eight systems, some systems may not be included in the merged TSAF collection. Because of the timing involved with TSAF collection communication, you cannot predetermine which systems will be excluded from the new TSAF collection. When this happens, the TSAF virtual console of the system, through which the ninth system was trying to join, gets this message:

```
ATSMD0513I Node nodeid cannot join, maximum collection size has
           been reached
```

In message ATS513I, the *nodeid* is the node ID of the system trying to join the collection. The TSAF virtual console of the losing z/VM system also receives a message.

Maintaining a Single TSAF Collection

When multiple TSAF virtual machines join to form a TSAF collection, efficiency is kept highest if the virtual machines are connected in a single TSAF collection. This efficiency is created by establishing multiple paths to the systems and enabling resources to be known throughout the TSAF collection. Also, when two multiple system TSAF collections try to merge, the process can be slow and involved as one of the collections will disband and join the other collection one system at a time.

This section provides two scenarios for keeping your TSAF collection as a single collection. This section also contains suggestions to simplify the tasks of adding a node to, or deleting a node from, a TSAF collection.

Note: TSAF collections that contain less than four systems may not see appreciable differences in efficiency, but the techniques that follow may be worth noting and following.

Node-By-Node Scenario

When started, every TSAF virtual machine that has an ATSLINKS FILE will attempt to join a TSAF collection. If two TSAF virtual machines are started at the same time, they will try to join with each other and form a single TSAF collection. The TSAF collection is considered joined when both systems issue the ‘Synchronization Normal’ message.

In the node-by-node scenario, you must start the TSAF virtual machine on two systems that have an ATSLINKS FILE defined. When these two systems have joined, each system will issue the ‘Synchronization Normal’ message. Before starting the TSAF virtual machine on another system, be sure that you receive this message.

For a TSAF collection that contains two or three systems, you may start the TSAF virtual machines in any order. You can start TSAF on two of the systems and then add the third system after the first two systems complete their connections.

If the TSAF collection contains four or more systems, you can improve the speed with which the TSAF collection forms by choosing the order in which you start the TSAF virtual machines. Select one of the systems to be used as the focal node that all the other nodes have in their ATSLINKS FILE. Start TSAF on this system and another system and wait for the systems to join a TSAF collection. You can then start the next TSAF virtual machine and bring that node into the TSAF collection. By maintaining a single TSAF collection, you can avoid interrupting communications among programs in the TSAF collection. Communications can be interrupted if multiple TSAF collections are created and then attempt to form a single TSAF collection.

To bring down the TSAF collection in this scenario, enter STOP TSAF on all the TSAF virtual machines.

To delete one system from the TSAF collection, enter STOP TSAF on the node you wish to delete. A node deleted message signifies that the system is no longer in the TSAF collection.

Link-by-Link Scenario

This scenario describes how to start all the TSAF virtual machines first, controlling the collection by selectively adding the links.

The ATSLINKS FILE contains all the last-known TSAF links for the node. When the TSAF virtual machine is restarted, it will automatically try to re-establish those links identified in the ATSLINKS FILE. Therefore, you need to erase this file, if it exists on the TSAF virtual machines, before you enter the RUNTSAF command. If an ATSLINKS FILE does not exist on your TSAF virtual machine, the node will come up, but it will not attempt to join to any other nodes.

After all TSAF virtual machines have started, pick a focal node that all the other nodes will join. Issue an ADD LINK command to link any other node to the focal node. Issue an ADD LINK command at this other node to complete the link. Wait for the completion of the connection, signaled by the ‘Synchronization Normal’ message, before adding the next node. Repeat this step for the remaining nodes in this TSAF collection.

To bring down the entire TSAF collection in this scenario, either delete all the links on every node then enter STOP TSAF, or enter STOP TSAF and then erase the ATSLINKS FILE at all the nodes. If the RUNTSAF command is issued from the PROFILE EXEC, you may choose to add an ERASE ATSLINKS FILE command just before the RUNTSAF command, so that any start-up will ensure that the ATSLINKS FILE is not present.

To delete one system from the TSAF collection in this scenario, either delete all the links on that node then enter STOP TSAF, or enter STOP TSAF and then erase the ATSLINKS FILE on that system. Optionally, you can delete the link to this system at all the other system.

Additional Considerations

When starting TSAF collections, try to keep only one TSAF collection. To do so, add one TSAF virtual machine to the TSAF collection by individually starting the TSAF virtual machine or selectively adding TSAF links. Following these guidelines should help in setting up and maintaining reliable single TSAF collections.

- If global resources and gateways are being identified or revoked while the TSAF collection is being activated, the TSAF collection will take longer to come together. It is better to have all global resources and gateways identified before or after all TSAF virtual machines have joined the single TSAF collection.
- Before deleting another TSAF virtual machine from the TSAF collection, wait until the remaining TSAF virtual machines acknowledge the loss of the first TSAF virtual machine. At this point, repeat the process for deleting the next TSAF virtual machine.
- Carefully consider the physical configuration of your TSAF collection when deleting systems. Avoid isolating the systems by deleting a link that provides the only path between two sub-collections. If this is done, the systems will partition to form two separate TSAF collections. When the deleted node is later restarted, the merging of the partitioned TSAF collections could take some time to complete.
- If you are deleting all TSAF virtual machines from a TSAF collection, you can remove them in any order.

Performance Considerations

This section describes performance considerations for using TSAF. For information about other z/VM performance considerations, see *z/VM: Performance*.

Communication Path Characteristics

Programs that use local paths perform faster than programs that use TSAF and remote APPC/VM paths. When user and requester programs are on the same system, only one APPC/VM path is needed. As Figure 95 on page 197 shows, the communication request does not go to the TSAF virtual machine.

When communicating with a remote program within the TSAF collection, the following are involved:

- Two APPC/VM paths
- Two or more TSAF virtual machines
- One or more physical connections.

TSAF performance also depends on the speed of the communication line routing the path. To improve performance of the remote paths, use the class A SET command with the SHARE or QUICKDSP parameters. See the *z/VM: CP Commands and Utilities Reference* for more details on the SET command.

Line Performance Characteristics

TSAF functions that affect all the systems in a TSAF collection include:

- Identifying a new global resource or gateway in the TSAF collection
- Revoking a global resource or gateway from the TSAF collection
- Joining another TSAF collection.

How fast any of these functions complete is directly related to the speed of the slowest line that TSAF is using in the TSAF collection. A CTC link is faster than a LAN link, which is faster than a BSC link. So, using a BSC line can significantly slow down TSAF functions that affect all the systems in a TSAF collection. The speed of an APPC link depends on the type of physical link that is controlled by VTAM. On average, the speed on an APPC link is comparable to a BSC link. TSAF always sends user data on the fastest route in the collection. Therefore, slow lines in the route would only slow down the transmission of user data if these lines had to be used for routing.

You should also consider the transmission error rate associated with each line in the TSAF collection. A BSC line with a fixed-line speed is less reliable and not as available if the number of transmission errors increases. TSAF assumes the error rate to be less than 1-bit in every 500,000 bits sent.

When the error rate is high, TSAF tends to break the communication path and mark the line *down*. The performance of the entire TSAF collection can degrade when TSAF must continually change the status of the line.

APPC Link and VTAM Performance Considerations

If your TSAF collection includes systems that are connected by physical VTAM links, you should establish SNA sessions and define explicit routing between the VTAM virtual machines on each system. You establish SNA sessions when you enter the AGW CNOS command (see "Purpose" on page 274). Use the VTAM PATH definition statement to define explicit routes between the VTAM virtual machines; see *ACF/VTAM Installation and Resource Definition* for more information.

By establishing explicit VTAM session and routes, you enable VTAM to perform the routing between intermediate processors in the TSAF collection when you establish APPC links to those systems. The TSAF virtual machines at the intermediate processors will not be involved in routing these conversations, which reduces the data traffic through the TSAF virtual machines. In this way, explicit VTAM sessions and routes can enhance the over-all performance of the TSAF collection. When you establish explicit VTAM sessions and routes between all systems in the TSAF collection, you create a logically fully-connected TSAF collection.

For example, Figure 115 on page 213 shows a group of three z/VM systems, each with TSAF and VTAM running in virtual machines. VMSYS2 is connected to the other remote systems by physical links, labeled A, that are controlled by VTAM. To form a TSAF collection, each system establishes an APPC link to each of the other systems. Each system also defines VTAM sessions and explicit routing to each of the other systems in the group.

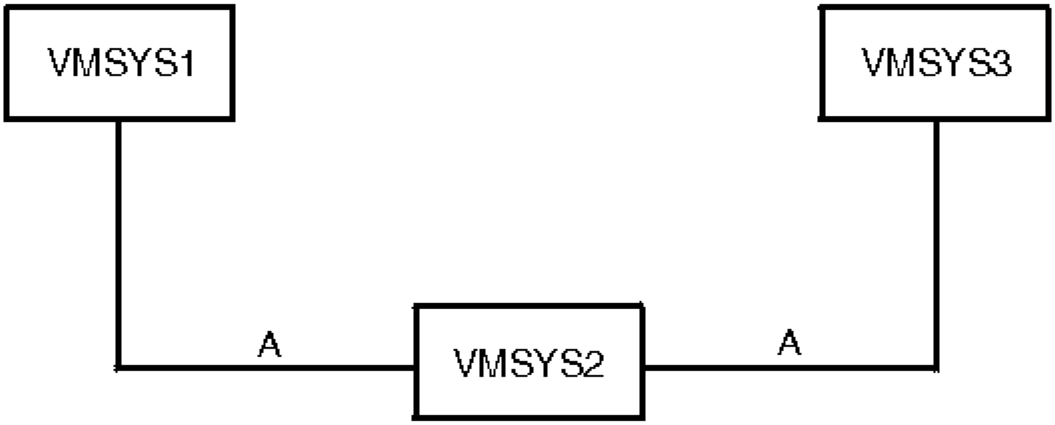


Figure 115. Remote Systems Connected by Two Physical VTAM Controlled Links

As Figure 116 on page 214 shows, the TSAF collection becomes fully-connected because of the logical link between VMSYS1 and VMSYS3. VMSYS2 is the physical intermediate system between VMSYS1 and VMSYS3. Because VTAM sessions and explicit routing are defined between each system, data sent from VMSYS1 to VMSYS3 is routed through the VTAM virtual machine on VMSYS2, bypassing the TSAF virtual machine on VMSYS2.

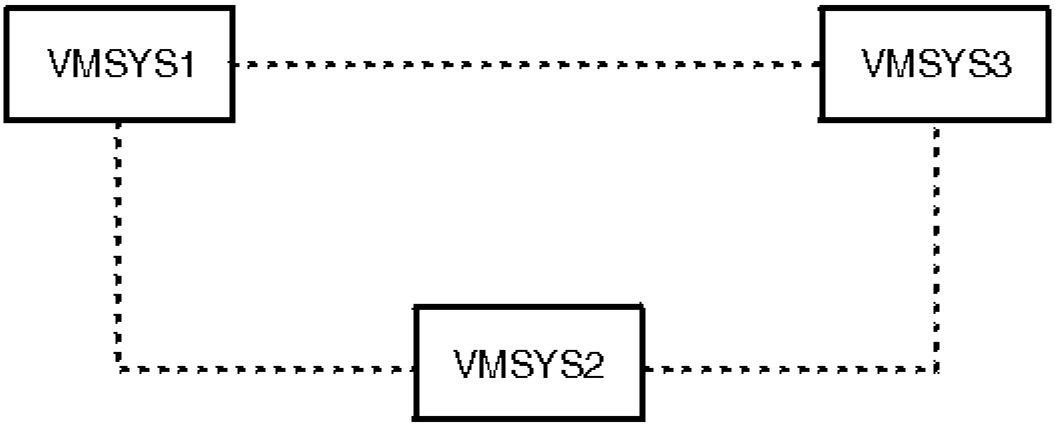


Figure 116. Logically Fully-Connected TSAF Collection

Chapter 14. Operating the TSAF Virtual Machine

This chapter focuses on operating the TSAF (Transport Services Access Facility) virtual machine within the z/VM environment. It outlines the primary commands used to manage TSAF, including how to add and delete communication links (ADD LINK, DELETE LINK), retrieve configuration and status information (QUERY), start and stop the TSAF virtual machine (RUNTSAF, STOP TSAF), and manage external tracing (SET ETRACE). Each command is explained with its purpose, syntax, operands, expected behavior, and associated messages or return codes. The chapter emphasizes the dynamic and self-configuring nature of TSAF, which requires minimal manual intervention. It also provides guidance on using the z/VM HELP facility for command assistance and troubleshooting. Overall, the chapter serves as a comprehensive operational guide for administrators who manage TSAF in a z/VM environment.

Overview of TSAF Commands

Because the TSAF virtual machine dynamically configures itself, you, as the operator, only need to enter a few commands from the TSAF console to operate the TSAF virtual machine.

Command	Function	Location
ADD LINK	Adds a link to the TSAF virtual machine.	"ADD LINK" on page 217
DELETE LINK	Deletes a link from the TSAF virtual machine.	"DELETE LINK" on page 220
QUERY	Displays information about the TSAF configuration.	"QUERY" on page 222
RUNTSAF	Starts the TSAF virtual machine.	"RUNTSAF" on page 227
SET ETRACE	Sets external tracing on or off.	"SET ETRACE" on page 230
STOP TSAF	Stops the TSAF virtual machine.	"STOP TSAF" on page 233

For information about how to read the syntax of the TSAF commands, see "Syntax, Message, and Response Conventions" on page xxii.

Using Online HELP for TSAF Commands

You can receive online information about TSAF commands by using the z/VM HELP Facility. For example, to display a menu of the TSAF commands, enter:

```
help tsaf menu
```

To display information about a specific TSAF command (QUERY in this example), enter:

```
help tsaf query
```

You can also display information about a message by entering one of the following commands:

```
help msgid or help msg msgid
```

For example, to display information about message ATSLLM701E, you can enter one of the following commands:

```
help atsllm701e or help ats701e or help msg ats701e
```

For more information about using the HELP Facility, see the *z/VM: CMS User's Guide*. To display the main HELP Task Menu, enter:

```
help
```

For more information about the HELP command, see the *z/VM: CMS Commands and Utilities Reference* or enter:

```
help cms help
```

ADD LINK

```
►► ADD ── LINK ── linkunit ─►◄
```

Purpose

Use the ADD LINK command to identify a communication link to TSAF when the TSAF virtual machine is running. Possible link types are: ELAN, TLAN, CTCA, 3088, BSC, and APPC.

You only need to add a link once to the TSAF virtual machine. After that, if you do not delete the link, the link will automatically be added when the TSAF virtual machine starts. The TSAF virtual machine can use the added link to reconfigure or join the TSAF collection.

Operands

linkunit

is one of the following values, which identifies the link:

- A symbolic destination name that has been defined in the CMS communications directory for TSAF. If you are assigning a symbolic destination name for an APPC link, do not use a symbolic destination name that is also a valid virtual device address, such as 123 or DAD.
- A virtual device address you want to use as a link. You may specify a virtual device address of up to four digits and omit leading zeros. (The virtual device address 0000 is not valid.)
- A virtual device address associated with a rack-mounted LAN subsystem. This must be the first address of a sequence of four addresses associated with a rack-mounted LAN subsystem's Continuously Executing Transfer Interface (CETI) group.

What Happens When You Enter this Command

After you enter ADD LINK, TSAF checks if the *linkunit* is defined as a symbolic destination name in the CMS communications directory file. If it is not a symbolic destination name, TSAF processes the *linkunit* as a virtual device address and attempts to initialize the device at that virtual address. TSAF also adds the link information to the ATSLINKS FILE on the TSAF virtual machine file mode A.

For example, to add an APPC-type link to the TSAF virtual machine you would enter:

```
add link vtamlink
```

If the symbolic destination name `vtamlink` is defined in the CMS communications directory for the TSAF virtual machine, you receive the message:

```
ATSLLM724I Link vtamlink added
```

For bisynchronous links, you may also get message ATS795I with message ATS724I. If you receive a few 795I messages, communication could be slow across the links.

However, if you receive 10 or more 795I messages, the indicated link may be inoperative or the system on the other side of the link may be down. Determine if the other system is running. If the system is up, delete the link and try to add it again. If this does not work, stop TSAF (enter STOP TSAF) and restart it (enter RUNTSAF). If the ADD LINK command for the link fails again, the device may have been set up incorrectly.

Messages and Return Codes

ATSCOP004E
Parameter *parameter* is not valid

ATSCOP005E
A required parameter is missing

ATSNHR602E
Incompatible release or service level detected on link *linkunit*

ATSNHR603E
Duplicate node *nodeid* detected on link *linkunit*

ATSLLM700E
Link-Definition table overflow, unable to add the new link *linkunit*

ATSLLM701E
Driver rejected the new link *linkunit*

ATSLLM702E
Link unit address *vdev* is not valid

ATSLLM703E
Link *linkunit* is not a supported link type

ATSL1A710E
Unable to allocate control block for link *linkunit*

ATSL1A711E
Unable to allocate I/O buffer for link *linkunit*

ATSLLM712E
Link unit address *vdev* is a duplicate

ATSLLM714E
Link symbolic destination name *symdest* is a duplicate

ATSLLM715E
Failed to add the definition of link *linkunit* to ATSLINKS FILE A1. Return code from FSWRITE was *nnnn*.

ATSLLM724I
Link *linkunit* added

ATSLLM725E
vdev is not a valid device address for link type *linktype*

ATSLLM726E
Link-Definition table expansion failed, unable to add logical link for LAN link *vdev*

ATSL0Y727E
Unit *vdev* is not a valid device for link type *linktype*

ATSL4Y727E
Unit *vdev* is not a valid device for link type *linktype*

ATSL0A730E
Unit *linkunit* DIAGNOSE X'98' error, return code *rc*

ATSL4A730E
Unit *linkunit* DIAGNOSE X'98' error, return code *rc*

ATSL5I738E
Unable to identify resource *resid* to CMS. HNDIUCV SET function failed. R15 = *xxxx*

ATSL5O740E
CMSIUCV ACCEPT function failed on link *symdest*. R15 = *xxxx*

ATSL5W742E
CMSIUCV CONNECT function failed on link *symdest*. R15 = *xx*

ATSLLM744E
Invalid transaction program name *tpn1* specified in the CMS communications directory for link *symdest*. The transaction program name must be *tpn2*.

ATSL5A745E
The gateway name *gatelu* and target LU name *targetlu* combination for link *symdest* is a duplicate

ATSL3W795I
Retry limit exceeded on unit *vdev*

ATSL1A799I
Unit *vdev* is not operational

For information on these messages, see *z/VM: Other Components Messages and Codes*.

DELETE LINK

```
►►— DELETE — LINK — linkunit —►◄
```

Purpose

Use the DELETE LINK command to remove a communication link from the TSAF table of communication links when the TSAF virtual machine is running.

Operands

linkunit

is one of the following values, which identifies the link:

- A symbolic destination name defined in the CMS communications directory for TSAF. If you specify a symbolic destination name for an APPC link, do not use a symbolic destination name that is also a valid virtual device address, such as 123 or DAD.
- A virtual device address you want to use as a link. You may specify a virtual device address of up to four-digits and omit leading zeros.
- A virtual device address associated with a rack-mounted LAN subsystem. This must be the first address of a sequence of four addresses associated with a rack-mounted LAN subsystem's Continuously Executing Transfer Interface (CETI) group.

What Happens When You Enter this Command

TSAF purges any link information related to the link unit that you specified from the TSAF table of communication links. TSAF also comments the link information out of ATSLINKS FILE located on the TSAF virtual machine's file mode A. The next time you start TSAF, the link will be deleted from the ATSLINKS FILE.

For example, you enter the following command to delete an APPC link with the symbolic destination name APPCLINK:

```
delete link appclink
```

The TSAF virtual machine reconfigures the collection when the command completes and you get this message:

```
ATSLLM713I Link appclink deleted
```

Messages and Return Codes

ATSCOP004E

Parameter *parameter* is not valid

ATSCOP005E

A required parameter is missing

ATSLLM702E

Link unit address *vdev* is not valid

ATSLLM713I

Link *linkunit* deleted

ATSLLM716E

Driver rejected the request to delete link *linkunit*

ATSLLM720E
Failed to delete the definition of link *linkunit* from ATSLINKS FILE A1. Return code from FSREAD was *nnnn*.

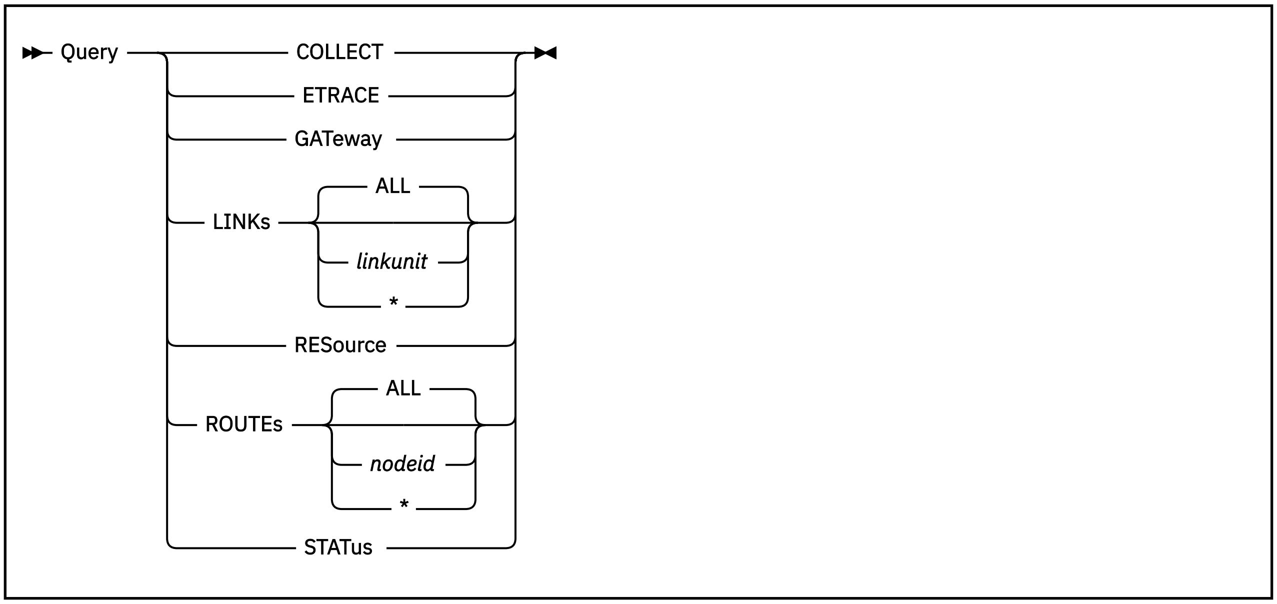
ATSLLM721E
Failed to delete the definition of link *linkunit* from ATSLINKS FILE A1. Return code from FSWRITE was *nnnn*.

ATSLLM723E
Link *linkunit* not found

ATSL3W795I
Retry limit exceeded on unit *vdev*

For information on these messages, see *z/VM: Other Components Messages and Codes*.

QUERY



Purpose

Use the QUERY command to get information about the TSAF configuration when the TSAF virtual machine is running.

Operands

COLLECT
displays the names of the systems currently in the TSAF collection.

ETRACE
displays the current setting of the external tracing.

GATeway
displays the current list of gateways defined in the TSAF collection.

LINKs
displays information about the links that TSAF currently has in the Link-Definition table. Possible link types are: ELAN, TLAN, CTCA, 3088, VTAM, BSC, and APPC.

ALL
displays either the symbolic destination name or virtual device address, the link type, and operational status for all of the links and CETI groups that TSAF currently has in its definition table. ALL is the default.

linkunit
is one of the following values, which identifies the link:

- A symbolic destination name that has been defined in the CMS communications directory for TSAF. If you are specifying a symbolic destination name for an APPC link, do not use a symbolic destination name that is also a valid virtual device address, such as 123 or DAD.
- A virtual device number for a specific link. You may specify up to four digits in the virtual device address and omit leading zeros.
- A virtual device number associated with a rack-mounted LAN subsystem. This must be the first address of a sequence of four addresses associated with a rack-mounted LAN subsystem's Continuously Executing Transfer Interface (CETI) group.

*****
: displays either the symbolic destination name or virtual device address, the link type, and operational status for all of the links and CETI groups that TSAF currently has in its definition table.

RESource
displays the current list of global resources in the TSAF collection.

ROUTEs
displays the route information for your node.

ALL
displays all of the route information currently known at your system or node; ALL is the default.

nodeid
displays the route information to the specified node.

*****
displays all of the route information currently known at your system or node.

STATus
displays the current information about the correlation of other TSAF virtual machines in the collection.

What Happens When You Enter this Command

Responses vary according to the operands you specify on the QUERY command.

QUERY COLLECT: If you enter QUERY COLLECT, TSAF displays the node IDs of the systems in the TSAF collection:

```
node01  [node02  node03  node04  node05  node6  node7  node08]
```

QUERY ETRACE: If you enter QUERY ETRACE, TSAF displays one of the following responses, which indicates the external trace option setting.

```
ETRACE OFF
ETRACE ON
ETRACE ON LINK linkunit
ETRACE ON USERID userid
ETRACE ON LINK linkunit USERID userid
```

QUERY GATEWAY: If you enter QUERY GATEWAY, TSAF displays the following response for each gateway known in the TSAF collection:

```
gatewayid at node nodeid
```

If no gateways are defined, you receive the following response:

```
No gateways identified
```

QUERY LINKS: When you enter QUERY LINKS, TSAF displays the status of the link in the following format:

```
Link: linkunit
     Type:         type    [LAN Address:    addr]
     Status:       status  Delay:          dtime
     Neighbor: nodeid      Neighbor State: state
```

linkunit
The link identifier.

type
One of the following link types: ELAN, TLAN, CTCA, 3088, VTAM, BSC, and APPC.

addr
The LAN address, if this is a LAN-based link.

status

One of the following values:

Up

The link is active.

Down

The link is not active.

Up/Busy

The link is up but is too busy to respond to link line tests; the link may be overloaded and the TSAF collection may need to be more fully connected.

Reset

TSAF is in the process of adding the link.

Unknown

The link's status is changing between up and down.

dtime

The round trip time of the link measured by TSAF in milliseconds.

nodeid

The TSAF node ID of the remote node (neighbor) connected by this link.

state

The neighbor state is the state of the link from the collection point of view. Possible values for neighbor state are: unknown, DISABLED, UNCERTAIN, LOSER, WINNER, AGENT, CLIENT, ESTABLISHED, NEW, and WAITING.

When the link is first added, the state will be unknown. When two neighboring nodes are in a collection together, the state should be ESTABLISHED. When two neighboring nodes are joining, LOSER, WINNER, AGENT, and CLIENT are possible states. When neighbors leave or enter the collection, DISABLED, UNCERTAIN, NEW, and WAITING are possible states.

For example, if you enter `query link 04a0`, where 04A0 is the virtual device address of a non-LAN subsystem, you would get the following response:

```
Link: 04A0
     Type:         CTCA
     Status:         up    Delay:             5
     Neighbor: KGN002      Neighbor State: ESTABLISHED
```

If the link is a LAN subsystem, TSAF displays the active logical links on the LAN by specifying the LAN address. For example, if you enter `query link 0400`, where 400 is the virtual device address of a Token Ring Local Area Network subsystem, you would get the following response:

```
Link: 0400
     Type:         TLAN
     Status:         up    Delay:             5
     Neighbor:             Neighbor State: unknown

Link: 0400
     Type:         TLAN    LAN Address:    100005A04 0129
     Status:         up    Delay:             5
     Neighbor: KGN003      Neighbor State: ESTABLISHED
```

When you enter QUERY LINKS ALL in a TSAF collection with two CTC links at addresses 03A0 and 04A0, one BSC link at address 0550, and an APPC link with the symbolic destination name VTAMLINK, you would get the following group of messages:

```
Link: 03A0
     Type:         CTCA
     Status:         up    Delay:             5
     Neighbor: KGN002      Neighbor State: ESTABLISHED

Link: 04A0
     Type:         CTCA
     Status:       down    Delay:             0
     Neighbor:             Neighbor State: unknown
```

```
Link: 0550
     Type:         BSC
     Status:         up    Delay:             5
     Neighbor: KGN003      Neighbor State: ESTABLISHED

Link: VTAMLINK
     Type:         APPC
     Status:         up    Delay:             5
     Neighbor: KGN002      Neighbor State: ESTABLISHED
```

Note: When you enter QUERY LINKS ALL, TSAF displays the status of all the links that have been added to the TSAF virtual machine. It does not query the status of an individual link with the symbolic destination name ALL.

QUERY RESOURCE: If you enter QUERY RESOURCE, TSAF displays the following response for each global resource known in the TSAF collection:

```
resourceid at node nodeid
```

If the TSAF collection does not contain any global resources, you get the following response:

```
No global resources identified
```

QUERY ROUTES: If you enter QUERY ROUTES, TSAF displays one of the following:

- The node ID and symbolic destination name of the link
- The node ID and virtual device address of the link
- The first address of the CETI group and the logical link information, if it is a LAN link.

For each non-LAN system in the TSAF collection, you would get a response in the following format:

```
Node nodeid via link linkunit
```

For each LAN system in the TSAF collection, you would get a response in the following format:

```
Node nodeid via logical link to xxxxxxxx xxxx on link vdev
```

If no routes are defined in the TSAF collection, you get the following response:

```
No routes exist
```

QUERY STATUS: When you enter the QUERY STATUS command, you receive a set of messages in the following format:

```
Local node id:                     NODEA
Current collection time:           00001026
Collection ID:                     NODEA
Last synchronous update time:      00000008
Next scheduled sync time:          000E1000
Sync period:                           7200 seconds
Transmission Delay:                    0020
Maximum clock deviation:               0000
Worst extra clock deviation:       00000001
Atomic broadcast duration:         00000101
Worst case diffusion hop count:          00
Current checksum:                  040731A1
Number of Nodes in collection:           01
Highest bit in use for signatures:       01
Current Node map:                  80000000
1)Processor node: NODEA     CPU ID: FF10000230900000
2)Processor node:           CPU ID: 0000000000000000
3)Processor node:           CPU ID: 0000000000000000
4)Processor node:           CPU ID: 0000000000000000
5)Processor node:           CPU ID: 0000000000000000
6)Processor node:           CPU ID: 0000000000000000
7)Processor node:           CPU ID: 0000000000000000
8)Processor node:           CPU ID: 0000000000000000
```

```
Collection bitmap: 80000000  Base signature: D5D6C4C5C1404040
Latest Manual adjustment to duration: 00000000
```

The response shows a detailed view of the TSAF virtual machine's operating state. If issued on each TSAF virtual machine at the same time, the QUERY STATUS response should be similar for each TSAF virtual machine in the TSAF collection.

The local node ID should match the node name of the local node corresponding to the processor identification in the SYSTEM NETID file. The collection ID should match one of the processor nodes that are displayed. The atomic broadcast duration estimates the delay before a global resource or gateway is known to all nodes in the TSAF collection. Other fields can be used to establish consistency between the data shown and the output from QUERY STATUS at other TSAF virtual machines in the collection.

Messages and Return Codes

ATSCOP004E
Parameter *parameter* is not valid

ATSCOP005E
A required parameter is missing

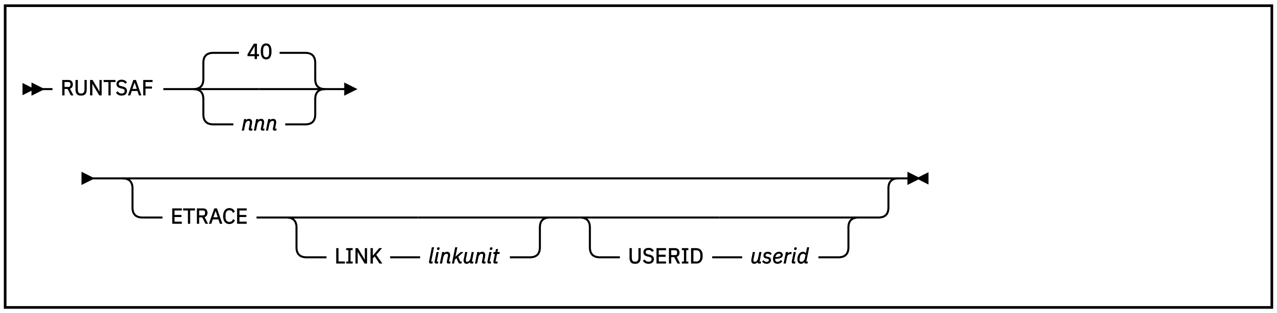
ATSLLM702E
Link unit address *vdev* is not valid

ATSLLM722I
No links are defined

ATSLLM723E
Link *linkunit* not found

For more information on these messages, see *z/VM: Other Components Messages and Codes*.

RUNTSAF



Purpose

Use the RUNTSAF command to start the TSAF virtual machine. Because TSAF runs as a CMS application, you must be in the CMS environment to start TSAF.

Operands

40
reserves 40 1KB blocks of virtual storage for the TSAF internal trace table; this is the default.

nnn
is the number of 1KB blocks of virtual storage for the TSAF internal trace table. Valid numbers are from 1 to 999; TSAF rounds up to the next 4K-byte boundary.

ETRACE
sets external tracing on. This causes TSAF to issue the CMS command ETRACE with the DMSTRACE operand and to write certain internal TSAF trace records externally to the TRSOURCE file when the TRSOURCE command was previously issued on the system. This is the only way to get an external trace during TSAF initialization. For information about ETRACE, see the *z/VM: CMS Callable Services Reference*.

If you do not specify this option, external tracing is initially off. When external tracing is off, TSAF writes trace records only to TSAF virtual storage.

LINK
specifies a link on which data tracing occurs when the TSAF virtual machine starts.

linkunit
is one of the following:

- A symbolic destination name defined in the CMS communications directory for TSAF. If you are assigning a symbolic destination name for a VTAM link, do not use a name that is also a valid virtual device address, such as 123 or DAD.
- A virtual device address you want to use as a link — specify a virtual device address of up to four digits. You can omit leading zeros.
- A virtual device address associated with a rack-mounted LAN subsystem. This must be the first address of a sequence of four addresses associated with a rack-mounted LAN subsystem's Continuously Executing Transfer Interface (CETI) group.

USERID
specifies the user ID of a virtual machine for which APPC/VM data transfers are traced when the TSAF virtual machine starts.

userid
is the user ID of a virtual machine that initiated an APPC/VM connection through the TSAF virtual machine.

What Happens When You Enter this Command

After you enter RUNTSAF, the TSAF virtual machine gets necessary parameters, such as the local node ID, using the CMS IDENTIFY command. If links have been previously defined, then the TSAF virtual machine joins the TSAF collection. It does this by exchanging data with other TSAF virtual machines over the links.

After the TSAF virtual machine has successfully started all of its permanent tasks, you may get a group of messages, including this message:

```
ATSCST001I Initialization is complete.  The service level is ssss.
```

When RUNTSAF is processed successfully, you do not receive the CMS ready message (Ready;) because you are now in TSAF.

When you enter RUNTSAF with the ETRACE LINK operand, you receive message ATS024W. This happens because TSAF's Link Definition table is not yet built when the TSAF virtual machine is initialized. The operand and link information you specified on the command are saved. When the table is built and the specified link is added to the TSAF virtual machine, data tracing starts for that link.

Similarly, when you enter RUNTSAF with the ETRACE USERID operand, you receive message ATS025W. This occurs because TSAF has not accepted any connections. The information you specified on the command are saved. When the table is built and some activity occurs for the specified user ID, data tracing starts for that user ID.

If you specify the LINK and USERID operands at the same time, you receive messages ATS024W and ATS025W.

Messages and Return Codes

ATSCST001I
Initialization is complete. The service level is *ssss*.

ATSCTL002T
Parameter *parameter* is a duplicate or is not valid

ATSCAC006I
TSAF link statistics and session accounting records will be generated

ATSCTL013I
Trace area size is *nnn*K

ATSCOP024W
Link *linkunit* has not been added

ATSCOP025W
User *userid* has no APPC/VM connection to TSAF

ATSMJK513I
Attempting JOIN with node *nodeid* as the agent

ATSMRZ518I
RESET: collection now has size 1

ATSMYC520I
Synchronization is now NORMAL

ATSMYC521I
Collection is roughly synchronized

ATSLMN707I
Link *linkunit* came up

ATSL3Z795I
Retry limit exceeded on unit *vdev*

If you specified the ETRACE operand on RUNTSAF, you may also receive some of the following CMS messages:

DMSABE1329I
Trace records were lost RC=*xx*

DMSDIE1329I
Trace records were lost RC=*xx*

DMSETC1330E
TRSOURCE must be enabled before calling ETRACE

DMSTRE1331E
The DMSTRACE facility (Monitor Call Class 10) is not enabled.

DMSTRE1332E
Invalid value specified for the ID parameter

DMSTRE1332E
Invalid value specified for the *parm* parameter. It was not in the range of *value1* to *value2*. RC=24

DMSETC1333E
TRSOURCE is disabled. Buffer not written. RC=40

DMSETC1333E
TRSOURCE is in EVENT mode. Buffer not written. RC=40

DMSETC1333E
TRSOURCE is disabled. Record not added to buffer.

DMSITP1333E
TRSOURCE is in EVENT mode. Record not added to buffer.

DMSITP1333E
ETRACE is disabled. Record not added to buffer.

DMSITP1333S
I/O or severe error. Buffer not written.

DMSITP1334E
Buffer not initialized. Reissue ETRACE.

DMSITP1335E
Incorrect Monitor Call Class 10 Code.

DMSITP1336E
This function needs the CP Diagnose X'E0' command.

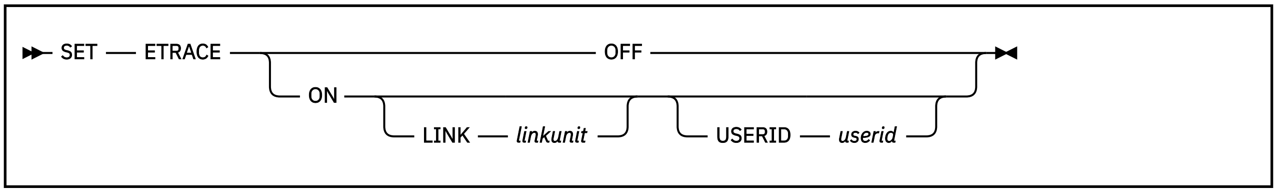
DMSITP1336E
This function needs the CP Diagnose X'EC' command. RC=40

DMSITP1337E
Insufficient storage to set up buffering. RC=40

DMSITP1338E
ETRACE set [ON|OFF] for DMSTRACE. RC=40

For information on these messages, see *z/VM: CMS and REXX/VM Messages and Codes* and *z/VM: Other Components Messages and Codes*.

SET ETRACE



Purpose

Use the SET ETRACE command to enable or disable external tracing.

Operands

OFF
causes TSAF to issue the CMS ETRACE command with the DMSTRACE operand, and write TSAF trace records only to TSAF virtual storage. No external tracing is done. OFF is the initial setting until you specify the ETRACE option when you entered RUNTSAF.

ON
causes TSAF to issue the CMS ETRACE command with the DMSTRACE operand, and write certain internal TSAF trace records externally to the TRSOURCE file when the TRSOURCE command was previously issued on the system.

Before using SET ETRACE and starting TSAF, you should enter the CP TRSOURCE ID command with the BLOCK parameter to improve tracing performance. See *z/VM: CP Commands and Utilities Reference* for information about TRSOURCE.

LINK
specifies a link on which full data tracing occurs. TSAF issues the CMS ETRACE command with the END operand, and all data transferred over this link to the TSAF virtual machine is traced.

linkunit
is one of the following values, which identifies the link:

- A symbolic destination name defined in the CMS communications directory for TSAF. If you are assigning a symbolic destination name for a VTAM link, do not use a name that is also a valid virtual device address, such as 123 or DAD.
- A virtual device address you want to use as a link— specify a virtual device address of up to four digits. You can omit leading zeros.
- A virtual device address associated with a rack-mounted LAN subsystem. This must be the first address of a sequence of four addresses associated with a rack-mounted LAN subsystem's Continuously Executing Transfer Interface (CETI) group that you no longer want to use as a link.

USERID
specifies the user ID of a virtual machine for which APPC/VM data transfers are traced. TSAF issues the CMS ETRACE command with the DMSTRACE operand and all data transferred through the TSAF virtual machine for the specified user ID will be traced.

userid
is the user ID of a virtual machine that initiated a connection through the TSAF virtual machine.

What Happens When You Enter this Command

If you enter SET ETRACE ON, TSAF starts to write trace records to a trace file and you get the following TSAF message:

```
ATSCOP010I  External trace started
```

If you enter SET ETRACE OFF, TSAF stops external tracing and you get the following TSAF message:

```
ATSCOP011I  External trace ended
```

What Happens When You Invoke SET ETRACE LINK: When you enter SET ETRACE with the LINK operand, all data transferred over the specified link to the TSAF virtual machine is traced. You can change the link that is traced by reissuing the SET ETRACE LINK command and specifying another *linkunit*. Full buffer tracing of data remains in effect until you enter the SET ETRACE OFF command.

For example, to trace data transferred to the TSAF virtual machine over a channel-to-channel adapter (CTCA) link that has the virtual device address 04A0, enter:

```
set etrace on link 04a0
```

To trace data transfers over a binary synchronous communication (BSC) link with the virtual device address 0400, enter:

```
set etrace on link 0400
```

Data tracing for the CTCA link (04A0) stops and the information is stored in a trace file. Tracing for the BSC link (0400) continues until you specify another *linkunit* on the LINK operand or enter SET ETRACE OFF.

What Happens When You Invoke SET ETRACE USERID: When you enter SET ETRACE with the USERID operand, all data transferred through the TSAF virtual machine for the specified *userid* is traced. Only one user ID can be traced at a time. However, you may change the user ID that is being traced by entering the SET ETRACE USERID command again and specifying another *userid*. Full buffer tracing of data remains in effect until you enter SET ETRACE OFF.

For example, to trace data for a virtual machine (with the user ID UserA) that has established a connection through the TSAF virtual machine, enter:

```
set etrace on userid usera
```

To trace data for a virtual machine with the user ID UserB, enter:

```
set etrace on userid userb
```

Data tracing for UserA stops and the information is stored in a TRSOURCE file. Tracing for UserB continues until you specify the user ID of another virtual machine on the USERID operand or enter SET ETRACE OFF.

Tracing a Link and a User ID at the Same Time: To trace data for a link and a user ID, you can enter two SET ETRACE commands or specify the LINK and USERID operands on one command.

For example, to trace data on a CTCA link with the virtual device address 04A0 and a virtual machine with the user ID UserA, you enter:

```
set etrace on link 04a0
set etrace on userid usera
```

Alternatively, you can specify both operands when you enter the command:

```
set etrace link 04a0 userid usera
```

Messages and Return Codes

ATSCOP004E
   Parameter *parameter* is not valid

ATSCOP005E
   A required parameter is missing

ATSCOP010I
External trace started

ATSCOP011I
External trace ended

ATSCOP024W
Link *linkunit* had not been added

ATSCOP025W
User *userid* has no APPC/VM connection to TSAF

You may also receive some of the following CMS messages:

DMSABE1329I
Trace records were lost. RC=*xx*

DMSDIE1329I
Trace records were lost. RC=*xx*

DMSETC1330E
TRSOURCE must be enabled before calling ETRACE

DMSTRE1331E
The DMSTRACE facility (Monitor Call Class 10) is not enabled.

DMSTRE1332E
Invalid value specified for the ID parameter

DMSTRE1332E
Invalid value specified for the *parm* parameter. It was not in the range of *value1* to *value2*. RC=24

DMSETC1333E
TRSOURCE is disabled. Buffer not written. RC=40

DMSETC1333E
TRSOURCE is in EVENT mode. Buffer not written. RC=40

DMSETC1333E
TRSOURCE is disabled. Record not added to buffer.

DMSITP1333E
TRSOURCE is in EVENT mode. Record not added to buffer.

DMSITP1333E
ETRACE is disabled. Record not added to buffer.

DMSITP1333S
I/O or severe error. Buffer not written.

DMSITP1334E
Buffer not initialized. Reissue ETRACE.

DMSITP1335E
Incorrect Monitor Call Class 10 Code.

DMSITP1336E
This function needs the CP Diagnose X'E0' command.

DMSITP1336E
This function needs the CP Diagnose X'EC' command. RC=40

DMSITP1337E
Insufficient storage to set up buffering. RC=40

DMSITP1338E
ETRACE set [ON|OFF] for DMSTRACE. RC=40

For information on these messages, see *z/VM: Other Components Messages and Codes* and *z/VM: CMS and REXX/VM Messages and Codes*.

STOP TSAF

```
►►─ STOP ── TSAF ─►◄
```

Purpose

Use the STOP TSAF command to stop running the TSAF virtual machine.

What Happens When You Enter this Command

After you enter the STOP TSAF command and TSAF accepts the STOP TSAF command, you get this message:

```
ATSCTL003I Termination is in progress
```

After the TSAF virtual machine has stopped, you get the CMS ready message (Ready;).

Messages and Return Codes

ATSCTL003I
 Termination is in progress

ATSCOP004E
 Parameter *parameter* is not valid

ATSCOP005E
 A required parameter is missing

If external tracing is ON when you enter STOP TSAF, TSAF issues the CMS command ETRACE with the END operand and processing ends. If this happens, you may receive any of the following CMS messages:

DMSABE1329I
 Trace records were lost RC=*xx*

DMSDIE1329I
 Trace records were lost RC=*xx*

DMSETC1330E
 TRSOURCE must be enabled before calling ETRACE

DMSTRE1331E
 The DMSTRACE facility (Monitor Call Class 10) is not enabled.

DMSTRE1332E
 Invalid value specified for the ID parameter

DMSTRE1332E
 Invalid value specified for the *parm* parameter. It was not in the range of *value1* to *value2*. RC=24

DMSETC1333E
 TRSOURCE is disabled. Buffer not written. RC=40

DMSETC1333E
 TRSOURCE is in EVENT mode. Buffer not written. RC=40

DMSETC1333E
 TRSOURCE is disabled. Record not added to buffer.

DMSITP1333E
 TRSOURCE is in EVENT mode. Record not added to buffer.

DMSITP1333E
 ETRACE is disabled. Record not added to buffer.

DMSITP1333S
I/O or severe error. Buffer not written.

DMSITP1334E
Buffer not initialized. Reissue ETRACE.

DMSITP1335E
Incorrect Monitor Call Class 10 Code.

DMSITP1336E
This function needs the CP Diagnose X'E0' command.

DMSITP1336E
This function needs the CP Diagnose X'EC' command. RC=40

DMSITP1337E
Insufficient storage to set up buffering. RC=40

DMSITP1338E
ETRACE set [ON|OFF] for DMSTRACE. RC=40

For information on these messages, see *z/VM: CMS and REXX/VM Messages and Codes* and *z/VM: Other Components Messages and Codes*.

Chapter 15. Generating TSAF Accounting and Link Statistics

PI

This chapter focuses on generating TSAF (Transport System Access Facility) accounting and link statistics within the z/VM environment. It outlines the types of accounting records that TSAF can produce: **Initialization**, **Session**, **Link Statistics**, and **Termination**. These records are generated when the `OPTION ACCT` directive is specified in the TSAF virtual machine's CP directory entry. Each record type captures specific data:

- **Initialization** and **Termination** records mark the start and end of TSAF activity.
- **Session** records log connection durations and data exchanged between user programs and resource managers.
- **Link Statistics** records monitor data flow and link status over time.

The records are generated by using the `DIAGNOSE X'4C'` instruction and include detailed fields such as user IDs, timestamps, data volumes, and symbolic destination names. This accounting framework supports performance monitoring and troubleshooting in TSAF-managed virtual networking environments.

To generate TSAF accounting records, you should specify the OPTION ACCT directory control statement in the TSAF virtual machine's CP directory entry (see "Creating the CP Directory Entry" on page 188). If you specify OPTION ACCT, you receive the following message when TSAF initializes:

```
ATSCAC006I TSAF link statistics and session accounting records will be
           generated
```

CP will send the accounting information to the virtual machine specified on the ACCOUNT1 parameter of the SYSTEM_USERIDS statement in the system configuration file. See *z/VM: CP Planning and Administration* for more information.

If you do not specify OPTION ACCT, you receive the following message:

```
ATSCAC007I No TSAF link statistics or session accounting records will be
           generated
```

Initialization Accounting Record

TSAF produces the initialization accounting record while it is starting. Use this record and the TSAF termination record to determine when TSAF was active.

Column	Contents
1 - 8	CP-provided user ID of the TSAF virtual machine
9 - 16	Reserved for IBM use
17 - 28	Date and time when the accounting record was generated, 'MMDDYYHHMMSS' (month, day, year, hours, minutes, and seconds).
29 - 32	Unit address (*vdev*) of the link, four characters in the form 'xxx'. If a symbolic destination name is recorded in columns 51 through 58, these columns will contains blanks (' ').
33 - 50	Reserved for IBM use.

Column	Contents
51 - 58	Symbolic destination name of the link when the virtual device address in columns 29 through 32 contains blanks (‘ ’). The symbolic destination name is a 1- to 8-character alphanumeric name, which is left-justified and padded on the right with blanks.
59 - 74	Reserved for IBM use
75 - 78	‘1ATS’ identifies the record as the initialization accounting record
79 - 80	‘C0’ identifies the accounting record code that CP provides using DIAGNOSE code X'4C'

Session Accounting Record

The session accounting record contains information about how long a user program is connected to a resource manager, using a specific resource. The record also indicates the amount of data the user program and the resource manager have exchanged.

This record is only generated for sessions routed through the TSAF virtual machine. If the connection is made to a resource in the same system as the user program, the connection is not routed through the TSAF virtual machine and session accounting records are not generated.

TSAF generates a session accounting record at the following times:

- Every hour
- When an APPC/VM session ends (SEVER)
- When the TSAF virtual machine stops.

The TSAF virtual machine (on the same system as the requester virtual machine that started the connection) issues the DIAGNOSE code X'4C' instruction to generate the records. For a session between the user program and the resource manager, only the system on which the user program is running would generate the session records.

Column	Contents
1 - 8	CP-provided user ID of the TSAF virtual machine
9 - 16	User ID of the requester virtual machine. If the security level of the conversation was SECURITY(NONE) for connections to a private or global resource, the user ID will be binary zeros.
17 - 28	Date and time when the accounting record was generated, ‘MMDDYYHHMMSS’ (month, day, year, hours, minutes, and seconds)
29 - 36	Resource name (for a connection to a global resource), a gateway name (for a connection to a resource in the SNA network), or the user ID of the private resource server virtual machine (for a connection to a private resource)
37 - 40	Time, in seconds, the session was active (unsigned binary fullword). This is the time since the session started or since the last accounting record was taken for this session.
41 - 44	Number of bytes of data were sent (unsigned binary fullword)
45 - 48	Number of bytes of data were received (unsigned binary fullword)
49	Type of name in bytes 29 to 36: ‘R’ for a global resource, ‘G’ for a gateway, or ‘U’ for a user ID of a private resource server virtual machine.
50 - 74	Reserved for IBM use
75 - 78	‘SATS’ identifies the record as a session accounting record issued by TSAF

Column	Contents
79 - 80	‘C0’ identifies the accounting record code that CP provides using DIAGNOSE code X'4C'.

Link Statistics Record

Use the link statistics record to determine the load on the link over time. TSAF generates a link statistics record at the following times:

- Every hour when a link is up
- When the system declares a link down
- When the TSAF virtual machine stops.

The TSAF virtual machines at both ends of the link generate link statistics records using the DIAGNOSE code X'4C' instruction.

Link statistics are kept for each logical link established by each TSAF virtual machine. Each LAN system connected to a TSAF collection has a Continuously Executing Transfer Interface (CETI) group of four consecutive virtual device addresses associated with it. The TSAF virtual machine uses the first virtual device address of the CETI group to represent the LAN system connection.

Column	Contents
1 - 8	CP-provided user ID of the TSAF virtual machine
9 - 16	Reserved for IBM use
17 - 28	Date and time when the accounting record was generated, ‘MMDDYYHHMMSS’ (month, day, year, hours, minutes, and seconds)
29 - 32	Unit address (*linkunit*) of the link (four characters in the form ‘xxxx’). If a symbolic destination name is recorded in columns 51 through 58, these columns will contain blanks (‘ ’).
33 - 36	Reserved for IBM use
37 - 40	Number of bytes of data were sent since the link came up or since the last accounting record was generated for this link (unsigned binary fullword)
41 - 44	Number of bytes of data that were received (unsigned binary fullword)
45 - 50	LAN node address that is the target of the logical link (binary). If there are no LAN connections in the collection, the field is filled with binary zeros.
51 - 58	Symbolic destination name of the link when the virtual device address in columns 29 through 32 contains blanks (‘ ’). The 1- to 8-character symbolic destination name is left-justified and padded on the right with blanks.
51 - 74	Reserved for IBM use
75 - 78	‘LATS’ identifies the record as a link statistics record issued by TSAF
79 - 80	‘C0’ identifies the accounting record code that CP provides using DIAGNOSE code X'4C'

Termination Accounting Record

TSAF produces the termination accounting record when you enter the STOP TSAF command and for some types of abnormal termination. When TSAF writes the termination accounting record, all other applicable session accounting and link statistics records have also been written. If, however, TSAF abnormally terminates without writing the termination accounting record, some session accounting and link statistics data may be lost.

Column	Contents
1 - 8	CP-provided user ID of the TSAF virtual machine
9 - 16	Reserved for IBM use
17 - 28	Date and time when the accounting record was generated, ‘MMDDYYHHMMSS’ (month, day, year, hours, minutes, and seconds)
29 - 74	Reserved for IBM use
75 - 78	‘0ATS’ identifies the record as the termination accounting record
79 - 80	‘C0’ identifies the accounting record code that CP provides using DIAGNOSE code X'4C'

PI end

Chapter 16. Collecting TSAF Problem Diagnosis Information

This chapter focuses on diagnosing problems within TSAF (Transport Services Access Facility) in the z/VM environment. It outlines several tools and methods for collecting diagnostic information, including the following:

- Console logs
- Dumps
- System trace data

When a TSAF abend (abnormal end) occurs, users are instructed to gather console output, dumps, and TRSOURCE files, assess system status, and recover the system by using specific commands. The chapter explains how to use the console log to capture error messages and abend details, and how to analyze dumps by using the Dump Viewing Facility. It also covers the use of system trace data and the TRSOURCE file for deeper diagnostics, and describes how to enable and manage external tracing. Additionally, the TSAF QUERY command is introduced as a way to retrieve real-time configuration and status information to aid in troubleshooting. The chapter emphasizes the importance of systematic data collection and analysis for effective problem resolution in TSAF environments.

The TSAF QUERY command is used to get problem diagnosis information. See "Using Interactive Service Queries" on page 241.

Note: The TSAF operator does not necessarily diagnose problems, especially from the TSAF virtual machine. The system programmer or whoever is responsible for diagnosing system problems usually uses dumps and system trace data.

Collecting Abend Information

When a TSAF ABEND occurs, you must do the following steps:

1. Collect information about the error.
   - Save the console sheet or spooled console output from the TSAF virtual machine.
   - Save and process any dumps that TSAF produces.

     When an abend occurs in TSAF, either because TSAF issued an abend or because a TSAF or CMS operation caused a program exception, TSAF produces a dump using the VMDUMP command (described in the *z/VM: CP Commands and Utilities Reference*). The dump is sent to TSAF's virtual reader.
   - Save any TRSOURCE file that contains TSAF data.
2. Collect other types of information about system status.
   - The status of real and virtual devices that TSAF is using.
   - The system load at the time of the failure on any systems using TSAF and the status of each system (for example, did another system abend?).
   - The types of applications using TSAF at the time and any information about them.
   - The physical connection configuration of the systems in use.
3. Recover from the abend to continue processing.

   After TSAF creates a dump, TSAF issues a LPSW (load program status word) instruction. If the CONCEAL option is on, as recommended, CP automatically IPLs CMS. Otherwise, the system operator must re-IPL CMS. Similarly, if the PROFILE EXEC does not contain the RUNTSAF command, you must restart the TSAF virtual machine.

z/VM: Other Components Messages and Codes lists the TSAF ABEND codes and their causes. See the *z/VM: Diagnosis Guide* for more information about diagnosing TSAF problems.

Using the Console Log

TSAF provides informational messages and error messages that can help you with problem determination. To track the console messages, enter:

```
spool console start to userid
```

The *userid* can be the user ID of the TSAF virtual machine or another virtual machine user ID to which you want TSAF to send the console log. You may want to add this to TSAF's PROFILE EXEC so a console log is always created.

To close the console log, enter:

```
spool console close
```

The log of messages received is sent to the specified user ID. See the *z/VM: CP Commands and Utilities Reference* for more information on the SPOOL command.

TSAF provides additional information at the time of an abend to help you diagnose the problem. The console log contains the abend code, the old program PSW (Program Status Word), and the contents of the general-purpose registers. TSAF also tries to determine the displacement of the module in which the abend occurred and the displacement of the calling module. Figure 117 on page 240 shows some of the messages that TSAF can issue in response to an abend condition.

```
ATSCAC999T TSAF system error
ATSCAB017I Abend code ATS999 at 022730
ATSCAB018I Program old PSW is FFE002FF 40022730
GPR0-7  00022FFC 000003E7 00022FDA 00052BC0 00208080 00020C58 0033E811 00000001
GPR8-F  7F3B78AF 603C0000 00020B64 00022D6F 50021D70 00022B48 40022718 00023FB0
ATSCAB019I Abend modifier is ATSCAC
ATSCAB021I Failure at offset 0A06 in module ATSCAC dated 87.020
ATSCAB022I Called from offset 04B4 in module ATSSCN dated 87.078
ATSCAB023I VMDUMP ATSCAB*ATSCAB1 07/10/87 19:35:00 taken
```

Figure 117. TSAF Console Log Example

Using TSAF Dumps to Diagnose Problems

The Dump Viewing Facility is a dump analysis and problem tracking tool. Use the Dump Viewing Facility to collect and diagnose problem data for the TSAF virtual machine. The console listing, as described in "Using the Console Log" on page 240, can help you diagnose problems without using dumps.

Because the TSAF virtual machine cannot process dumps, you must send the TSAF dump to the user ID specified on the DUMP parameter of the SYSTEM_USERIDS system configuration statement. See *z/VM: CP Planning and Administration* for more information.

The steps involved in using dumps to diagnose problems are:

1. Create a TSAF map (if it does not already exist), using the Dump Viewing Facility MAP TSAF command.
2. Create the TSAF dump, by entering:

   ```
   vmdump system format tsaf
   ```

3. Process the TSAF dump, using the Dump Viewing Facility DUMPLOAD command.
4. Diagnose the TSAF dump by doing the following:
   a. Look at the symptom record.
   b. Use the FDISPLAY subcommand of Dump Viewing Facility DUMPSCAN to display dump information.
   c. Format and display the trace records, using the TRACE subcommand of DUMPSCAN.

5. Optionally, print the TSAF dump, using the Dump Viewing Facility PRTDUMP command.

See the *z/VM: Dump Viewing Facility* for a description of the Dump Viewing Facility.

Using System Trace Data to Diagnose Problems

TSAF maintains an internal trace table within the TSAF virtual machine. Use the DUMPSCAN TRACE subcommand to display the internal trace table entries. TSAF also writes event trace entries to the system TRSOURCE file. Use the DUMPSCAN command to view TSAF entries and the PRTDUMP command to print the TRSOURCE file.

The TSAF SET ETRACE command lets you enable or disable external tracing for the TSAF virtual machine (see "SET ETRACE" on page 230). You can trace data on specific links to the TSAF virtual machine. You can also trace data for other virtual machines (user IDs) that have established an APPC/VM path through the TSAF virtual machine. To collect TSAF event trace records, enter the following command from the TSAF virtual machine:

```
set etrace on
```

Before entering the SET ETRACE command, you should enter the CP TRSOURCE command from a class C user ID. TRSOURCE enables external tracing of TSAF events and writes the TSAF trace records in blocks, rather than individually. Enter the command in the following format:

```
cp trsource traceid id type gt block for userid
```

The privilege class C, QUERY TRSOURCE command displays information about the current TRSOURCE settings. This information helps with problem determination. For more information about the TRSOURCE and the CP QUERY commands see the *z/VM: CP Commands and Utilities Reference*.

To access and review the TRSOURCE reader file, use the DUMPSCAN command. To print selected portions or the entire TRSOURCE file, use the PRTDUMP command. For more information about Dump Viewing Facility commands, see the *z/VM: Dump Viewing Facility*.

Using Interactive Service Queries

You can enter the TSAF QUERY command to display more information to help you diagnose problems. The TSAF QUERY command displays data about the TSAF configuration when the TSAF virtual machine is running. You can enter the following QUERY commands from the TSAF virtual machine:

Command Option	Function
QUERY COLLECT	Displays the processor names currently in the TSAF collection.
QUERY ETRACE	Displays the current setting of the external tracing.
QUERY GATEWAY	Displays the current list of gateways defined in the TSAF collection.
QUERY LINKS	Displays information about the links currently defined to TSAF.
QUERY RESOURCE	Displays the current list of global resources in the TSAF collection.
QUERY ROUTES	Displays route information at the node where the command was entered.
QUERY STATUS	Displays information about the state of the TSAF virtual machine.

See "QUERY" on page 222 for more specific information about the TSAF QUERY command.

Part 5. AVS Virtual Machine

This part of the document describes how to set up and use the AVS virtual machine.

- Chapter 17, "Preparing to Use the AVS Virtual Machine," on page 245 describes how to set up and prepare to use the AVS virtual machine.
- Chapter 18, "Using AVS," on page 255 describes how connections are routed between the AVS virtual machine, VTAM, and the SNA network.
- Chapter 19, "Operating the AVS Virtual Machine," on page 269 describes the commands that run and maintain the AVS virtual machine.
- Chapter 20, "Generating AVS Accounting Information," on page 291 describes AVS accounting records.
- Chapter 21, "Collecting AVS Problem Diagnosis Information," on page 295 describes how to diagnose AVS virtual machine problems by using dumps and system trace data.

Related Information: See the following sources for more information on the tasks in this section:

Task	Source
Establishing required virtual machine environments	Chapter 11, "Setting Up Server and Requester Virtual Machines," on page 165 *z/VM: Group Control System* *z/VM: CP Planning and Administration*
Communicating with VTAM and the SNA Network	*VTAM Resource Definition Reference* *SNA Transaction Programmer's Reference Manual for LU Type 6.2*
Installing and servicing code	*z/VM: CMS Planning and Administration* *z/VM: CMS Commands and Utilities Reference*
Improving system performance	*z/VM: Performance*

Chapter 17. Preparing to Use the AVS Virtual Machine

This chapter describes the following steps, which you must perform before using AVS:

- Define the AVS virtual machine to GCS
- Set up the AVS virtual machine and install AVS
- Prepare the AVS message repository (optional)
- Define the AVS virtual machine to VTAM
- Adjust the tuning parameters to improve performance (optional)
- Prepare the accounting module (optional)
- Set up the AVS virtual machine profiles
- Assure secure TSAF collection-to-SNA network communications
- Enable the AVS virtual machine to be autologged
- Set up the system to centralize operations.

Defining the AVS Virtual Machine to GCS

Before you can start AVS, you must use the AUTHUSER macro to define the AVS virtual machine to GCS as an authorized user ID. The user ID of the AVS virtual machine shipped with z/VM is AVSVM. You need to specify this user ID when you configure GCS during the installation process. See *z/VM: Group Control System* for information on how to define authorized user IDs to GCS.

Setting Up the AVS Virtual Machine

The AVS virtual machine handles communications between APPC/VM programs in a TSAF or CS collection and APPC programs in other systems in the SNA network. AVS is a VTAM application that runs with VTAM either in a separate virtual machine within the same GCS (Group Control System) group (see Figure 118 on page 245) or in the VTAM virtual machine in a GCS group. For information about setting up AVS in the VTAM virtual machine or in a separate virtual machine, see *z/VM: CP Planning and Administration*. For more information on GCS, see *z/VM: Group Control System*.

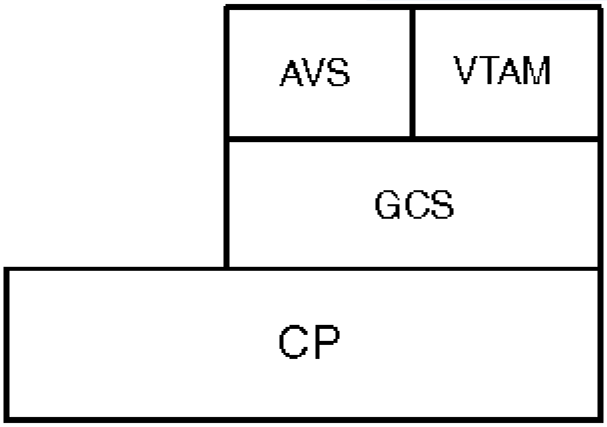


Figure 118. An AVS Virtual Machine Running in a z/VM System

AVS and VTAM provide the SNA network with a view of the TSAF or CS collection. The AVS operator defines the logical units (LUs), called gateways in VM, that represent the TSAF or CS collection. The SNA network views each collection as one or more LUs. For more information about the SNA network and LUs, see the *SNA Transaction Programmer's Reference Manual for LU Type 6.2*.

Updating the AVS Virtual Machine's CP Directory Entry

To define the AVS virtual machine, you must create a CP directory entry. This CP directory entry must contain the following statements:

- IUCV statements
- OPTION statement parameters
- Other directory statements.

The CP directory entry should also define at least 2MB of virtual storage and privilege class G, the default, for the AVS virtual machine. You should use the default value if you are not using alternate user ID support or if the Conversation Manager Routine (CMR) does not perform any system functions requiring a privilege class of B. (Privilege class BG is required when using alternate user ID support or when the CMR performs any system functions requiring a privilege class of B.)

The following sections describe the required CP directory statements. For more information about CP directory statements, privilege classes, and user access, see *z/VM: CP Planning and Administration*. See "Sample CP Directory Entry" on page 247 for an example of the CP directory entry for the AVS virtual machine.

IUCV Statements

The following IUCV statements should be specified for the AVS virtual machine; these statement specify how AVS allows access to resources. See Figure 119 on page 248 for examples of these statements.

Statement	Function
IUCV ALLOW	Allows any virtual machine to connect through the AVS virtual machine. **Note:** You may not want to let every requester virtual machine connect through the AVS virtual machine. Instead, you may explicitly authorize each requester virtual machine that wants to connect to a gateway. To do so, include IUCV *gatewayid* or IUCV ANY statements in each requester virtual machine's CP directory entry; see "IUCV Statement Syntax (Gateway Management)" on page 314. When you explicitly authorize each virtual machine, you should also give explicit directory authorization to the AVS virtual machine and each TSAF virtual machine in a TSAF collection. To do so, specify the IUCV *gatewayid* or IUCV ANY statement.
IUCV ANY	Allows the AVS virtual machine to connect to any virtual machine, resource, or gateway. See "IUCV Statement Syntax (Gateway Management)" on page 314 for more information.
IUCV *IDENT GATEANY GATEWAY REVOKE	Allows the AVS virtual machine connect to the Identify System Service (*IDENT) and identify any gateways in the TSAF or CS collection. This statement authorizes the AVS virtual machine as a gateway manager. You can also identify a specific gateway in the CP directory entry. See "Authorizing the AVS Virtual Machine to Manage Gateways" on page 314 for more information. One system can identify many global resources and gateways by specifying multiple IUCV statements. For more information about specifying IUCV statements, see Appendix A, "IUCV Directory Control Statement," on page 311. See "Resource Naming Considerations" on page 31 for information about the maximum number of gateways and resources each system can define.

OPTION Directory Control Statement

Specify the following parameters on the OPTION directory control statement; see Figure 119 on page 248 for an example of this statement.

Operand	Function
MAXCONN *nnnnn*	Defines the maximum number of IUCV and APPC/VM connections allowed for this virtual machine. The *nnnnn* value should be large enough to handle all planned intersystem APPC/VM paths that start from, or end at, your system. You should also consider the number of paths other programs in the same VM system may need. The default MAXCONN value is 64 connections; you should increase this value if AVS will process many allocation requests.
COMSRV	Enables the AVS virtual machine to act as a communications server. Communication servers can establish connections to other communications servers while handling requests for other users. With this option, the AVS virtual machine or any other communications server can put the user ID of the virtual machine that issued the APPCVM CONNECT in the connection parameter list. When AVS sends the connection request, the request contains information about the originating requester virtual machine. Without this option, CP would send the connect request with the communications server's user ID. This option also authorizes AVS to specify any SEVER or SENDERR code. CP does not verify the SEVER code. When AVS specifies a SENDERR code, CP does not generate a SENDERR code, but instead uses the one provided. For more information on APPCVM CONNECT and SEVER and SENDERR codes, see *z/VM: CP Programming Services*.
ACCT	Causes the AVS virtual machine to generate accounting records if the AVS-supplied accounting exit AGWACI is used; see "Preparing the AVS Accounting Module" on page 251.

Other Directory Statements

You also specify the following statements in the CP directory entry for the AVS virtual machine. See Figure 119 on page 248 for examples of these statements.

Statement	Function
IPL GCS PARM AUTOLOG	Requests automatic initialization of the GCS system. When the AVS virtual machine is autologged, GCS will be IPLed without operator intervention.
MDISK 191	Defines a minidisk for the AVS virtual machine; the minidisk may contain user execs, the PROFILE GCS exec and the AGWPROF GCS exec; see "Setting Up the AVS Virtual Machine Profile" on page 252 and "Creating the AGWPROF GCS Exec" on page 253.
MACHINE XA|XC	Defines the mode in which the AVS virtual machine will run. If using a Conversation Management Routine (see Appendix C, "Introduction to Service Pool Support," on page 325) that uses VM Data Spaces, specify MACHINE XC, otherwise use MACHINE XA.
NAMESAVE GCS	Allows AVS to IPL the GCS saved system.

Sample CP Directory Entry

Figure 119 on page 248 shows a sample CP directory entry for an AVS virtual machine. In this example, the IPL GCS statement causes the AVS virtual machine to automatically IPL GCS when the AVS virtual machine is logged on. For information on how to autolog the AVS virtual machine, see "Setting Up the AVS Virtual Machine for Autologging" on page 253. The 190 and 19E minidisks let the system operator use

CMS in the AVS virtual machine. See *z/VM: CP Planning and Administration* for more information about CP directory statements.

```
IDENTITY AVSVM AVSVM 32M 64M G
 INCLUDE IBMDFLT
 BUILD ON * USING SUBCONFIG AVSVM-1
 ACCOUNT  1   AVSVM
 MACH ESA
 IUCV ANY
 IUCV *IDENT GATEANY GATEWAY REVOKE
 IUCV ALLOW
 OPTION COMSRV MAXCONN 20 ACCT
 NAMESAVE GCS
 IPL  GCS PARM AUTOCR

SUBCONFIG AVSVM-1
 LINK    MAINT  193 193 RR
 MDISK 191 3390 3335 003 MO1RES MR RAVSOBJ  WAVSOBJ  MAVSOBJ
```

Figure 119. Sample AVS Virtual Machine CP Directory Entry

Installing AVS

AVS is installed into either Shared File System (SFS) or minidisk during initial install of z/VM. The install process also copies the PROFILE GCS to the AVS virtual machine's 191 minidisk.

Preparing the AVS Message Repository

AVS messages are located in the following common American English message files:

AGWUME TEXT
Mixed-case messages object code

AGWUMEB TEXT
Uppercase messages object code

AGWUME REPOS
Mixed-case messages source

AGWUMEB REPOS
Uppercase messages source.

The AGWUME REPOS message repository is already link-edited in the AVS run module shipped. You do not need to make any changes to use this repository. If you want to use the AGWUMEB repository, rename AGWUMEB TEXT to AGWUME TEXT then enter the VMFBLD command to link edit the object code into the AVS load library.

You cannot dynamically change AVS message repositories. You can, however, change the message text by changing the messages source file with these steps:

1. Make your changes to the source code repository.
2. Rebuild the object code file using the VMFNLS command. The VMFNLS command erases and rebuilds the object code file with the same file name as the repository file you specify on the command.

   If you changed the AGWUME REPOS, you may want to rename the current AGWUME TEXT file before you enter the VMFNLS command.
3. Ensure the object code file is named AGWUME TEXT.
4. Enter the VMFBLD command to link the AGWUME TEXT file into the AVS load library, AGW LOADLIB.

To perform these steps, use the VMSES/E Local Modification procedure for source maintained parts in the *z/VM: Service Guide*.

Defining AVS to VTAM

Before you can use AVS to communicate with programs in the SNA network, you must define the AVS virtual machine to VTAM. To define the AVS virtual machine to VTAM, you need to:

1. Set up a logon mode table.
2. Define an APPL definition statement for each AVS gateway in an application program major node.

Logon Mode Table

A logon mode table (also called a logmode table) contains session parameters, which describe how a session is conducted. A sample logon mode table, AGWTAB ASSEMBLE, is shipped with z/VM. Because the session send and receive pacing count parameters defined in this table affect AVS and VTAM performance, it is important that you set these parameters appropriately for your installation or modify or create your own logon mode tables. Setting up a logon mode table is described in the *VTAM Resource Definition Reference*. If you plan to use the predefined logon mode table, you do not need to create a separate table.

Application Program Major Node

To define an application program major node, you must enter a VTAM VBUILD statement (or add entries to an existing application program major node). You must also enter an APPL statement for each gateway that is managed by this AVS virtual machine. See the *VTAM Resource Definition Reference* for more information about the VTAM VBUILD and APPL statements.

Figure 120 on page 249 is an example of an APPL statement that defines an application program major node for the AVSVM virtual machine and the global gateway GLBLGAT1. (You may specify other parameters, depending on the needs of your installation.) For more information about the APPL macro, see the *VTAM Resource Definition Reference*.

```
AVSVM    VBUILD TYPE=APPL
GLBLGAT1 APPL   APPC=YES,                                              X
                AUTHEXIT=YES,                                          X
                AUTOSES=10,                                            X
                DSESLIM=100,                                           X
                DMINWNL=50,                                            X
                DMINWNR=50,                                            X
                MODETAB=AGWTAB,                                        X
                PARSESS=YES,                                           X
                SYNCLVL=SYNCPT,                                        X
                VERIFY=NONE,                                           X
                SECACPT=ALREADYV
```

Figure 120. VTAM APPL Statement Example

The `AVSVM VBUILD TYPE=APPL` statement defines an application program major node to VTAM. The APPL definition statement defines GLBLGAT1 as an application program that is in the VTAM domain and includes it in the major node. This APPL statement defines GLBLGAT1 as an LU to the SNA network.

The APPC parameter, which is required, indicates that communications use the APPC protocol.

The AUTHEXIT=YES parameter, which is recommended, lets AVS exit routines run in supervisor state even if the AVS virtual machine is not authorized by GCS to do so. This means VTAM and GCS will not validate the data coming through the AVS virtual machine. Because the AUTHEXIT parameter affects AVS performance, it is important that this parameter be specified as YES.

The AUTOSES parameter defines the number of contention winner sessions that should be automatically activated. In this example, 10 sessions are activated.

The DSESLIM, DMINWNL, and DMINWNR parameters define the default session limit, contention winner, and contention loser values. These values are used for session limit negotiation whenever a CNOS is sent to or received from a remote LU for which no explicit AGW CNOS command has been issued. When remote LUs under user control are present in the network, these default values should kept small to avoid waste of network resources. When the local LU issues an AGW CNOS command, its values are used for the duration of the session.

The DSESLIM parameter defines the maximum number of sessions, in this example 100, to be allocated between the remote LU and the local LU (GLBLGAT1) for a given mode name. The DMINWNL parameter defines the minimum number of parallel sessions, within a mode name group, for which the application program is guaranteed to be the contention winner. In this example, the local LU, GLBLGAT1, is guaranteed to be the contention winner for 50 sessions. (A mode name group is a group of sessions for a given mode name.) The DMINWNR parameter defines the minimum number of parallel sessions, in this case 50, for which the remote LU is guaranteed to be the contention winner.

The MODETAB parameter identifies AGWTAB as the logon mode table that associates each logon mode name with a set of session parameters. (AGWTAB ASSEMBLE is the sample logon mode table shipped with z/VM.) This parameter is not necessary if you plan to use only VTAM-provided mode names.

The PARSESS parameter lets the gateway GLBLGAT1 have multiple parallel sessions with another LU.

The SYNCLVL parameter defines the synchronization level allowed on conversations. You can specify CONFIRM, which allows a partner to request confirmation, or SYNCPT, which allows a partner to enter SYNCPT functions. SYNCPT is the highest level supported on z/VM.

The VERIFY parameter specifies whether VTAM performs LU-LU verification of partner LUs:

VERIFY=NONE
No verification of the partner LU's identity will be performed. This is the default.

VERIFY=OPTIONAL
Verification of the partner LU's identity will only be performed if a password has been defined for the LU-LU pair.

VERIFY=REQUIRED
The identity of every partner LU will be verified. Every partner LU must have an LU-LU password defined. Any partner LUs that do not have an LU-LU password defined cannot establish LU 6.2 sessions with this AVS gateway.

For VERIFY=OPTIONAL or VERIFY=REQUIRED, a security management product, such as RACF, must be installed and the APPCLU class must be active. For more information on how to define the password for a pair of LUs, consult the security managment product publications.

For RACF/VM, use the following procedure.

1. Give AVS permission to use the RACROUTE interface:

```
PERMIT ICHCONN CLASS(FACILITY) ID(avsvm) ACCESS(UPDATE)
```

2. Establish connection between AVS and RACF:

```
LINK RACFVM 305 305 RR readpwACCESS 305 fmGLOBAL LOADLIB RPIGCS
LOADCMD RPIUGCS RPIATGCS
LOADCMD RPISSSRC RPISSSRC
RPISSSRC
RPIUGCS INIT
```

3. Define an APPCLU profile for each partner LU that is to be verified:

```
RDEFINE  APPCLU  netid.gateway.partnerlu            SESSION(SESSKEY(password))
                 UACC(NONE)
```

4. Activate the APPCLU class:

```
SETROPTS CLASSACT(APPCLU)
```

5. Perform whatever actions are required on the partner LU to define the same password for the same pair of LUs.

The SECACPT parameter defines the security acceptance level that this LU will accept from a remote LU for which no explicit AGW CNOS command has been issued. You should specify the highest level of security that can be accepted on a conversation through this LU.

SECACPT Level	When To Specify
ALREADYV	Transaction programs can specify any security level (SAME, PGM, and NONE).
CONV	Transaction programs specify security levels PGM or NONE; security level SAME is not accepted.
NONE	Security levels SAME and PGM are not accepted.

Note: If a local AGW CNOS is issued, this parameter has no meaning.

Performance Considerations

The performance of the AVS virtual machine is affected by several parameters:

- Session pacing count parameters in the logon mode table (see “Logon Mode Table” on page 249)
- AUTHEXIT parameter on the APPL statement issued for each gateway defined (see Figure 120 on page 249) and the statement descriptions following the figure.
- AVS tuning parameters.

The AGWTUN ASSEMBLE file, which is linked to the AVS module, contains AVS tuning parameters. The AVS virtual machine can use the default parameter settings that are provided. However, you may change the following parameters to meet the needs of your installation:

Pause count
Defines the number of events an AVS task processes before giving control to another task.

Accounting record timer
Defines the number of hours between the creation of accounting records for all currently active conversations.

VTAM-VM request transformation control
Defines the balance of translating requests between APPC/VM and APPC/VTAM protocols and the resources used during the translations.

Problem dump count
Defines the number of problem dumps that can be taken when errors occur during an AVS session. You can review this dump to diagnose the cause of the problem.

For more information about the AVS tuning parameters and about the overall system performance, see *z/VM: Performance*.

Security Level Considerations

With ACF/VTAM 3.3 or later, connections with SECURITY(NONE), SECURITY(PGM), and SECURITY(SAME) are allowed between APPC/VM programs in a TSAF or CS collection and APPC programs in the SNA network. With earlier releases of ACF/VTAM, connections support only SECURITY(NONE) and SECURITY(PGM). See “What Is Conversation Security?” on page 26 for a description of the different levels of security.

Preparing the AVS Accounting Module

PI

AVS accounting inputs three types of accounting records to the AGWACI module, which is an installation-wide exit. AVS generates the following accounting records (see Chapter 20, “Generating AVS Accounting Information,” on page 291 for more information):

Initialization
Generated during AVS initialization and contains information about when AVS started.

Conversation
Generated while a conversation is active and contains information about conversations established through AVS.

Termination
Generated during AVS termination and contains information about when AVS ended.

AGWACI ASSEMBLE uses the DIAGNOSE code X'4C' subcode X'0010' instruction to write these preformatted 70 byte input accounting records to a CP spool file. See *z/VM: CP Programming Services* for more information on the DIAGNOSE instruction. The status of the DIAGNOSE code is returned to the caller in register 15. AGWACI returns this status when it returns to AVS.

The AGWACI ASSEMBLE file is written in assembler language and contains internal DSECTs of the accounting data passed to it. You may change the destination or format of the accounting records by coding your own version of AGWACI. (To reassemble the file, use the CMS ASSEMBLE command or VMFASM.) Refer to the IBM-supplied routine as a model. Consideration should be given to the execution time of this module because it runs inline with AGW routines. Excessive times may cause data loss or conversations to time-out.

On entry to, and return from, AGWACI, the registers must have the following values:

Access	Register	Contents
Entry	1	Address of the pointer to the accounting information.
	13	Pointer to the standard register save area.
	14	Return address.
	15	Entry point address.
Exit	14	Return address
	15	Set to 0 by the DIAGNOSE instruction; AVS accounting will not be initialized if a nonzero value is returned.

PI end

Setting Up the AVS Virtual Machine Profile

Before you can operate the AVS virtual machine, you must load the AVS load library, define the command name for the AVS load module, and start the AVS virtual machine.

To do so, you can enter the required GCS and AVS commands each time the GCS and AVS virtual machines are started. However, for increased efficiency, you can place the GCS and AVS commands in a profile. This profile, called PROFILE GCS, runs automatically when the GCS saved segment is initialized in the AVS virtual machine.

Figure 121 on page 252 shows a portion of a sample PROFILE GCS file for AVS. This profile loads the AVS load module, assigns the module command name, and starts AVS.

```
/*********************************************************************/
/* PROFILE GCS - loads the AVS load module, assigns the module's     */
/* command name, and starts AVS.                                     */
/*********************************************************************/
Trace 0
'CP SET RUN ON'
'ACCESS 193 D'

'GLOBAL LOADLIB AGW'
'LOADCMD AGW AGW'
'AGW START'                /* Start AVS                              */
```

Figure 121. PROFILE GCS Example

The GLOBAL LOADLIB AGW command loads the load library that contains the AVS load module (AGW).

The GCS command LOADCMD AGW AGW assigns the command name AGW to the AVS program module. The GCS virtual machine then recognizes that the AGW prefix identifies AVS commands. After the command name is defined, AVS commands can be entered. The AVS command AGW START starts the AVS virtual machine.

Creating the AGWPROF GCS Exec

As the last step in AVS initialization, the AGWPROF GCS exec is started. This exec contains commands that are issued when AVS has initialized. For example, you can specify AGW ACTIVATE GATEWAY and AGW CNOS commands. You can also include non-AVS commands or other REXX execs in the AGWPROF GCS exec. Figure 122 on page 253 is an example of an AGWPROF GCS exec.

```
/* AGWPROF GCS, which activates one gateway for global resource */
/* communications and sets values for the session.              */
'AGW ACTIVATE GATEWAY GLBLGAT1 GLOBAL'
'AGW CNOS GLBLGAT1 REMOTLU1 TABLDAT 10 5 5'
```

Figure 122. AGWPROF GCS Example

In this example, the global gateway GLBLGAT1 is activated. (GLBLGAT1 is defined to VTAM on an APPL definition statement described Figure 120 on page 249.) The AGW CNOS command specifies that a maximum of 10 sessions may be established between the gateway GLBLGAT1 and the remote LU REMOTLU1. For more information, see the AGW ACTIVATE GATEWAY command (described in “Purpose” on page 271) and the AGW CNOS command (described in “Purpose” on page 274).

Setting Up the AVS Virtual Machine for Autologging

The GCS, VTAM, and AVS virtual machines can be autologged when the system is started. The following scenario describes the steps needed to autolog the virtual machines. This scenario assumes that AVS has been previously defined to GCS and VTAM and that AVS and VTAM are running in separate virtual machines.

1. The AUTOLOG1 virtual machine autologs the GCS virtual machine.
2. The GCS virtual machine autologs the VTAM virtual machine. (The GCS virtual machine's PROFILE GCS contains an XAUTOLOG statement for VTAM.)
3. The VTAM virtual machine autologs the AVS virtual machine after VTAM START is issued. (The VTAM virtual machine's PROFILE GCS contains an XAUTOLOG statement for AVS.)
4. The AVS virtual machine IPLs GCS. (The AVS virtual machine's CP directory entry contains an IPL GCS PARM AUTOLOG statement as shown Figure 119 on page 248.)
5. AVS is loaded and started. (The AVS virtual machine's PROFILE GCS contains the statements described in “Setting Up the AVS Virtual Machine Profile” on page 252.)
6. Gateways are activated and session limits and contention values are set. (The AVS virtual machine's AGWPROF GCS contains the AVS commands described in “Creating the AGWPROF GCS Exec” on page 253.)

Setting Up Centralized System Operations

You can centralize system operations so that a central site oversees system operation for the TSAF or CS collection; this site can also receive messages from systems in the collection that cannot be automatically handled at the distributed system (described “Setting Up Centralized System Operations” on page 194). As with the TSAF virtual machine, you can also use the Programmable Operator Facility and the Single Console Image Facility (SCIF) to centralize operation of the AVS virtual machine.

If you use NetView or RSCS to centralize the operation of your TSAF or CS collection, the Programmable Operator Facility can forward messages that are acted upon at the local site to a central site virtual machine where a real operator is located.

For information about the Programmable Operator Facility, see *z/VM: CMS Planning and Administration*. For information about SCIF, see *z/VM: Virtual Machine Operation*.

Chapter 18. Using AVS

If at least one AVS virtual machine is installed and running in a system is a TSAF or CS collection, APPC/VM programs can communicate with resources in an SNA network. AVS, along with VTAM, lets APPC/VM programs in a TSAF or CS collection communicate with:

- APPC/VM programs in another TSAF or CS collection
- APPC/VM programs in a VM system in the SNA network
- APPC programs in a non-VM system in the SNA network
- APPC programs in a workstation in the SNA network.

APPC programs in the SNA network can communicate with global and private resources in the TSAF or CS collection. VTAM provides the LU 6.2 protocol services necessary to communicate with the remote LU where the APPC program is located. AVS translates requests between APPC/VM and APPC/VTAM. VTAM and AVS run in a GCS group.

Note: An LU in the SNA network must specify LU type 6.2 when starting a session with an AVS gateway.

Understanding the SNA Network View

A TSAF or CS collection is defined to the SNA network as one or more LUs. These LUs are called ***gateways*** or gateway LUs in VM. APPC programs in the SNA network view a TSAF or CS collection as one of these LUs. The APPC programs view the global and private resources in the TSAF or CS collection as transaction programs residing at their corresponding LU.

Gateways are used for either inbound or outbound connections. Inbound connections are connection requests issued by APPC programs in the SNA network to access resources in a TSAF or CS collection. APPC/VM programs in the TSAF or CS collection issue outbound connection requests to access APPC programs in the SNA network.

APPC programs in the SNA network can request connections to global and private resources in the TSAF or CS collection. The APPC programs identify the target of their request by specifying an LU name and a transaction program name in the connection request. The LU name identifies the gateway and the transaction program name identifies the resource.

To route the connection request to the correct resource, AVS identifies the type of resource specified in the connection request. AVS uses the gateway name (the LU name) specified by the APPC program (the remote transaction program) to identify the resource name space (that is, what type of resource is the target of the connection request). AVS can define multiple LUs (gateways) to VTAM to identify the resource name space.

APPC/VM programs in the TSAF or CS collection can request to connect to resources located in remote LUs in the SNA network. The user programs identify the target by specifying either a symbolic destination name or a locally known LU name and a transaction program name. VM and VTAM translate the locally-known LU name into a fully-qualified name, which routes the connection in the SNA network.

There are two types of gateways: global and private. A global gateway is used for inbound connections to global resources and for outbound connections from an APPC/VM program. A private gateway is used for inbound connections to private resources and for outbound connections from an APPC/VM program. Gateway types are only known to AVS. VTAM and the SNA network only know LU names. Figure 123 on page 256 shows the inbound and outbound connection requests being routed on the two types of gateways.

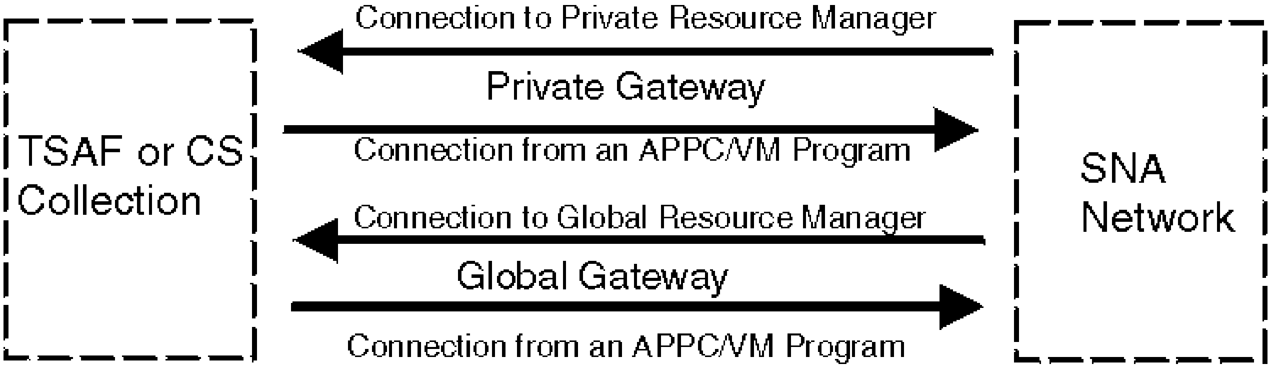

TSAF or CS Collection
Connection to Private Resource Manager
Private Gateway
Connection from an APPC/VM Program
Connection to Global Resource Manager
Global Gateway
Connection from an APPC/VM Program
SNA Network

Figure 123. Gateway Overview

Global Gateways

The AVS operator defines a gateway to be a global gateway on the AGW ACTIVATE GATEWAY command. Global resource managers identify themselves to the TSAF or CS collection as the owner of a particular global resource. In a TSAF collection, the TSAF virtual machine ensures that global resource names and the user IDs of server virtual machines are unique. The APPC program specifies the global resource name (the transaction program name) and the global gateway name (the LU name) on its connection request. On an inbound connection through a global gateway, the connection request is routed to the server virtual machine that TSAF identifies as the owner of the global resource.

Generally, only one global gateway needs to be defined for a TSAF or CS collection. This is possible because global resource names are unique and TSAF knows the location of the server virtual machines for global resources. However, if AVS and VTAM virtual machines are running in each system in the TSAF collection, a global gateway could be defined on each system. Connections with SECURITY(NONE), SECURITY(SAME), and SECURITY(PGM) are supported through global gateways.

Private Gateways

The AVS operator defines a gateway to be a private gateway on the AGW ACTIVATE GATEWAY command. The AVS operator can specify a server virtual machine user ID to be associated with the private gateway. Private resources are not identified to the TSAF or CS collection. Private resources have unique names within the server virtual machine in which they reside. Therefore, an inbound request to connect to a private resource must identify the private resource server virtual machine.

The APPC program specifies the location of the private resource by one of the following:

- Routing the connection to a dedicated private gateway that has only one private resource server virtual machine user ID associated with it.
- Routing the connection to a nondedicated private gateway and specifying the user ID and password of the target private resource server virtual machine in its connection request.

Dedicated private gateways are defined using the AGW ACTIVATE GATEWAY command and identify a particular private resource server virtual machine user ID. The transaction program requesting a connection to a transaction program (resource) at this LU (dedicated private gateway) is routed to the private resource server virtual machine that was identified when the gateway was defined. Dedicated private gateways should be used to access multiuser private resources. Like global gateways, connections with SECURITY(NONE), SECURITY(SAME), and SECURITY(PGM) are supported through dedicated private gateways. A dedicated private gateway should be defined for each multiuser private resource server virtual machine (for example, departmental virtual machines or batch-like server virtual machines).

Nondedicated private gateways are also defined using the AGW ACTIVATE GATEWAY command. The transaction program requesting a connection to a resource at this LU (nondedicated private gateway) is routed to the private resource server virtual machine identified by the access security user ID specified in the connection request. Only connections with SECURITY(PGM) or SECURITY(SAME) are supported through nondedicated private gateways because the access security user ID is used to identify the target server virtual machine. Only one nondedicated private gateway needs to be defined in a TSAF or CS collection. For example, this gateway can be reserved for users accessing their own virtual machines. If AVS and VTAM virtual machines are running in each system in the TSAF or CS collection, a nondedicated private gateway could be defined on each system.

A private gateway can be associated with a Conversation Management Routine (CMR). The CMR is an application supplied service pool manager. It routes incoming APPC connections to available service pool machines. See Appendix C, "Introduction to Service Pool Support," on page 325 for more information on CMR.

AVS Examples

This section contains scenarios for setting up gateways and using them to share resources in TSAF collection and the SNA network. Note that APPC/VM programs in a CS collection can also use AVS gateways to access resources in the SNA network. The scenarios are:

- Inbound connection from an APPC program in the SNA network to a global resource in a TSAF collection (see "Inbound Connection to Global Resources" on page 257)
- Inbound connection from an APPC program in the SNA network to a private resource in the TSAF collection through a dedicated private gateway (see "Inbound Connection to Private Resources through Dedicated Private Gateways" on page 259)
- Inbound connection from an APPC program in the SNA network to a private resource in the TSAF collection through a nondedicated private gateway (see "Inbound Connection to Private Resources through Nondedicated Private Gateways" on page 261)
- Outbound connection from an APPC/VM program in the TSAF collection to an APPC/VM program in another TSAF collection (see "Outbound Connection to a Global Resource in Another TSAF Collection" on page 263)
- Outbound connection from a user in the TSAF collection to an SFS file pool in another TSAF collection (see "Outbound Connection to a Shared File System File Pool" on page 265).

See Chapter 19, "Operating the AVS Virtual Machine," on page 269 for a description of the AVS commands mentioned in the scenarios. See *z/VM: CP Programming Services* for a complete description of APPC/VM functions mentioned in the scenarios.

Inbound Connection to Global Resources

An APPC program in an SNA network can access a global resource in a TSAF collection without the remote user being logged onto a system in a TSAF collection. The APPC program outside the TSAF collection does not need to identify itself to the TSAF collection. This scenario involves:

- A TSAF collection made up of one VM system with:
  - An AVS virtual machine called AVSa
  - A VTAM virtual machine called VTAMa
  - A global resource manager called RESMGR that manages the global resource RES1.
- An APPC transaction program TRANSP1 located at LU REMOTLU1 in the SNA network.

Note: Ensure you code the SECACPT parameter on the VTAM APPL statement (see Figure 120 on page 249). If you do not specify CONV or ALREADYV, VTAM downgrades the security from PGM to NONE; this causes the remote system to reject the request.

Step 1—AVS Virtual Machine Defines a Global Gateway

While AVSa is running, its operator defines the global gateway GLBLGAT1 to be used by APPC programs in the SNA network to communicate with global resources in the TSAF collection by entering the AVS command:

```
agw activate gateway glblgat1 global
```

The operator then sets the session limits and the contention winner and loser values for this gateway by entering the AVS command:

```
agw cnos glblgat1 remotlu1 tabldat 10 5 5
```

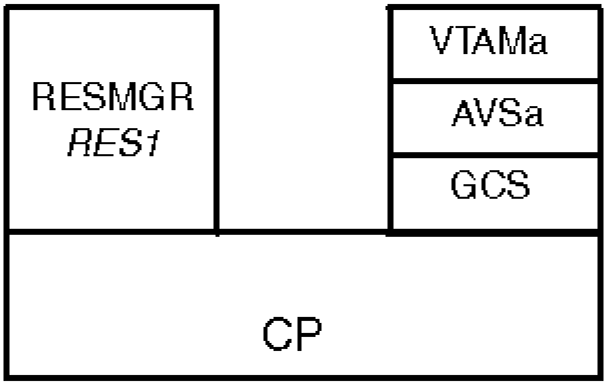


Figure 124. Defining a Global Gateway

Note: This step is not necessary if the AGWPROF GCS file for the AVSa virtual machine contains the entries shown in Figure 122 on page 253.

Step 2—A Transaction Program Requests a Connection to a Global Resource

TRANSP1 requests a connection to the transaction program RES1 located at LU GLBLGAT1. The connection request is routed to GLBLGAT1 (a global gateway in the TSAF collection). When the connection request comes in, AVSa translates the APPC/VTAM allocation request for the transaction program RES1 into an APPC/VM CONNECT request for the global resource RES1. AVS knows to make the request for a global resource because the connection request came into the TSAF collection through a global gateway.

If the connection was SECURITY(PGM), the user ID and password that TRANSP1 specified in its allocation request are verified to ensure that they are valid in the TSAF collection. CP then routes the APPC/VM CONNECT request to RESMGR (see Figure 125 on page 258).

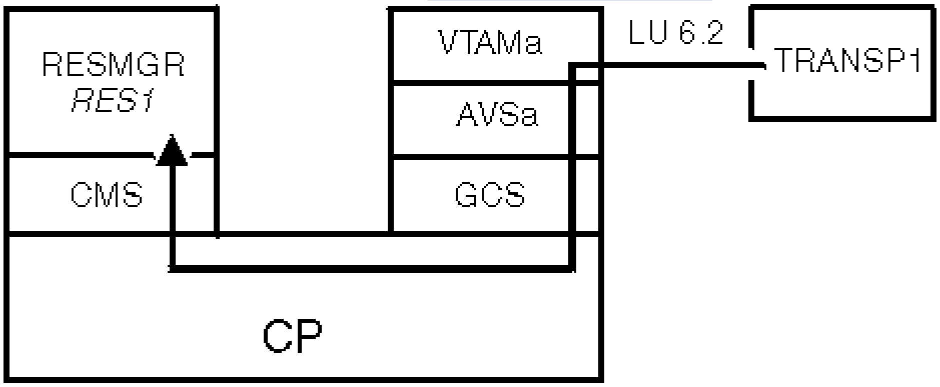


Figure 125. Remote APPC Transaction Program Issuing a Request

Step 3—A Transaction Program Communicates with a Global Resource

RESMGR, if necessary, receives the allocation data sent by TRANSP1 and accepts the connection for the global resource RES1. RESMGR and TRANSP1 exchange data using supported APPC functions until one ends the conversation (see Figure 126 on page 259).

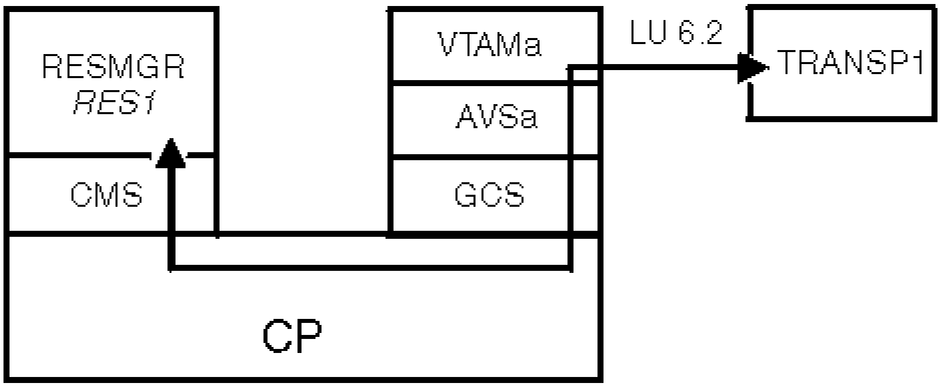


Figure 126. Remote APPC Transaction Program Communicating with a Global Resource

Inbound Connection to Private Resources through Dedicated Private Gateways

An APPC program in an SNA network can use a dedicated private gateway to access a private resource located in a server virtual machine in the TSAF collection. When the connection request is made, the private resource server virtual machine does not have to be logged on and the private resource manager does not have to be running. APPC programs in the SNA network should use this type of gateway to access multiuser private resources.

This scenario involves:

- A TSAF collection made up of one VM system with:
  - An AVS virtual machine called AVSa
  - A VTAM virtual machine called VTAMa
  - A private resource manager program named PRIMGR3, running in a virtual machine whose user ID is USERA, that manages a private resource RES3.
- An APPC transaction program TRANSP3 located at LU REMOTLU1 in the SNA network.

Step 1—AVS Virtual Machine Defines a Dedicated Private Gateway

While AVSa is running, its operator defines the dedicated private gateway PRIVGAT3 to be used by APPC programs to communicate with private resources managed by PRIMGR3 by entering the AVS command:

```
agw activate gateway privgat3 private userid usera
```

The operator sets the session limits and the contention winner and loser values for this gateway by entering the AVS command:

```
agw cnos privgat3 remotlu1 secure 30 10 20
```

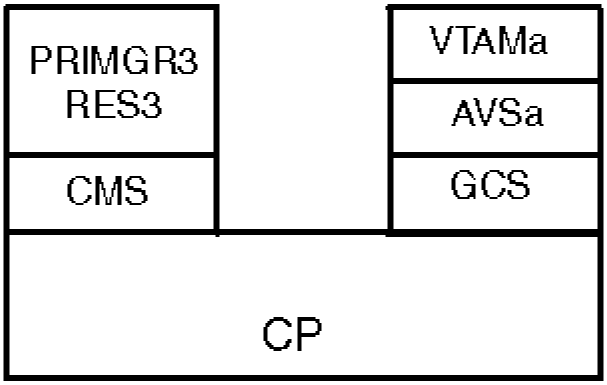


Figure 127. Defining a Dedicated Private Gateway

Step 2—A Transaction Program Requests a Connection to a Private Resource

TRANSP3 requests a connection to a transaction program RES3 located at LU PRIVGAT3. The connection request is sent to PRIVGAT3 (a dedicated private gateway in the TSAF collection). When AVSa receives the connection request, it translates the APPC/VTAM allocation request for the transaction program RES3 into an APPC/VM CONNECT request for the private resource RES3.

Because the connection is through a dedicated private gateway, CP routes the APPC/VM CONNECT request to USERA, the user ID defined on the AGW ACTIVATE GATEWAY command. If USERA is not logged on, it is autologged and the resource manager PRIMGR3 is started. CP queues the connection pending interrupt for USERA's CMS virtual machine. USERA receives the interrupt when it enables for APPC/VM interrupts. When CMS receives the interrupt, it checks its $SERVER$ NAMES file to see if TRANSP3 is authorized to connect (see Figure 128 on page 260).

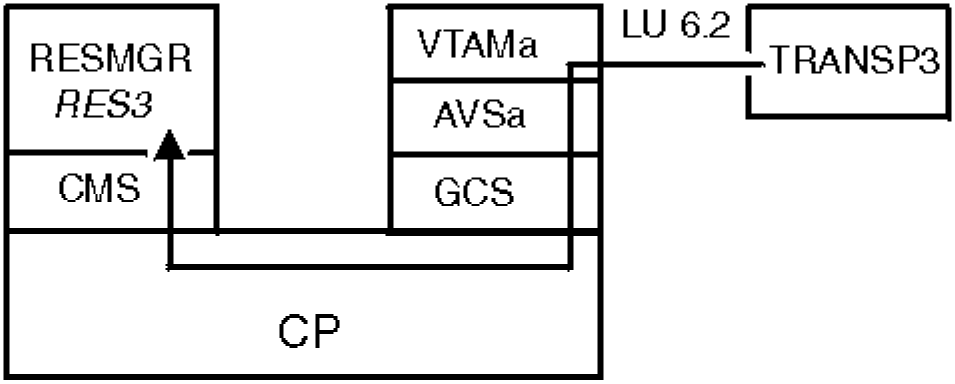


Figure 128. Remote APPC Transaction Program Issuing a Request

Step 3—A Transaction Program Communicates with a Private Resource

PRIMGR3, if necessary, receives the allocation data sent by TRANSP3 and accepts the connection for the private resource RES3. PRIMGR3 and TRANSP3 exchange data using supported APPC functions until one ends the conversation (see Figure 129 on page 261).

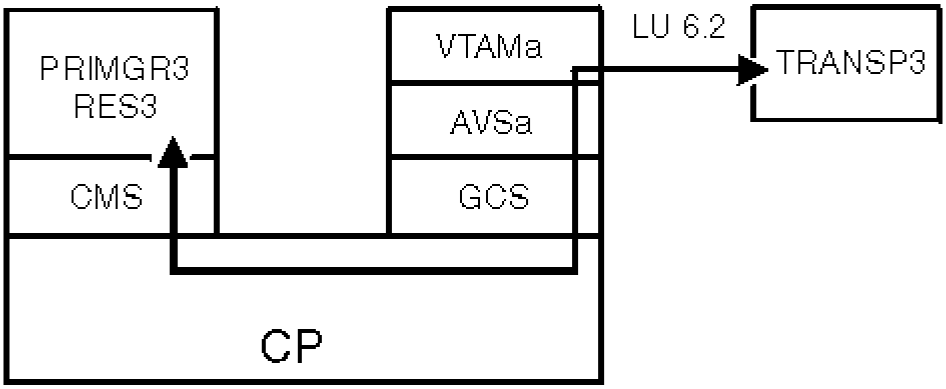


Figure 129. Remote APPC Transaction Program Communicating with a Private Resource

Inbound Connection to Private Resources through Nondedicated Private Gateways

An APPC program in an SNA network can use a nondedicated private gateway to access a private resource located in a server virtual machine in the TSAF collection. When the connection request is made, the server virtual machine does not have to be logged on and the private resource manager does not have to be running.

This scenario involves:

- A TSAF collection made up of one VM system with:
  - An AVS virtual machine called AVSa
  - A VTAM virtual machine called VTAMa
  - A private resource manager PRIMGR2, running in a virtual machine whose user ID is USR192 and password is PASS119, that manages a private resource RES2.
- An APPC transaction program TRANSP2 located at LU REMOTLU1 in the SNA network.

Step 1—AVS Virtual Machine Defines a Nondedicated Private Gateway

While AVSa is running, its operator enters the following command to define the nondedicated private gateway PRIVGAT2:

```
agw activate gateway privgat2 private
```

The operator then sets the session limits and the contention winner and loser values for this gateway by entering the AVS command:

```
agw cnos privgat2 remotlu1 secure 25 10 15
```

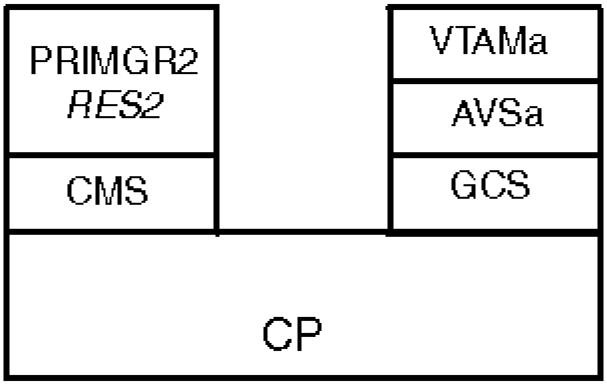


Figure 130. AVS Operator Defining a Nondedicated Private Gateway

Step 2—A Transaction Program Requests a Connection to a Private Resource

TRANSP2 requests a connection to a transaction program RES2 located at LU PRIVGAT2. It issues an allocation request that identifies RES2 as the transaction program name, SECURITY(PGM) as the conversation security level, and the user ID USR119 and the password PASS119 as the security information. The connection request is sent to PRIVGAT2. When AVSa receives the connection request, AVSa translates the APPC/VTAM allocation request for the transaction program RES2 into an APPC/VM CONNECT request for the private resource RES2.

The user ID and password (USR119 and PASS119) that TRANSP2 specified in the allocation request are verified to ensure that they are valid on a system in the TSAF collection. CP then routes the APPC/VM CONNECT request to the target virtual machine, USR119, specified as the access security user ID in the allocation request. If USR119 (the private resource server virtual machine) is not logged on, it is autologged and the resource manager PRIMGR2 is started. CP queues the connection pending interrupt for USR119's CMS virtual machine. USR119 receives the interrupt when it enables for APPC/VM interrupts (see Figure 131 on page 262). USR119's $SERVER$ NAMES file is not checked because the user ID and password specified in the connection request were the same as the server virtual machine's user ID and password.

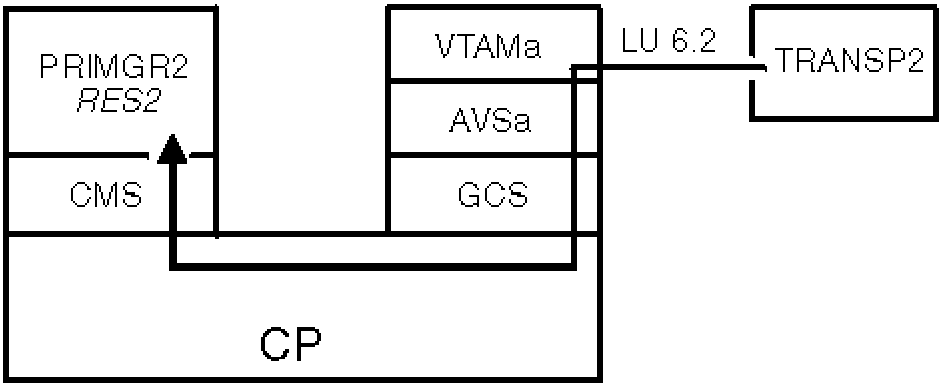


Figure 131. Remote APPC Transaction Program Issuing a Request

Step 3—A Transaction Program Communicates with a Private Resource

PRIMGR2, if necessary, receives the allocation data sent by TRANSP2 and accepts the connection for private resource RES2. PRIMGR2 and TRANSP2 exchange data using supported APPC functions until one ends the conversation (see Figure 132 on page 263).

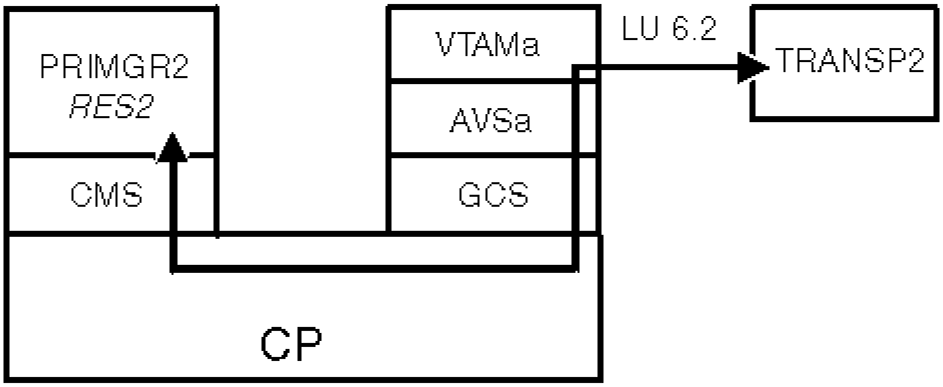


Figure 132. Remote APPC Transaction Program Communicating with a Private Resource

Outbound Connection to a Global Resource in Another TSAF Collection

An APPC/VM program in a TSAF collection can connect to an APPC program in a non-VM system or to an APPC/VM program in a VM system in an SNA network.

This scenario involves:

- A TSAF collection, Collection A, made up of one VM system with:
  - An AVS virtual machine called AVSa
  - A VTAM virtual machine called VTAMa
  - An APPC/VM program PGM1 that resides in the virtual machine USER1.
- A TSAF collection, Collection B, made up of one VM system with:
  - An AVS virtual machine called AVSb
  - A VTAM virtual machine called VTAMb
  - A global resource manager RESMGR4 that manages the global resource RES4.

Step 1—AVS Virtual Machines Define Global Gateways

While AVSa is running, its operator enters the following command to define a global gateway COLLECTA:

```
agw activate gateway collecta global
```

The operator sets the session limits and the contention winner and loser values for this gateway by entering the AVS command:

```
agw cnos collecta collectb secure 20 10 10
```

The Collection A system administrator sets up a CMS communications directory, SCOMDIR NAMES, entry that defines a symbolic destination name for RES4 located in Collection B:

```
:nick.inventry      :luname.collecta collectb
                    :tpn.res4
                    :modename.secure
                    :security.pgm
```

The USER1 virtual machine has the following APPCPASS directory statement in its CP directory entry that contains its user ID (REMUSER1) and password (REMPASS1) in Collection B:

```
appcpass collecta collectb remuser1 rempass1
```

While AVSb is running, its operator defines a global gateway COLLECTB to be used by APPC/VM programs in Collection A to communicate with global resources in Collection B by entering the AVS command:

```
agw activate gateway collectb global
```

After defining COLLECTB, the operator enters:

```
agw cnos collectb collecta secure 20 10 10
```

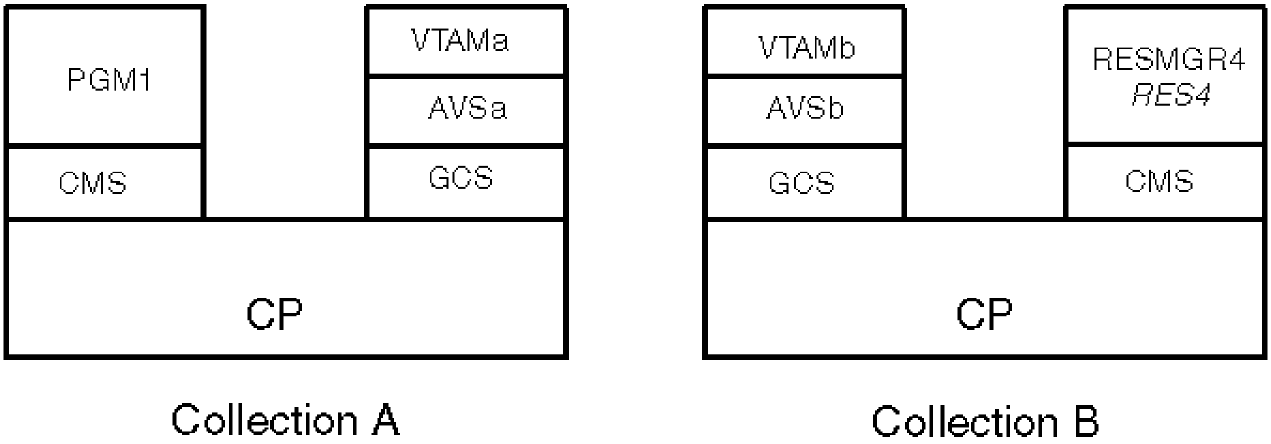


Figure 133. Defining Global Gateways

Step 2—An APPC/VM Program Requests a Connection to a Resource in Another Collection

PGM1, in Collection A, requests a connection to RES4 by specifying the symbolic destination name INVENTRY in its connection request. The symbolic destination name is resolved into a request to connect to the resource RES4 in remote LU COLLECTB through gateway COLLECTA. AVSa translates the request and sends it to VTAMa. VTAMa sends the request to the SNA network.

In Collection B, a connection request comes in through COLLECTB (a global gateway) from the network; AVSb translates the request. The user ID and password (REMUSER1 and REMPASS1) are validated in Collection B. CP then routes the connection to RESMGR4 (see Figure 134 on page 264). RESMGR4, if necessary, receives the allocation data sent by PROG1 and accepts the connection.

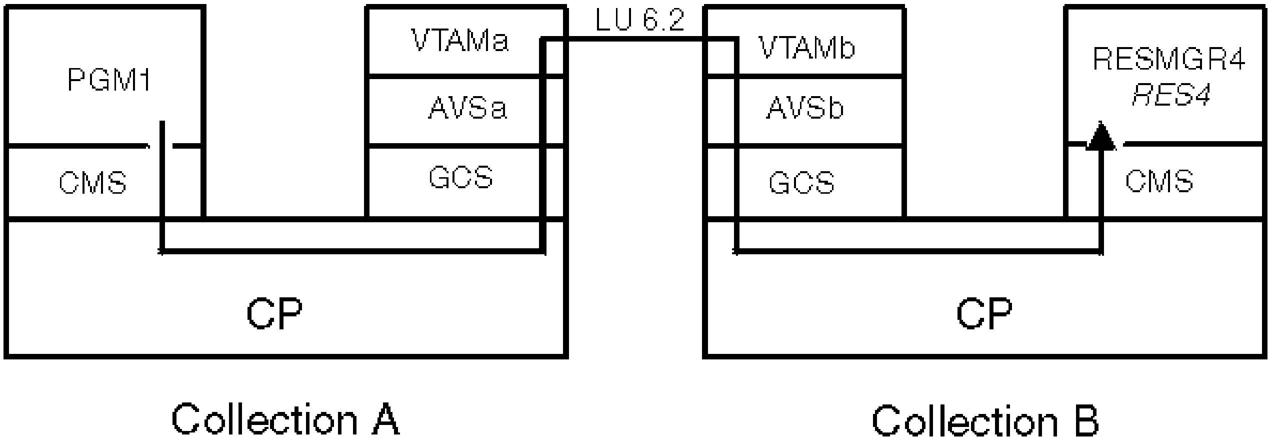


Figure 134. APPC/VM Program Requesting to Connect to a Global Resource in a Remote TSAF Collection

Step 3—An APPC/VM Program Communicates with a Resource in Another Collection

AVSa informs PGM1 that the connection to INVENTRY (RES4) is complete. PGM1 and INVENTRY exchange data using supported APPC/VM functions until one ends the conversation (see Figure 135 on page 265).

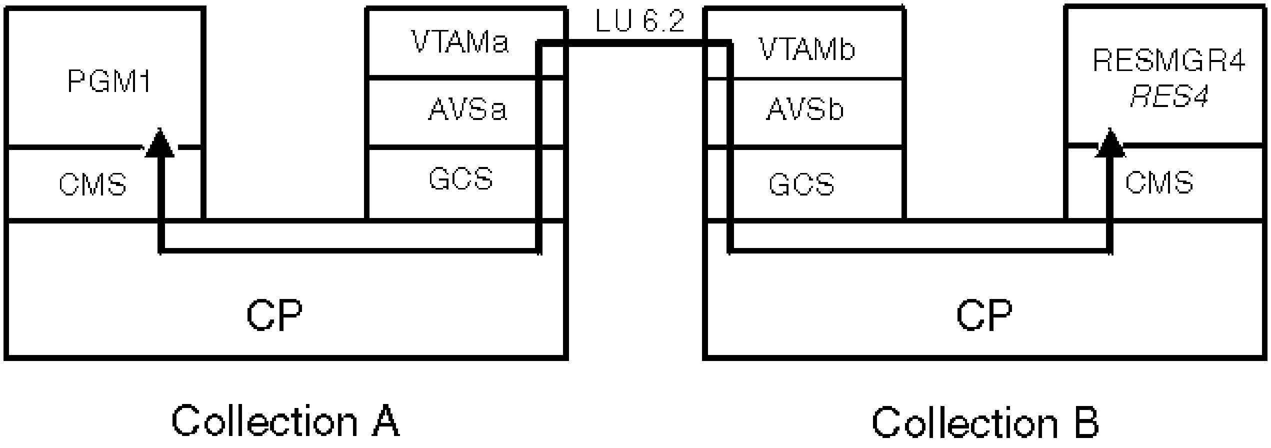


Figure 135. APPC/VM Program Communicating with a Global Resource in a Remote TSAF Collection

Outbound Connection to a Shared File System File Pool

Any user in a TSAF collection can connect to a Shared File System file pool located in another TSAF collection. This scenario involves:

- A TSAF collection, Collection A, made up of one VM system with:
  - An AVS virtual machine called AVSa
  - A VTAM virtual machine called VTAMa
  - An enrolled file pool user whose user ID in Collection A and in Collection B is TONY. The password in Collection B is PASSB.
- A TSAF collection, Collection B, made up of one VM system with:
  - An AVS virtual machine called AVSb
  - A VTAM virtual machine called VTAMb
  - A file pool server called POOLSRV that manages the file pool POOL15.

During initialization, the file pool server has automatically identified the file pool to the collection as a global resource. The file pool administrator has enrolled TONY in POOL15 and the top directory POOL15:TONY has been generated.

Note: To ensure conversation security is properly handled, you should include the SECACPT parameter on the VTAM APPL statement (see Figure 120 on page 249).

Step 1—AVS Virtual Machines Define Global Gateways

While AVSa is running, its operator defines a global gateway POOLGAT to be used by users in Collection A to access file pools in Collection B:

```
agw activate gateway poolgat global
```

The operator then sets the session limits and the contention winner and loser values for this gateway by entering:

```
agw cnos poolgat poolacc secure 20 15 5
```

The Collection A system administrator creates an entry in the SCOMDIR NAMES file to define a symbolic destination name for POOL15 located in Collection B:

```
:nick.pool15        :luname.poolgat poolacc
                    :tpn.pool15
                    :modename.secure
                    :security.same
```

While AVSb is running, its operator enters the following command to define the global gateway POOLACC to be used to communicate with file pools in Collection B:

```
agw activate gateway poolacc global
```

The operator then sets the session limits and the contention winner and loser values for this gateway by entering:

```
agw cnos poolacc poolgat secure 20 5 15
```

The system administrators have ensured that no duplicate user IDs are defined in either collection. This lets the AVSb operator enter the following command, which indicates that any user ID from Collection A can be accepted as being already-verified when SECURITY(SAME) is specified on a conversation:

```
agw add userid poolgat * =
```

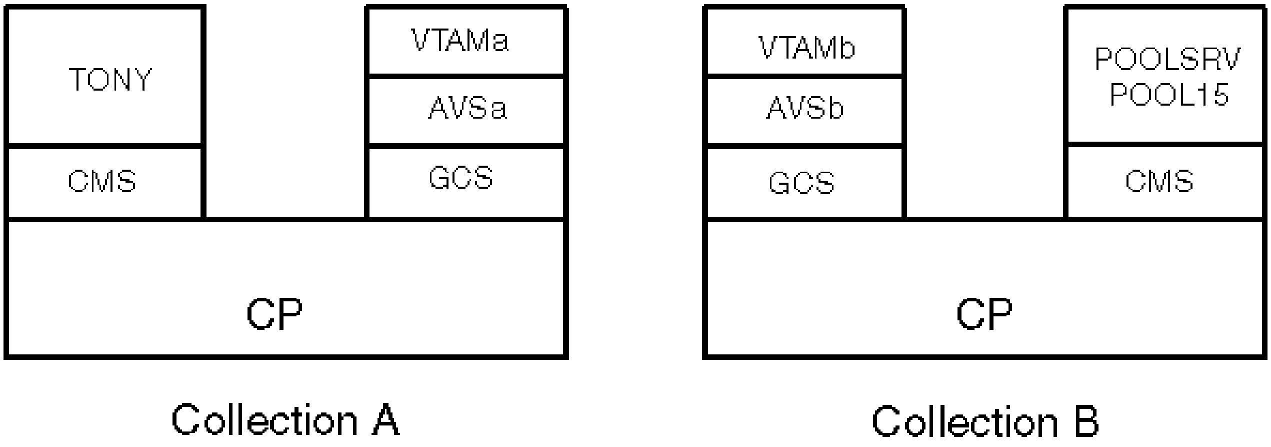


Figure 136. Defining Global Gateways

Step 2—A User Requests Access to a File Pool in Another TSAF Collection

The user TONY enters a CMS ACCESS command for his top directory, that is maintained in the file pool POOL15. In the TONY virtual machine, CMS tries to connect to the resource POOL15. The symbolic destination name is resolved into a request to connect to POOL15 in remote LU POOLACC through gateway POOLGAT. AVSa translates the request and sends it to VTAMa. VTAMa sends the request to the SNA network.

In Collection B, a connection request comes in through POOLACC (a global gateway) from the network and AVSb translates the request. The user ID and password (TONY and PASSB) are validated in Collection B. CP then routes the connection to POOLSRV. The user ID TONY is presented to POOLSRV for authorization to access POOL15. Because TONY is an authorized user, TONY is connected to file pool POOL15 (see Figure 137 on page 267).

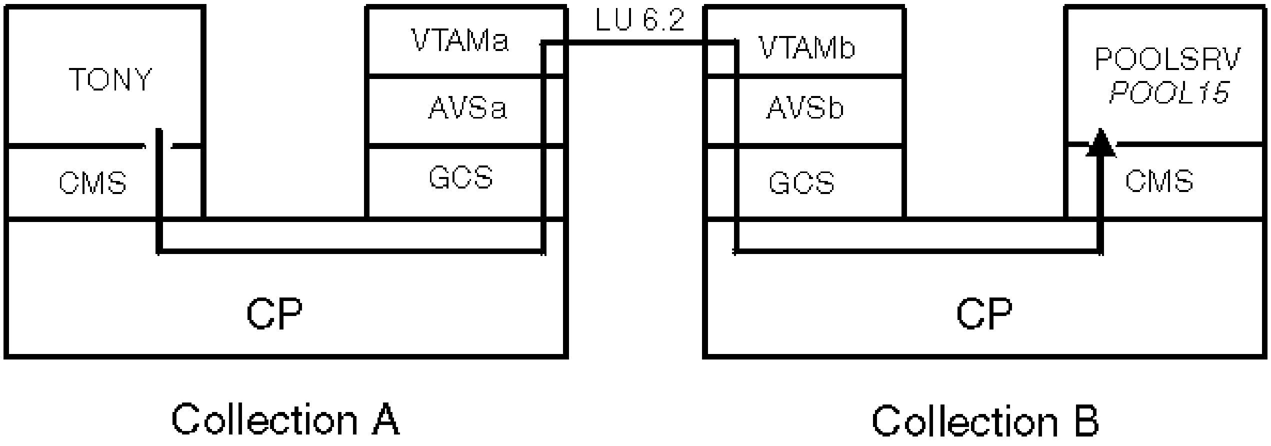


Figure 137. User Requesting Access to a File Pool in a Remote TSAF Collection

Step 3—A User Accesses a File Pool in Another TSAF Collection

AVSa informs TONY that the ACCESS command has completed. TONY uses his top directory in POOL15 (see Figure 138 on page 267).

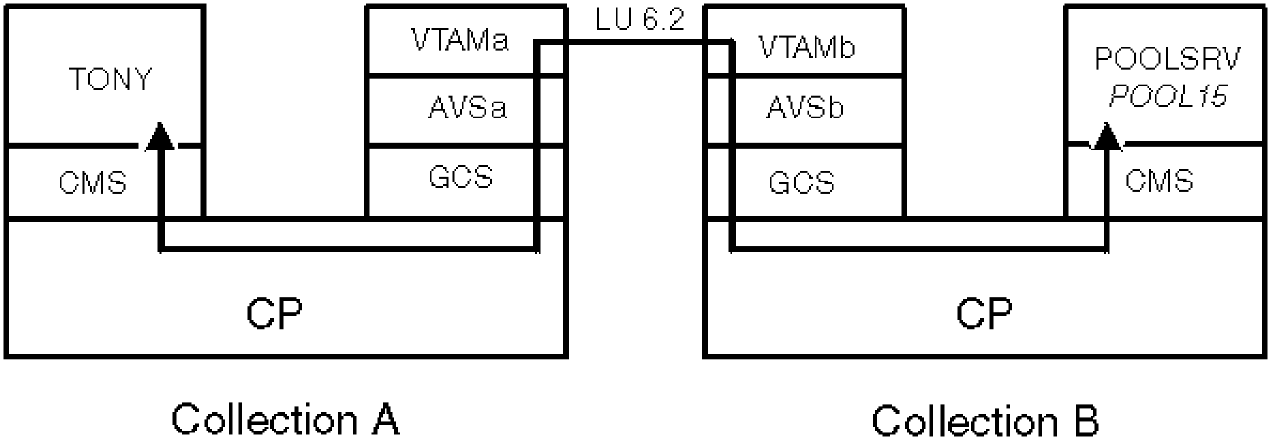


Figure 138. User Accessing a File Pool in a Remote TSAF Collection

Chapter 19. Operating the AVS Virtual Machine

This chapter describes the commands that control the AVS virtual machine.

Overview of AVS Commands

Use the following commands to run AVS and maintain communication paths. These commands must be entered from the AVS virtual machine.

Command	Function	Location
AGW ACTIVATE GATEWAY	Defines a gateway LU to VM and activates it to VTAM.	"AGW ACTIVATE GATEWAY" on page 271
AGW ADD USERID	Maps user IDs that have already been verified at the remote LU to user IDs that are valid on the local system.	"AGW ADD USERID" on page 273
AGW CNOS	Sets the session limit and contention winner/loser values between a local and remote LU.	"AGW CNOS" on page 274
AGW DEACTIVE CONV	Deactivates a conversation.	"AGW DEACTIVE CONV" on page 277
AGW DEACTIVE GATEWAY	Deactivates an active gateway.	"AGW DEACTIVE GATEWAY" on page 278
AGW DELETE USERID	Deletes remote and local user ID mappings.	"AGW DELETE USERID" on page 279
AGW QUERY	Displays status information about AVS.	"AGW QUERY" on page 280
AGW QUIESCE	Ends AVS in a nondisruptive manner.	"AGW QUIESCE" on page 285
AGW SET ETRACE	Sets external tracing on or off.	"AGW SET ETRACE" on page 286
AGW SET ITRACE	Sets internal tracing on or off.	"AGW SET ITRACE" on page 287
AGW START	Initializes AVS.	"AGW START" on page 288
AGW STOP	Ends AVS immediately.	"AGW STOP" on page 289

For information about how to read the syntax of the AVS commands, see "Syntax, Message, and Response Conventions" on page xxii.

Using Online HELP for AVS Commands

You can receive online information about AVS commands by using the z/VM HELP Facility. For example, to display a menu of the AVS commands, enter:

```
help avs menu
```

To display information about a specific AVS command (ADD USERID in this example), enter:

```
help avs add
```

You can also display information about a message by entering one of the following commands:

```
help msgid or help msg msgid
```

For example, to display information about message AGWVTC002I, you can enter one of the following commands:

```
help agwvtc002i or help agw002i or help msg agw002i
```

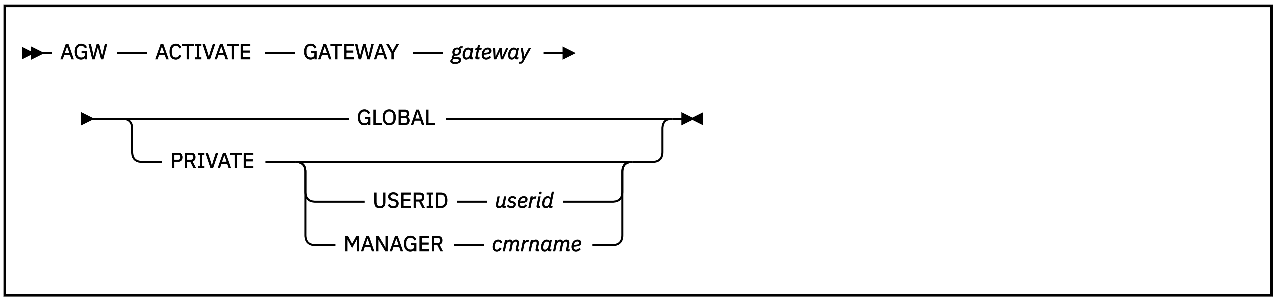
For more information about using the HELP Facility, see the *z/VM: CMS User's Guide*. To display the main HELP Task Menu, enter:

```
help
```

For more information about the HELP command, see the *z/VM: CMS Commands and Utilities Reference* or enter:

```
help cms help
```

AGW ACTIVATE GATEWAY



Purpose

Use the AGW ACTIVATE GATEWAY command to define a gateway LU in the TSAF or CS collection to z/VM and to activate it to VTAM.

The gateway LU being activated must have been previously defined to VTAM on an APPL statement. The virtual machine in which AVS is running must have the appropriate CP directory authorization to identify itself as the owner of the gateway (see "Authorizing the AVS Virtual Machine to Manage Gateways" on page 314).

Operands

gateway

is the name of the LU in the TSAF or CS collection that will be a gateway. AVS will identify itself to VTAM as the owner of this LU.

GLOBAL

indicates that programs outside the TSAF or CS collection can use the *gateway* as an LU to allocate conversations with global resources in the TSAF or CS collection.

PRIVATE

indicates that programs outside the TSAF or CS collection can use the *gateway* as an LU to allocate conversations with private resources in the TSAF or CS collection.

USERID *userid*

specifies the user ID of the private resource server virtual machine.

If you specify USERID *userid*, all inbound connections through *gateway* are routed to the server virtual machine whose user ID is *userid*. This means that *gateway* can only allocate conversations to private resources owned by the *userid* virtual machine. Only the *userid* virtual machine can allocate conversations that are outbound connections through *gateway*. AVS accepts connections with SECURITY(NONE) and the *userid* virtual machine owner needs to authorize the target private resources with a list (:list.) value of '*' in the $SERVER$ NAMES file. This type of private gateway is called a dedicated private gateway.

If you do not specify USERID *userid*, all inbound connections through *gateway* are routed to the server virtual machine whose user ID is the same as the access security user ID specified in the allocation request. This means that the gateway routes the connection to the virtual machine owned by the user ID specified on the allocation request. AVS rejects connections with SECURITY(NONE). This type of private gateway is called a nondedicated private gateway.

MANAGER *cmrname*

specifies a CMR (Conversation Management Routine) that is associated with a private gateway LU. The program *cmrname* must reside in a GCS load library that was defined by a GLOBAL LOADLIB command.

What Happens When You Enter this Command

When you enter the AGW ACTIVATE GATEWAY command, AVS identifies itself to VTAM as the owner of *gateway*, which is the name of an LU in the TSAF or CS collection. AVS also connects to *IDENT to identify itself to z/VM as the owner of this gateway.

Programs in the SNA network then view programs in the TSAF or CS collection as residing on this LU. To access programs outside the TSAF or CS collection, user programs within the TSAF or CS collection can specify this *gateway* value as the LU name qualifier of the locally known LU name. When the connection is made, the program outside the TSAF or CS collection views the program in the TSAF or CS collection as if it resided on the LU.

Global resource managers in the TSAF or CS collection should use a global gateway when they allocate conversations with resources outside the collection. Nonglobal resource managers in the TSAF or CS collection should use a private gateway when they allocate conversations with resources outside the collection.

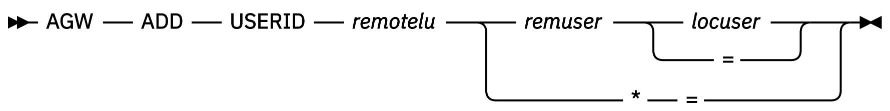
When the AGW ACTIVATE GATEWAY command completes, you receive the following message (*gateway* is the name of the gateway LU):

```
AGWVTC002I Gateway gateway is activated
```

The AGW ACTIVATE GATEWAY command will fail if you specify a gateway that has already been defined to AVS.

Before communications can begin through *gateway*, the session values for this gateway can be set by the AGW CNOS command (described in “Purpose” on page 274) or by a CNOS verb from a remote LU. However, if you enter the AGW CNOS command before the AGW ACTIVATE GATEWAY command processing is completed, its processing is deferred until the AGW ACTIVATE GATEWAY command processing is completed. Because AVS does not have an automatic restart, you must restart AVS if VTAM goes down.

AGW ADD USERID



Purpose

Use the AGW ADD USERID command to map a user ID that has already been verified from a remote LU into a user ID that is valid in your TSAF or CS collection.

If you specify a *locuser* for a *remotelu-remuser* pair that already has a *locuser* specified, the ADD USERID command fails. Use the AGW DELETE USERID command to delete the existing *remotelu-remuser* pair, then retry the ADD USERID with the new *locuser* for the *remotelu-remuser* pair.

Operands

remotelu
is the locally-known name of the remote LU where the verified remote user ID *remuser* resides.

remuser
is the already-verified remote user ID of the user on the remote LU *remotelu*.

*****
validates all user IDs that have already been verified from the remote LU. In this case, you must specify = as the *locuser* value. Wildcard characters (for example, ABC*) cannot be used in defining remote user IDs.

locuser
is the user ID by which the remote user is known within the local TSAF or CS collection. The system administrator should make sure that this user ID does not conflict with other user IDs defined within the TSAF or CS collection.

=
indicates that the local user ID is the same as the remote user ID (*remuser*). A local alias is not required for the already-verified user ID *remuser* from the remote LU *remotelu*. The local user ID *locuser* does not have to be defined in any CP user directory in the TSAF or CS collection.

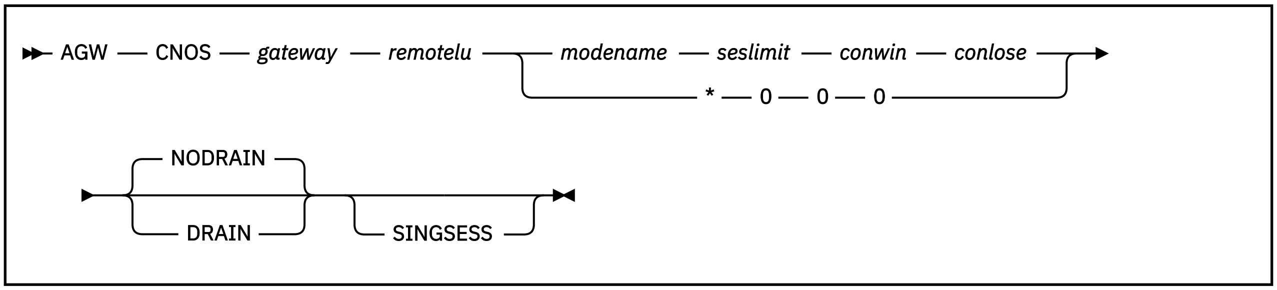
What Happens When You Enter this Command

The remote user ID, *remuser*, from the specified remote LU, *remotelu*, is mapped to a local user ID, *locuser*. The mapping is stored in the AVS user ID table. If AVS receives an already-verified user ID in the FMH5 header from an inbound connection, it looks for a mapping in the AVS user ID table; this means SECURITY(SAME) was specified on the connection request. The local user ID is used on the local system for this conversation.

The following example specifies a different local user ID for a specific remote user ID. From the following commands, the local LU accepts most remote user IDs from LU1 and the local user ID is the same as the remote ID. However, the remote ID FRED is accepted using the local user ID BILL.

```
agw add userid lu1 fred bill
agw add userid lu1 * =
```

AGW CNOS



Purpose

Use the AGW CNOS (Change Number of Sessions) command to set values for communications between a gateway (local LU) and a remote LU. Use the AGW CNOS command to:

- Set the initial session limit, contention winner, and contention loser values for a newly-activated gateway
- Change the existing session limit, contention winner, and contention loser values for an activated gateway
- Change the existing session limit, contention winner, and contention loser values to zeros to quiesce communications over an activated gateway.

Operands

gateway

specifies the name of the local LU that is the gateway. This name is identified by the AGW ACTIVATE GATEWAY command (see "Purpose" on page 271).

remotelu

is the name of the remote LU for which the session limits are set. If the *seslimit* value is greater than zero, the local gateway LU, *gateway*, will accept conversation allocation requests from the remote LU.

modename

is the logon mode name for which the session limit and contention winner and loser values are to be changed. The mode name, which identifies the characteristics of the sessions, must be defined in the logon mode table specified by the MODETAB parameter on the VTAM APPL statement for the *gateway*. See Figure 120 on page 249 and the *VTAM Resource Definition Reference* for more information.

Note: The mode name SNASVCMG is reserved for use by VTAM and should not be specified on the AGW CNOS command.

*****

indicates that the CNOS values apply to all mode names defined for the *gateway* and *remotelu* pair. If you specify *, you must set the *seslimit*, *conwin*, and *conlose* values to 0.

seslimit

is the maximum number of LU-to-LU sessions allowed between the gateway LU *gateway* and the remote LU *remotelu*. These sessions will have the characteristics defined by the specified logon mode name *modename*.

The *seslimit* may be a decimal value between 0 and 32,767. For single-session connections, *seslimit* should be 0 or 1.

If the *seslimit* value is greater than zero, the remote LU can negotiate this field to a value greater than zero and less than the specified session limit *seslimit*. If the *seslimit* value is negotiated, that value becomes the new session limit.

If you specify 0 as the *seslimit* value for an active gateway, activities through the gateway will quiesce. The remote LU cannot negotiate the session limit value.

If you specify * as the *modename*, you must specify 0 for the *seslimit* value.

conwin

specifies the number of parallel sessions for which the gateway LU is guaranteed to be the contention winner. The *conwin* value must be a decimal number between 0 and the specified *seslimit* value. The sum of *conwin* and *conlose* cannot exceed the *seslimit*.

If you specify * as the *modename*, you must specify 0 for the *conwin* value.

If the *conwin* value is greater than half of the session limit value (rounded downward), the remote LU can negotiate the contention winner value. The resulting *conwin* value will be greater than, or equal to, half the *seslimit*; this *conwin* value will also be smaller than the original specified contention winner value. The negotiated value becomes the new *conwin* value for this gateway LU.

If the specified *conwin* value is less than, or equal to, half of the *seslimit* value, the remote LU cannot negotiate the contention winner value.

conlose

specifies the number of parallel sessions of which the remote LU *remotelu* is guaranteed to be the contention winner. (In this case, the gateway LU is considered the contention loser.) The *conlose* value must be a decimal number between 0 and the specified session limit, *seslimit*. The sum of *conwin* and *conlose* cannot exceed *seslimit*.

The remote LU can negotiate the *conlose* value to be less than, or equal to, the *seslimit* value minus the gateway *conwin* value. If *conlose* is negotiated, it becomes the new minimum number of remote LU contention winners.

If you specify * as the *modename*, you must specify 0 for the *conlose* value.

NODRAIN

specifies that AVS should not satisfy currently pending allocation requests before deactivating sessions; NODRAIN is the default. NODRAIN is only valid when you specify a *seslimit* value of zero to quiesce an active gateway. If the current *seslimit* is not zero, VTAM rejects this operand.

DRAIN

specifies that AVS should satisfy currently pending allocation requests before deactivating sessions. DRAIN is only valid when you specify a *seslimit* value of zero to quiesce an active gateway. If the current *seslimit* is not zero, VTAM rejects this operand.

SINGSESS

specifies that the remote LU *remotelu* a single-session LU; it does not support parallel sessions. Some workstations, for example, can be single-session LUs. (Protected conversations are not allowed with single session partners.)

What Happens When You Enter this Command

When you enter the AGW CNOS command, AVS issues the VTAM macro APPCCMD CONTROL=OPRCNTL,QUALIFY=CNOS to set limits and contention winner and loser values. AVS then issues the VTAM macro APPCCMD CONTROL=OPRCNTL,QUALIFY=DEFINE with the same values specified on the AGW CNOS command. This macro sets the values VTAM uses to negotiate CNOS commands sent from remote LUs. See *VTAM Programming for LU 6.2* for more information about the APPCCMD macro and CNOS defaults.

When you enter AGW CNOS for the first time for a gateway or if you specify a *seslimit* value that is larger than the previous value specified for an activated gateway, the remote LU can satisfy queued allocation requests as sessions become available. Allocation requests issued **after** the AGW CNOS command is issued are added to the queue.

If you enter the AGW CNOS command to decrease the *seslimit* value for an active gateway, any existing conversations using that gateway are not disrupted. As each conversation ends, however, the session is not reused. This process continues until the lower *seslimit* is reached. The remote LU will then satisfy any queued allocation requests when sessions become available.

To quiesce an active gateway, specify a *seslimit* value of zero. Existing conversations through the gateway are not disrupted. If you specify the DRAIN operand, any queued requests are satisfied as sessions become available. However, no additional allocation requests are added to the queue. If you specify NODRAIN, no queued requests are satisfied.

If you enter AGW CNOS before the gateway has been activated, the AGW CNOS command is deferred until the AGW ACTIVATE GATEWAY command completes.

When the AGW CNOS command completes successfully, you receive the following message. The message displays the values specified on AGW CNOS and the current, negotiated values.

```
CNOS for gateway remotelu modename has been completed
CURRENT VALUES:      seslimit   conwin   conlose
CNOS COMMAND VALUES: seslimit   conwin   conlose
```

In addition, the local gateway LU can receive connections with any level of security from the remote LU, regardless of the security acceptance specified in the LU's APPL statement. AVS CNOS sets a default security level of ALREADYV. To ensure that the correct level of security is allowed, you should specify the appropriate SECACPT parameter on the AVS APPL statement in VTAM.

The AGW CNOS command fails if you specify a remote LU that is not active. However, when the remote LU is activated, it may send a CNOS verb to your gateway LU; this CNOS verb may specify a session limit value. If this session limit value exceeds the *seslimit* you specified on the AGW CNOS command, AVS reissues the VTAM macro APPCCMD with QUALIFY=CNOS. This ensures that the *seslimit* value that you specified is used. AVS also issues APPCCMD with QUALIFY=DEFINE to ensure that any negotiated session limit values do not exceed that *seslimit* value. If the remote LU does not issue a CNOS verb when it becomes activated, AVS will issue the AGW CNOS command when it receives an APPC/VM connection request for that remote LU. AVS will use the default session limits specified on the APPL statement.

AGW DEACTIVE CONV

```
▶▶─ AGW ── DEACTIVE ── CONV ── gateway ── convid ─▶◀
```

Purpose

Use the AGW DEACTIVE CONV command to deactivate a conversation on a gateway.

Operands

gateway
is the name of the gateway, a local LU, through which the conversation to be deactivated is established.

convid
is the VTAM conversation ID of the conversation to be deactivated. To find the VTAM conversation ID, enter the AGW QUERY CONV command (described in "Purpose" on page 280).

What Happens When You Enter this Command

The conversation immediately ends. The VM side of the conversation is severed and the VTAM side is deallocated. You should use this command when a conversation does not stop because an error has occurred or if you need to end a long conversation because you need to bring the gateway down.

AGW DEACTIVE GATEWAY

```
►► AGW ── DEACTIVE ── GATEWAY ── gateway ──┬───────┬──►◄
                                           └ FORCE ┘
```

Purpose

Use the AGW DEACTIVE GATEWAY command to deactivate an activated gateway. The gateway will deactivate when all existing conversations through it have completed.

Operands

gateway

specifies the gateway, a local LU, that AVS deactivates.

FORCE

immediately deactivates all active conversations through this gateway.

If you do not specify FORCE, all existing conversations through this gateway are allowed to complete, but no new conversations are established.

What Happens When You Enter this Command

The gateway is deactivated when all existing conversations through it have completed. AVS severs its connection to *IDENT for the specified gateway. When the command completes, AVS issues the following message (*gateway* identifies the gateway LU):

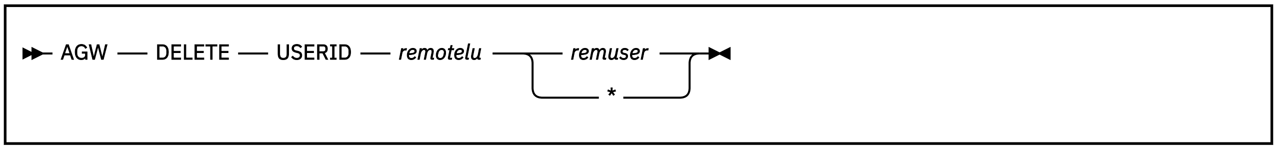
```
AGWVTC003I Gateway gateway is deactivated
```

You can also enter the AGW DEACTIVE CONV command (see "Purpose" on page 277) to deactivate conversations before entering AGW DEACTIVE GATEWAY.

If the *gateway* you specified has not been defined to AVS by an AGW ACTIVATE GATEWAY command, the AGW DEACTIVE GATEWAY command will fail.

You cannot reactivate a gateway that has any remaining active conversations. You must wait until all the conversations end on the gateway before you enter the AGW ACTIVATE GATEWAY command.

AGW DELETE USERID



Purpose

Use the AGW DELETE USERID command to remove a previously added user ID mapping. The AGW DELETE USERID command fails if the specified pair had not been previously defined to AVS.

Operands

remotelu
is the name of the remote LU where the remote user ID, *remuser,* resides.

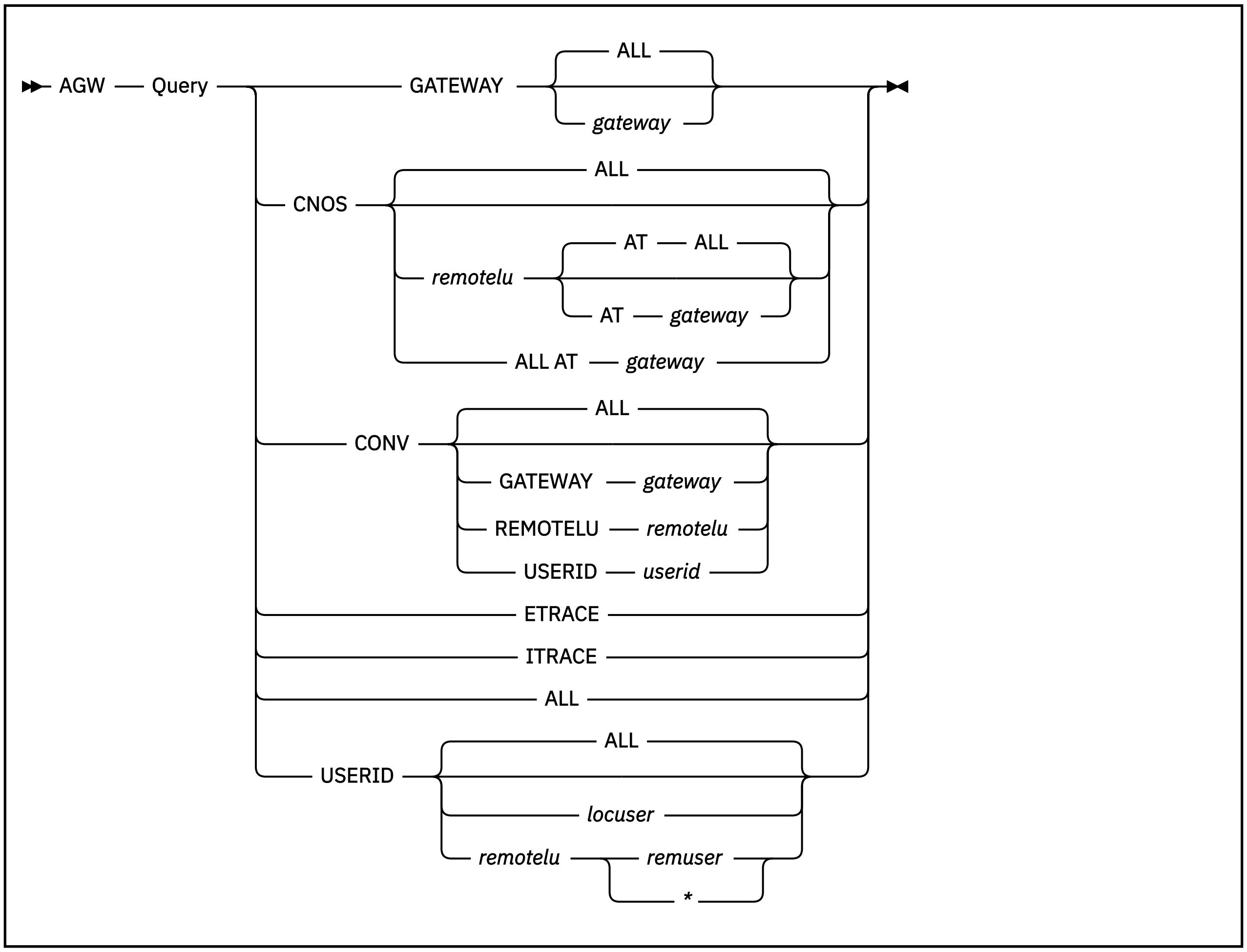
remuser
is the already-verified remote user ID of the user on the remote LU whose mapping is deleted from the user ID table.

*****
deletes all mappings that correspond to the remote LU, *remotelu*.

What Happens When You Enter this Command

The mapping for the specified remote LU, *remotelu*, remote user ID, *remuser*, pair are removed from the AVS user ID table. Future inbound connections from *remotelu* that specify this remote user ID as an already-verified user ID will fail. You can add the pair back to the AVS user ID table with the AGW ADD USERID command.

AGW QUERY



Purpose

Use the AGW QUERY command to display information about various settings and conditions of AVS.

Operands

GATEWAY
displays the status of the gateways.

ALL
displays information for all gateways at this AVS virtual machine; ALL is the default.

gateway
displays information for the specified gateway.

CNOS
displays the status of session limits and contention polarity.

ALL
displays information about all remote LUs being accessed through all gateways; ALL is the default.

remotelu
is the name of a specific remote LU.

AT *gateway*
is a specific gateway through which the *remotelu* is accessed.

AT ALL
displays information about all gateways through which the *remotelu* is accessed; AT ALL is the default.

ALL AT *gateway*
displays information about all remote LUs being accessed through the specified gateway.

CONV
displays the status of conversations in the gateways.

ALL
displays information about all active conversations; ALL is the default. Because this operand can generate a lot of output, you may want to issue the AGW QUERY CNOS first to determine how many conversations are active.

GATEWAY *gateway*
displays information about the active conversations through the specified gateway.

REMOTELU *remotelu*
displays information about the active conversations to the specified remote LU.

USERID *userid*
displays information about active conversations with the specified user ID or specified access user ID.

ETRACE
displays the current setting of external tracing.

ITRACE
displays the current setting of internal tracing.

ALL
displays all of the current information about the gateways, CNOS settings, conversations, trace settings, and user ID mappings.

USERID
displays mapping information about local user IDs.

ALL
displays information about all *remotelu*, *remuser*, and *locuser* mappings; ALL is the default.

locuser
displays mapping information about this local user ID.

remotelu remuser
displays the mapping for this *remotelu*-*remuser* pair.

remotelu *
displays all the mapping information that about this remote LU.

What Happens When You Enter this Command

AGW QUERY command responses vary depending on the operands you specify.

AGW QUERY GATEWAY: If you enter AGW QUERY GATEWAY, AVS displays the status of all gateways. You can get the following response for each gateway:

```
GATEWAY = gateway  type      status                  CONV COUNT = n
                  [ USERID = userid ]
                  [ MANAGER = cmrname ]
```

gateway
Gateway LU name

type
PRIVATE or GLOBAL

status
ACTIVE, ACTIVATE IN PROGRESS, or DEACTIVATE IN PROGRESS

n
Decimal number of active conversations through the specified gateway.

For private gateways, AGW QUERY also displays the USERID or MANAGER settings. If the USERID setting is displayed, the value of *userid* is the user ID defined in this gateway's AGW ACTIVATE GATEWAY command. If the MANAGER setting is displayed, the *cmrname* value identifies the Conversation Manager Routine (CMR) that is defined in this gateway's AGW ACTIVATE GATEWAY command.

AGW QUERY CNOS: If you enter AGW QUERY CNOS, AVS displays the status of the session limits and contention polarity. You can get the following response for each CNOS command entered:

```
GATEWAY = gateway   REMOTELU = remotelu   MODE = modename
                             status             CONV COUNT = n
CURRENT VALUES:       seslimit    conwin   conlose   [ DRAIN = setting ]
CNOS COMMAND VALUES:  seslimit    conwin   conlose   [ DRAIN = setting ]
```

gateway
Gateway LU name

remotelu
The remote LU name

modename
Mode name for this conversation

status
ACTIVE, QUIESCED, or CNOS IN PROGRESS

n
Decimal number of active conversations using the specified gateway.

seslimit
Maximum number of sessions allowed

conwin
Contention winner values

conlose
Contention loser values

setting
YES or NO; If the DRAIN setting is not displayed, the CNOS command was issued at the remote LU.

AGW QUERY CONV: If you enter AGW QUERY CONV, AVS displays the status of conversations in a gateway. You receive the following information for each conversation:

```
GATEWAY = gateway, REMOTELU = remotelu, RESOURCE = tpn
USERID   = userid, ACCESS USERID = accuser, CONVERSATION = convid
              VTAM STATE = n, VM STATE = n
```

gateway
Gateway LU name

remotelu
The remote LU name

tpn
Name of the transaction program (the communications partner)

userid
The user ID from the allocate or connect request; this value is not present for inbound conversations.

accuser
The access user ID specified on the allocate or connect request.

convid
The VTAM conversation ID (specified in hexadecimal).

n

One of the following hexadecimal numbers, which indicate the status of the VTAM or VM side of the path:

Value	Meaning
X'01'	No state
X'02'	Connection pending
X'03'	Connection in progress
X'04'	Send state
X'05'	Receive state
X'06'	Confirm state
X'07'	Sever state
X'08'	Allocate pending
X'09'	Confirm send state
X'0A'	Confirm deallocation state
X'0B'	Queued allocate state
X'0C'	Reserved
X'0D'	Prepare received
X'0E'	Solicited request for communications received
X'0F'	Unsolicited request for communications received
X'10'	Committed received
X'11'	Backout received
X'12'	Backout required
X'13'	Sever abend required
X'14'	CRR recovery server queued allocate

If you enter the AGW QUIESCE command and it has not completed, enter AGW QUERY CONV command to determine which conversations are still active. You can then enter AGW DEACTIVE CONV (see "Purpose" on page 277) to deactivate any remaining active conversations.

AGW QUERY ETRACE: If you enter AGW QUERY ETRACE, AVS displays the external trace option setting in one of the following responses:

```
ETRACE ON
ETRACE OFF
```

AGW QUERY ITRACE: If you enter AGW QUERY ITRACE, AVS displays one of the following responses to indicate the internal trace option setting:

```
ITRACE OFF
ITRACE ON ALL
ITRACE ON GATEWAY:  gateway
```

AGW QUERY USERID: If you enter AGW QUERY USERID, AVS displays the contents of the AVS user ID table that contains the remote LU-remote user ID mappings. When the user ID is known, the remote LU, remote user ID and local user ID mappings are displayed in the following format:

```
REMOTELU = remotelu REMOTE USERID = remuser
LOCAL USERID = locuser
```

remotelu
: The remote LU name

remuser
: The remote user ID

locuser
: The local user ID

If you had previously entered an `AGW ADD USERID` *`remotelu`* `*` = command, you get the following response from AGW QUERY USERID:

```
ALL ALREADY VERIFIED USERIDS FROM REMOTE LU remotelu ARE ACCEPTED
```

AGW QUIESCE

```
►►─ AGW ── QUIESCE ─►◄
```

Purpose

Use the AGW QUIESCE command to end AVS activities after the last existing conversation has completed.

Use the AGW STOP command (see "Purpose" on page 289) to quickly end AVS activities.

What Happens When You Enter this Command

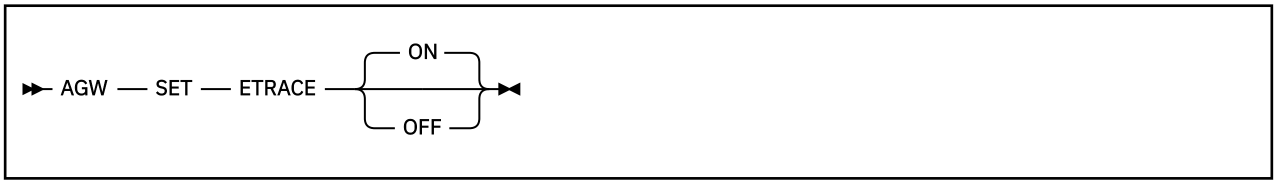
After you enter the AGW QUIESCE command, AVS issues the following message:

```
AGWCMJ017I Quiesce is in progress.
```

AGW QUERY and AGW STOP are the only AVS commands that are accepted after the AGW QUIESCE command is entered. AVS does not accept any new inbound or outbound connection requests. Existing conversations through AVS are allowed to complete. After the last existing conversation ends, AVS issues the following messages (if accounting is enabled) and stops all activity:

```
AGWACS502W AVS Accounting has ended.
AGWDSP201I AVS termination completed.
```

AGW SET ETRACE



Purpose

Use the AGW SET ETRACE command to enable or disable external tracing. External tracing is only in effect if internal tracing is set on. The type of tracing done externally is the same type of tracing done internally. Internal tracing is set by the AGW SET ITRACE command (described in"Purpose" on page 287).

Note: Before external tracing entries can be written, an operator (privilege class C) must enter a TRSOURCE command followed by the AGW SET ITRACE and ETRACE GTRACE commands. See *z/VM: CP Commands and Utilities Reference* for information on the TRSOURCE command.

Operands

ON
causes AVS to write the internal AVS trace records set with the AGW SET ITRACE command externally to the TRSOURCE file. This is the default.

OFF
causes no external tracing to be done.

What Happens When You Enter this Command

If you enter AGW SET ETRACE ON, AVS starts to write trace records to a TRSOURCE spool file, and you get this message:

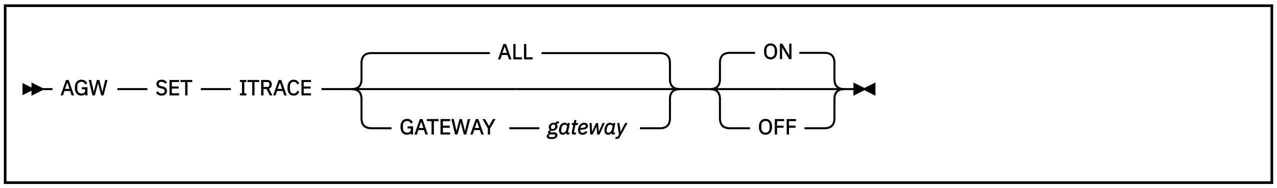
```
AGWCMH005I  External trace started
```

If you enter AGW SET ETRACE OFF, AVS stops external tracing and issues this message:

```
AGWCMH006I  External trace ended
```

While the performance of AVS is better if both internal tracing and external tracing are OFF, it will be difficult to service any errors encountered if both types of tracing are OFF.

AGW SET ITRACE



Purpose

Use the AGW SET ITRACE command to enable or disable internal tracing. (Internal tracing is always enabled during AVS initialization.) To enable external tracing, internal tracing must be set on. The type of external tracing set is the same as the type of internal tracing.

Operands

ALL
causes AVS to select trace records of certain events for all gateways; ALL is the default.

GATEWAY *gateway*
causes AVS to select trace records of certain events for the gateway *gateway*.

ON
causes AVS to write the selected trace records of certain events in AVS to the AVS internal trace table; ON is the default.

OFF
causes AVS to stop writing the selected trace records to the AVS internal trace table.

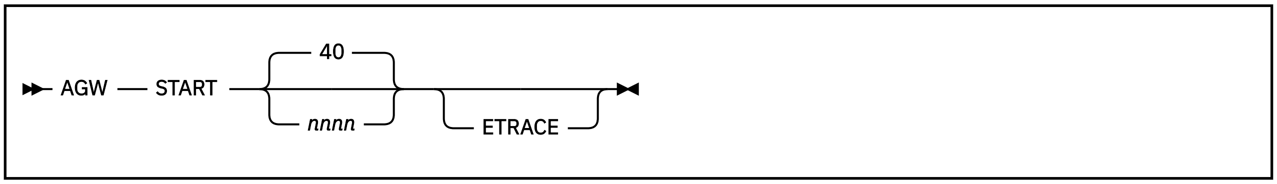
What Happens When You Enter this Command

If you enter AGW SET ITRACE, selected trace records for all gateways are written in the AVS internal trace table.

When you enter AGW START, internal tracing is set as if you entered an AGW SET ITRACE ALL ON command. Before you trace a specific gateway, enter an AGW SET ITRACE OFF command to ensure that the default ALL ON setting is not in effect. You can then enter the AGW SET ITRACE GATEWAY *gateway* command.

While the performance of AVS is better if both internal tracing and external tracing are OFF, it is difficult to service any errors encountered if internal tracing is OFF.

AGW START



Purpose

Use the AGW START command to start AVS and to enable external tracing. (Internal tracing is always enabled during AVS initialization.)

Operands

40
reserves 40 1KB blocks of virtual storage for the AVS internal trace table; this is the default.

nnnn
is the number of 1KB blocks of virtual storage that the AVS internal trace table uses. You may specify an *nnnn* value between 1 and 1000; if *nnnn* is not divisible by 4, AVS rounds up the value to the next 4K boundary.

ETRACE
enables external tracing. AVS externally writes certain AVS trace records to a TRSOURCE spool file. You must specify this parameter to get an external trace during AVS initialization.

If you do not specify ETRACE, external tracing is off and AVS writes trace records only to the AVS internal trace table, which is with the AVS virtual machine's virtual storage, until an AGW SET ITRACE OFF command is entered.

See the AGW SET ETRACE command (see "Purpose" on page 286) for the requirements that must be met before external tracing is in effect.

What Happens When You Enter this Command

After AGW START has completed, AVS is initialized and is ready to work. If there is an AGWPROF GCS exec (see "Creating the AGWPROF GCS Exec" on page 253), all of the commands contained in it will run. When the command completes, AVS issues the following message:

```
AGWINI001I AVS initialization is complete. The service level is nnnn.
```

After AVS initializes, you can disable internal tracing by entering the AGW SET ITRACE OFF command (described in "Purpose" on page 287).

AGW STOP

```
►► AGW ── STOP ►◄
```

Purpose

Use the AGW STOP command to immediately end AVS. All existing conversations through AVS are deactivated.

Use this command when AVS activities must be quickly ended. In most cases, however, you should enter AGW QUIESCE (see "Purpose" on page 285) to end AVS activities.

What Happens When You Enter this Command

When you enter AGW STOP, all AVS activities end. GCS then severs all existing APPC/VM paths and releases all resources that were in use (for example, free storage and open files). When VTAM is notified that AVS has stopped, it deallocates any outstanding conversations.

When the command completes, the following messages are issued (assuming accounting is enabled):

```
AGWACS502W AVS Accounting has ended.
AGWDSP201I AVS termination completed.
```

If VTAM stops quickly, AVS is also stopped; if VTAM stops slowly, AVS is quiesced.

Chapter 20. Generating AVS Accounting Information

PI

AVS generates three types of accounting records:

Initialization
Provides information about when AVS started

Conversation
Provides information about conversations established through the AVS virtual machine.

Termination
Provides information about when AVS ended.

To generate AVS accounting records, you must add the OPTION ACCT directory control statement to the CP directory entry of the AVS virtual machine (see "Updating the AVS Virtual Machine's CP Directory Entry" on page 246). During AVS initialization, you will receive the following message if you specify OPTION ACCT:

```
AGWACF500I  AVS Accounting records are being created
```

CP will send the accounting information to the virtual machine specified on the ACCOUNT1 parameter of the SYSTEM_USERIDS statement in the system configuration file. See *z/VM: CP Planning and Administration* for more information.

If you do not specify OPTION ACCT, you receive the message:

```
AGWACF501I  AVS Accounting was not started
```

When accounting is stopped because there was an error generating accounting records or AVS has stopped, you get the message:

```
AGWACS502I  AVS Accounting has ended
```

Initialization Accounting Record

AVS produces the initialization accounting record during its initialization. This record, and the AVS termination record, indicates when AVS was active.

Column	Contents
1 - 8	CP-provided user ID of the AVS virtual machine
9 - 12	Initialization record identifier, AGWI
13 - 17	Zeros
17 - 28	Date and time when the accounting record was generated, 'MMDDYYHHMMSS' (month, day, year, hour, minutes, seconds)
29 - 78	Reserved for IBM use
79 - 80	'C0' identifies the accounting record code that CP provides using DIAGNOSE code X'4C'

Conversation Accounting Records

AVS creates the following types of conversation accounting records:

Start (AGWS)
Identifies the resources, the users, and the time that a conversation started

Active (AGWA)
Number of bytes sent and received since the last accounting record

End (AGWE)
Number of bytes sent and received since the last conversation active accounting record was created.

Conversation Start Accounting Record

AVS produces the conversation start accounting record when a conversation through a gateway begins.

Column	Contents
1 - 8	CP-provided user ID of the AVS virtual machine
9 - 12	Conversation start accounting record identifier, AGWS
13 - 16	VTAM conversation ID
17 - 28	Date and time when the accounting record was generated, ‘MMDDYYHHMMSS’ (month, day, year, hour, minutes, seconds)
29 - 36	User ID of the user that initiated the conversation This field contains zeros on the remote (receiving) end of the conversation if SECURITY(NONE) is specified. This field contains the access security user ID on the remote (receiving) end of the conversation if SECURITY(PGM) is specified.
37 - 44	Local LU name being used by the conversation
45 - 52	Remote LU name being used by the conversation
53 - 60	Mode name
61 - 68	Transaction program name specified to start this conversation
69 - 78	Reserved for IBM use
79 - 80	‘C0’ identifies the accounting record code that CP provides using DIAGNOSE code X'4C'

Conversation Active and Conversation End Accounting Records

The conversation active accounting record is created only when one of the following events occur:

- A user specified time interval elapses
- The bytes received counter overflows
- The bytes sent counter overflows.

Therefore, some conversations may only have a conversation start and end accounting records recorded. **When the conversation active accounting record is created, the bytes received and bytes sent counters are reset to zero.**

The conversation end accounting record is created when a conversation ends.

Column	Contents
1 - 8	CP-provided user ID of the AVS virtual machine
9 - 12	Conversation active accounting record identifier (AGWA) or conversation end accounting record identifier (AGWE)
13 - 16	VTAM conversation ID

Column	Contents
17 - 28	Date and time when the accounting record was generated, ‘MMDDYYHHMMSS’ (month, day, year, hour, minutes, seconds)
29 - 32	Number of bytes received from VTAM since the conversation started or since the last conversation active accounting record was issued
33 - 36	Number of bytes sent to VTAM since the conversation started or since the last conversation active accounting record was issued
37 - 78	Reserved for IBM use
79 - 80	‘C0’ identifies the accounting record code that CP provides using DIAGNOSE code X'4C'

Termination Accounting Record

AVS produces the termination accounting record when you stop the AVS virtual machine and in some types of abnormal termination. When AVS writes the AVS termination accounting record, all applicable conversation accounting records have also been written. However, if AVS abnormally terminates without writing the termination accounting record, some conversation accounting records may be lost.

Column	Contents
1 - 8	CP-provided user ID of the AVS virtual machine
9 - 12	Termination record identifier, AGWT
13 - 16	Zeros
17 - 28	Date and time when the accounting record was generated, ‘MMDDYYHHMMSS’ (month, day, year, hour, minutes, seconds)
29 - 78	Reserved for IBM use
79 - 80	‘C0’ identifies the accounting record code that CP provides using DIAGNOSE code X'4C'

PI end

Chapter 21. Collecting AVS Problem Diagnosis Information

If a problem occurs with AVS virtual machine, you can use AVS dumps, system trace data, and interactive queries to diagnose the cause of the problem.

Note: The AVS operator does not necessarily diagnose problems, especially from the AVS virtual machine. Dumps and system trace data are usually used by the system programmer or whoever is responsible for diagnosing system problems.

Collecting Abend Information

When an AVS abend occurs, perform the following steps:

1. Collect information about the error.
   - Save the console sheet or spooled console output from the AVS virtual machine.
   - Save and process any dumps that AVS produces, using Dump Viewing Facility commands (see "Using AVS Dumps to Diagnose Problems" on page 295).
   - Save any TRSOURCE file that contains AVS data (see "Using System Trace Data to Diagnose Problems" on page 296).
2. Collect system status information. The following information can help you better determine problems:
   - The system load at the time of the failure on any systems using AVS and the status of each system (for example, did another system abend?).
   - The types of applications using AVS at the time and any information about them.
   - The physical configuration of the systems in use.
3. Recover from the abend to continue processing.

   When an abend occurs in AVS, either because AVS issued an abend or because an AVS or GCS operation caused a program exception, AVS produces a dump using the VMDUMP command (described in the *z/VM: CP Commands and Utilities Reference*).

z/VM: Other Components Messages and Codes lists AVS abend codes. See *z/VM: Diagnosis Guide* for more information about diagnosing problems in AVS.

Using AVS Dumps to Diagnose Problems

Dump Viewing Facility is a dump analysis and problem tracking tool. Use Dump Viewing Facility commands to collect and diagnose problem data for the AVS virtual machine. Dump Viewing Facility must have access to the AVS message repository that resides in the object code file pool (VMPSFS:MAINT*vrm*.AVS.OBJECT) or on the AVS 191 minidisk. Because AVS runs in a GCS group, when a dump is processed, the GCS subcommands for the DUMPSCAN command can be entered as well as the AVS subcommands for the DUMPSCAN command.

The steps involved in using dumps to diagnose problems are:

1. Obtain a GCS Dump Viewing Facility map, if it does not already exist, using the Dump Viewing Facility MAP GCS command.
2. Create the AVS dump using the following command:

   ```
   gdump 0-end format avs dss
   ```

3. Process the AVS dump, using the Dump Viewing Facility DUMPLOAD command.
4. Diagnose the AVS dump by doing the following steps:

a. Look at the symptom record.

b. Use the GDISPLAY subcommand of the Dump Viewing Facility DUMPSCAN to display dump information.

c. Format and display trace records, using the TRACE subcommand of DUMPSCAN.

See the *z/VM: Diagnosis Guide* for a description of using AVS dumps. See the *z/VM: Dump Viewing Facility* for a description of the Dump Viewing Facility.

Using System Trace Data to Diagnose Problems

AVS maintains an internal trace table within the AVS virtual machine. Use the DUMPSCAN TRACE subcommand to display the internal trace table entries in a dump. AVS also writes trace entries to the system TRSOURCE file. Use the DUMPSCAN command to view the TRSOURCE file.

To internally or externally trace AVS events, you must turn on internal tracing. To turn on internal tracing, use the AGW SET ITRACE command (described in "Purpose" on page 287). Internal tracing can be done on a gateway basis. Internal tracing information is written to an internal trap table in the AVS virtual machine. If you want to internally collect AVS trace records, enter the following command from the AVS virtual machine before TRSOURCE is started:

```
agw set itrace on
```

The AGW SET ETRACE command (see "Purpose" on page 286) lets you enable or disable external tracing for the AVS virtual machine. External tracing is not in effect unless you have also set internal tracing on. The type of external tracing you receive is the same as the type of internal tracing you requested. If you want to collect AVS trace records, enter the following command from the AVS virtual machine after TRSOURCE is started:

```
etrace gtrace
agw set etrace on
```

When you have set external tracing on (and internal tracing), AVS trace records are written externally to a TRSOURCE spool file.

The TRSOURCE command collects AVS information in a reader file. QUERY TRSOURCE, which is a privilege class C command, displays information about the current TRSOURCE setting. This information helps with problem determination. For more information about these CP commands, see the *z/VM: CP Commands and Utilities Reference*.

To access the TRSOURCE file and review the entries contained in that file, use the DUMPSCAN command. For more information about the DUMPSCAN command, see the *z/VM: Dump Viewing Facility*.

Using Interactive Queries

When the AVS virtual machine is running, you can enter the AGW QUERY command (see "AGW QUERY" on page 280). You can enter the following AGW QUERY commands at the AVS virtual machine:

Command Option	Function
AGW QUERY GATEWAY	Displays the gateway names that are currently defined at this AVS virtual machine in the TSAF collection.
AGW QUERY CNOS	Displays the session limits, the contention winner, and the contention loser information for the gateways.
AGW QUERY CONV	Displays information about the current conversations.
AGW QUERY ETRACE	Displays the current setting of the external tracing.
AGW QUERY ITRACE	Displays the current setting of the internal tracing.
AGW QUERY USERID	Displays the current user ID mappings.

Command Option	Function
AGW QUERY ALL	Displays all of the above information.

Using AVS Error Messages

For certain types of APPC application errors, AVS will display an error message on the AVS console. Using this message it may be possible to quickly determine what is causing the problem. The meanings for RTNCD/FDB2 pairs, as well as RCPRI/RCSEC pairs, can be found in *VTAM Programming for LU 6.2*.

If AVS has difficulties opening its VTAM ACB, it will display the open error code. The meaning of this code can be found in *VTAM Programming Reference*.

Using VTAM Trace Facilities

If an APPC application is getting unexpected results, and there is no associated AVS error message, it may be necessary to obtain a VTAM LU buffer trace to examine the SNA data and control flows passing to and from the AVS gateway LU.

Consult the VTAM Diagnosis reference for detailed information on how to obtain buffer traces and how to format them using the trace analysis program (ACF/TAP) that is a part of the SNA system support services (SSP). With the formatted SNA detail (SD) trace you can see BINDs, UNBINDs, FMH-5s, FMH-7s, application data, and so on.

Pay special attention to the sense codes associated with FMH-7s, as it is the FMH-7 that is translated into an APPC return value. They often provide more detailed information than the APPC return value.

To help with interpreting the data and control information in the buffer trace, consult *Systems Network Architecture Network Product Formats*.

Part 6. Planning for ISFC

This part of the document describes how to plan for ISFC and includes the tasks that the system planner and network planner must do to set up the collection and program-to-program communications.

- Chapter 22, "Planning a CS Collection," on page 301 describes how to plan and design an CS collection, how to determine which types of links to use, how to define the links in the CS collection, and how security is supported in a CS collection.
- Chapter 23, "Operating the VM Domain Controller," on page 309 provides an overview of how to operate the VM domain controller.

Related Information: See the following sources for more information on the tasks in this section:

Task	Source
Defining VM domain controller characteristics	*z/VM: CP Planning and Administration*
Operating the VM domain controller	*z/VM: CP Commands and Utilities Reference*

Chapter 22. Planning a CS Collection

This section describes the following tasks to help plan your CS collection:

- Designing a CS collection
- Defining unique identifiers
- Defining links within the CS collection
- Reducing CTC links between multiple VM systems with broadcast routing
- Understanding how ISFC relates to TSAF and SNA network communications
- Migrating from TSAF collections to CS collections.

CS Collection Structure

The Inter-System Facility for Communications (ISFC) is a function of CP that provides communication services between transaction programs. Using the services ISFC provides, transaction programs running on a VM system can communicate with programs that run on other VM systems. The transaction programs can be written using the CPI Communication programming interface or APPC/VM routines.

Transaction programs can use ISFC to access, manage, and share resources defined in a CS collection. ISFC also enables programs in a CS collection to communicate with APPC programs on systems in the SNA network. A CS collection is formed when domains are interconnected. A domain consists of a domain controller, which manages transaction programs, and users.

The VM systems that run ISFC are called ***VM domain controllers***. CP acts as the domain controller for the server and requester virtual machines that have IUCV authorization to manage or access resources on that system. Transaction programs run in virtual machines on the z/VM systems running ISFC.

Systems running any current release of z/VM can be part of a CS collection.

Planning the CS Collection Configuration

CS collections form dynamically. When a domain controller becomes active, its domain (itself and the users signed on to it) is a CS collection of one domain. This single domain CS collection attempts to establish communications with the domain controllers at the end of each link defined to it. Domain controllers at each end of a link exchange information to determine if they can join and form a CS collection. After they verify that they are authorized to communicate, the domain controllers share the names of the global resources and the AVS or system gateways that each domain controller knows.

When a physical link between domain controllers becomes inactive, communications between programs stop. Each domain controller then deletes information about the global resources and the AVS or system gateways that reside in the other domain.

Defining Unique Identifiers

Within a CS collection, domain controller identifiers and global resource names must be unique. ISFC enforces unique domain controller and global resource names.

Within a CS collection, the VM domain controllers are identified by node names. These names must be unique because connection requests are routed using the name of the domain controller. If a domain controller has the same name as another domain controller that is already in the CS collection, the requesting domain controller cannot join the collection.

Defining the VM Domain Controller Name

The name of the VM domain controller is the same as the system gateway name defined for the system. The system gateway is created automatically when CP is initialized on the system. The system gateway name is identified on the SYSTEM_IDENTIFIER or SYSTEM_IDENTIFIER_DEFAULT statements in the system configuration file.

The default system gateway name is the system identifier. However, if you specify a unique system gateway name, the domain controller name of the VM system will also change. If you specify NOSYSGAT as the system gateway name on the SYSTEM_IDENTIFIER_DEFAULT or SYSTEM_IDENTIFIER statements, a system gateway will not be defined for the system and the system cannot join a CS collection.

For example, the following system configuration file statement defines SYSTEM1 as the name of the system identifier, system gateway, and the VM domain controller:

```
system_identifier 9221 051087 system1
```

In the next example, SYSTEM2 is specified as the system identifier. However, the system gateway name and domain controller name of this system are identified as SYSGATE2:

```
system_identifier 9221 071065 system2 gateway sysgate2
```

See *z/VM: CP Planning and Administration* for more information about system configuration file statements.

Defining Links in a CS Collection

VM domain controllers must use channel-to-channel (CTC) links to be attached to other VM domain controllers in the CS collection. The VM domain controllers may be running ISFC or VM PWSCF to participate in the CS collection.

The communication links that the VM domain controller will use must be defined to the VM system. To define a link, you may need to specify RDEVICE statements in the system configuration file. These statements must accurately represent the real hardware configuration of the system's I/O devices. See *z/VM: CP Planning and Administration* for more information about how to define real I/O devices, including a description of the types of links that must be defined by RDEVICE statements.

After the communication links are defined to the system, you must enter the ACTIVATE ISLINK command to enable the VM domain controller to use the link. See *z/VM: CP Commands and Utilities Reference* for information about the ACTIVATE ISLINK command.

Reducing CTC Links with Broadcast Routing

Broadcast Routing removes the requirement that all z/VM systems be directly connected in a CS collection. APPC and IUCV traffic between a source and target system is now forwarded through the intermediate systems. Global *IDENT Resource and Gateway identifies and revokes are also broadcast throughout the CS collection. This reduces the number of CTC links in collections of three or more z/VM systems. Note that while fully connected collections will still work, it is recommended that z/VM systems be connected in a circle configuration. This will provide optimal reliability and will result in better performance characteristics.

ISFC will always choose the shortest path when routing communications. Existing conversations are not re-routed when a shorter path becomes active between a source and target system. However, any new conversations will use the shorter path.

When an ISFC link goes down, existing conversations over the affected path are terminated. ISFC does not attempt to re-route these conversations over an alternate path. If the conversation is re-started and an alternate path exists, the conversation will succeed.

Your CS collection can contain pre-z/VM 7.3 systems in addition to z/VM 7.3 systems and subsequent releases. However, care must be taken to ensure that full connectivity is maintained because the pre-z/VM 7.3 systems will not forward communications.

Note: Access to private resources using an LU_name_qualifier of *USERID and originating on a pre-z/VM 7.3 system requires a direct link to the target system. This is because the pre-z/VM 7.3 system does not provide enough information for the z/VM 7.3 systems and subsequent releases to properly route private server connections. A direct link would not be required if the pre-z/VM 7.3 system uses the System Gateway of the target system as the LU_name_qualifier instead of *USERID.

z/VM systems will allow their Global *IDENT resources and gateways to be forwarded by a z/VM 7.3 system and subsequent releases. VM systems running the PWSCF PRPQ connected to a z/VM 7.3 system will forward their Global *IDENT resources and gateways to the z/VM 7.3 system. However, the z/VM 7.3 system and subsequent releases will *not* forward these resources and gateways to other VM systems in the CS collection.

Operating with TSAF and SNA Network Communications

A VM system that is running ISFC can also run a TSAF virtual machine; in this case, the system becomes part of a CS collection and a TSAF collection. ISFC communications services can coexist with the communications services provided by TSAF, AVS, and VTAM.

The following sections contain resource, security, and system definition considerations for a system that is part of a CS collection and a TSAF collection.

Managing Resources

ISFC and TSAF enforce unique resource identifiers. ISFC knows about the global resources, AVS gateways, and system gateways defined in the CS collection. TSAF knows about the global resources, AVS gateways, and system gateways defined in the TSAF collection. If a VM system is in both a CS collection and a TSAF collection, ISFC does not know about the global resources and gateways defined at other systems in the TSAF collection. TSAF does not know about the global resources and gateways defined at other systems in the CS collection.

If a user program requests a connection to a resource, the request is first sent to ISFC. If ISFC does not know about the resource, the request is sent to TSAF. For example, there are three systems, SYSTEMA, SYSTEMB, and SYSTEMC. SYSTEMA and SYSTEMB are in a CS collection, and SYSTEMB and SYSTEMC are in a TSAF collection. A global resource PRINTER is defined on SYSTEMA and SYSTEMC. If a program on SYSTEMB requests a connection to PRINTER, the request is sent to PRINTER on SYSTEMA in the CS collection.

Accessing Resources

To access a global resource that resides in the same CS collection, a transaction program can specify *IDENT or a system gateway name as the LU name qualifier; the target LU name is omitted. The resource name is specified as the TPN name.

To access a private resource within the CS collection, a transaction program can specify *USERID or a system gateway name as the LU name qualifier. The target LU is the user ID that manages the private resource.

However, if a resource resides in an adjacent TSAF collection, transaction programs must specify the system gateway name of the system that is part of each collection. To access a global resource, the system gateway name is specified as the LU name qualifier and the target LU name is omitted. To access a private resource, the system gateway name is specified as the LU name qualifier. The target LU is the user ID that manages the private resource.

For more information about specifying information to access resources in a CS collection, see Table 8 on page 170 and global resource.

Security Considerations

In general, when a transaction program specifies SECURITY(PGM), the user ID and password it supplies on the request must be valid in the TSAF or CS collection in which the target resource resides.

When a VM domain controller that is in both a TSAF and CS collection receives a connection request with SECURITY(PGM), it checks if the specified user ID and password are valid in the CS collection. If the user ID and password are valid, ISFC sends the request to the appropriate resource manager in the CS collection.

If the user ID and password are not defined in the CS collection, CP sends the request to TSAF and asks TSAF to determine if the user ID and password are valid in the TSAF collection. If the user ID and password are valid, ISFC sends the request to the appropriate resource manager. If TSAF indicates that the user ID and password are not valid, ISFC rejects the connection request.

For example, there are three systems, SYSTEMA, SYSTEMB, and SYSTEMC. SYSTEMA and SYSTEMB are in a CS collection, and SYSTEMB and SYSTEMC are in a TSAF collection. A user on SYSTEMB can specify a user ID and password defined on SYSTEMC for SECURITY(PGM) to connect to a resource on SYSTEMA.

System Definitions

If programs in the CS collection communicate with programs in the TSAF collection, you should determine if it is necessary to:

- Enable the server virtual machines to accept connections for any requester (IUCV ALLOW)
- Increase the number of connections that can be established with the virtual machine (OPTION MAXCONN)
- Add statements to the CMS communications directory to define resources that are located in the CS collection
- Add information to the $SERVER$ NAMES file on the private server virtual machines to define the CS collection users that can access private resources.

If AVS gateways are defined in the VM system, you should determine if it is necessary to:

- Enable the AVS virtual machine to accept connections for any requester (IUCV ALLOW)
- Increase the number of connections that can be established with the AVS virtual machine (OPTION MAXCONN)
- Add additional gateways to support the increased number of connections.

See Part 5, “AVS Virtual Machine,” on page 243 for information about AVS support.

Migrating from a TSAF Collection to a CS Collection

If all systems in a TSAF collection are able to run ISFC (or VM PWSCF), the systems may form a CS collection. Unlike TSAF, which runs in a separate virtual machine, ISFC support is provided on the system by CP. This may improve the performance of communications between programs within your collection. A CS collection does not enforce a maximum of 8 systems, as was true in a TSAF collection.

Migration Considerations

Before migrating your systems from a TSAF collection to a CS collection, you should review the following considerations:

- Systems running VM/SP 5 cannot join a CS collection.
- ISFC requires a CTC-type link to communicate with ISFC running on another VM host system. This is not required in a TSAF collection.
- If a link between two systems in the CS collection goes down, all conversations on that link also end. When the link becomes available again, any required conversations must be restarted. TSAF attempts to reroute conversations over an alternate link when a link fails.

Migrating from TSAF to ISFC

The following sections describe how you can use ISFC to migrate systems from a TSAF collection to form a CS collection. You can migrate one system at a time and, by using the system gateway, transaction programs in the TSAF collection can continue to access resources defined on other nodes.

To migrate, you must add IS links to systems; you may also have to delete and detach TSAF links between two systems. (When deleting TSAF links, be careful to maintain one TSAF collection; see "Avoiding Partitioned TSAF Collections" on page 307 for more information.)

Migrating the First System

In Figure 139 on page 305, a TSAF collection is made up of eight systems running z/VM. VMSYS1 has defined a CTC link, with address 404, to VMSYS2 and a CTC link, with address 4A0, to VMSYS8.

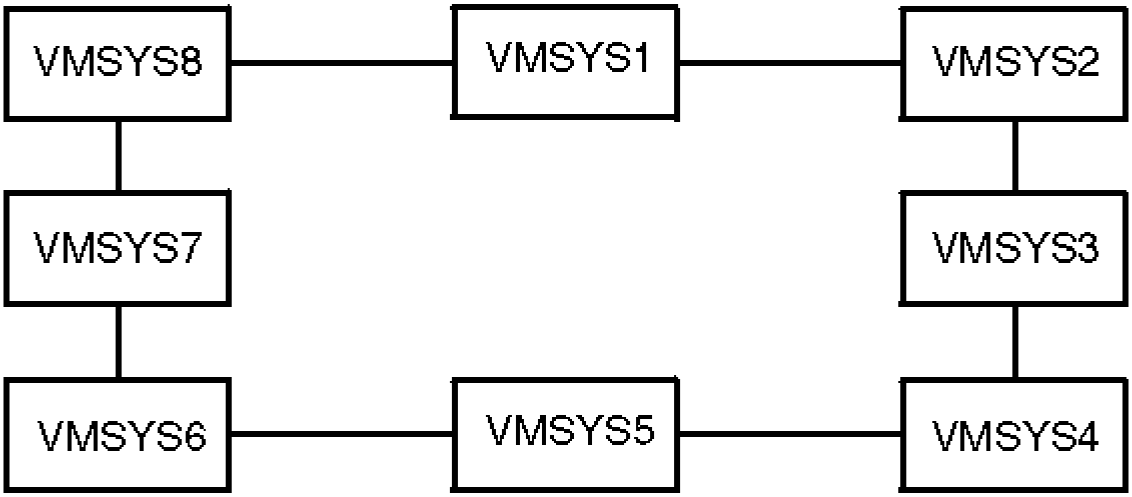


Figure 139. TSAF Collection with CTC Links

The first step in migrating from a TSAF collection to a CS collection is to delete the TSAF links for the first system. As Figure 139 on page 305 shows, a CTC link is defined between VMSYS1 and VMSYS2. To delete and detach this TSAF link, the TSAF operator on each system enters the following commands from the TSAF console:

```
delete link 404
detach 404
```

On VMSYS1 and VMSYS8, the TSAF operators enter the following commands from the TSAF console to delete the TSAF link:

```
delete link 4a0
detach 4a0
```

To establish an ISFC link between VMSYS1 and VMSYS8, the following command is entered from a class B user ID on each system:

```
activate islink 4a0
```

To establish an ISFC link between VMSYS1 and VMSYS2, the following command is entered from a class B user ID on each system:

```
activate islink 404
```

When the TSAF links are successfully deleted and the ISFC links are established, a CS collection and an adjacent TSAF collection are formed. As Figure 140 on page 306 shows, the CS collection contains three systems (VMSYS1, VMSYS2, and VMSYS8) and the TSAF collection contains seven systems (VMSYS2 through VMSYS8).

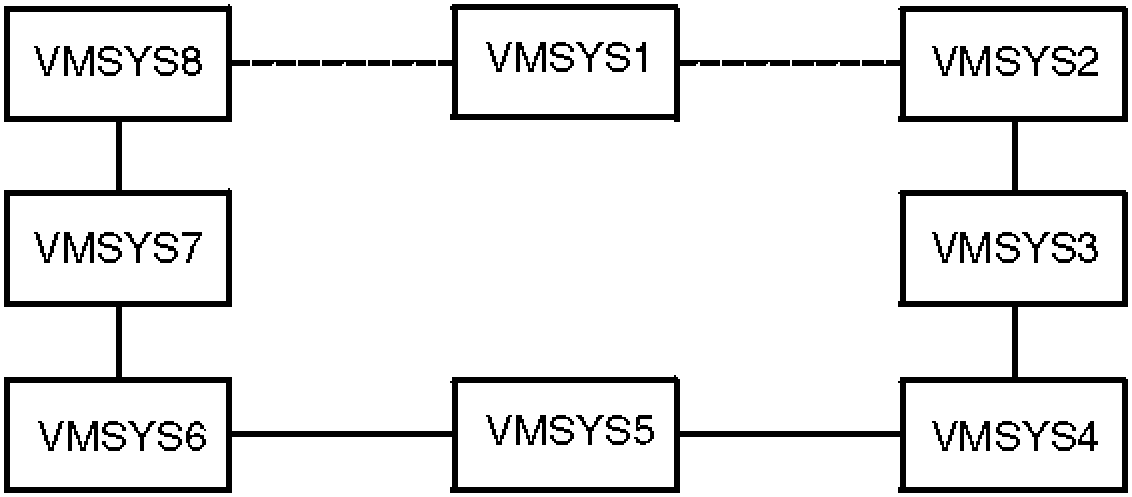


Figure 140. Forming a CS Collection and a TSAF Collection

Even though VMSYS2 and VMSYS8 are in both the TSAF and CS collections, communications between these two systems will always go through the ISFC links. Communications between VMSYS2 and VMSYS3 through VMSYS7 will continue to go through the TSAF links. The same is true for communications between VMSYS8 and VMSYS3 through VMSYS7.

Communications between VMSYS1 and VMSYS3 through VMSYS7 is still possible through the use of the system gateway of either VMSYS8 or VMSYS2. Choose the system gateway which results in the shortest path to the target system. For example, Communications between VMSYS1 and VMSYS3 would use the system gateway of VMSYS2. If symbolic destination names are defined for the resources, the system gateway name can be updated in the communications directory entries.

For example, the global resource GLOBRES1 is defined on VMSYS1 and a private resource, PRIVRES6, is defined on VMSYS6. The system gateway name SYSGATE8 is defined on VMSYS8. To access GLOBRES1, a program on VMSYS6 would specify SYSGATE8 as the LU name qualifier and omit the target LU name. The transaction program name for the request is GLOBRES1.

Similarly, to access the private resource on VMSYS6, a program on VMSYS1 would specify SYSGATE8 as the LU name qualifier. The target LU name is the user ID on VMSYS6 that manages the private resource. The transaction program name for the request is PRIVRES6.

Migrating Other Systems

Other systems in the TSAF collection can now join the CS collection. In Figure 141 on page 307, VMSYS3 joins the CS collection by establishing an ISFC link to VMSYS2. VMSYS2 must also establish an ISFC link to VMSYS3.

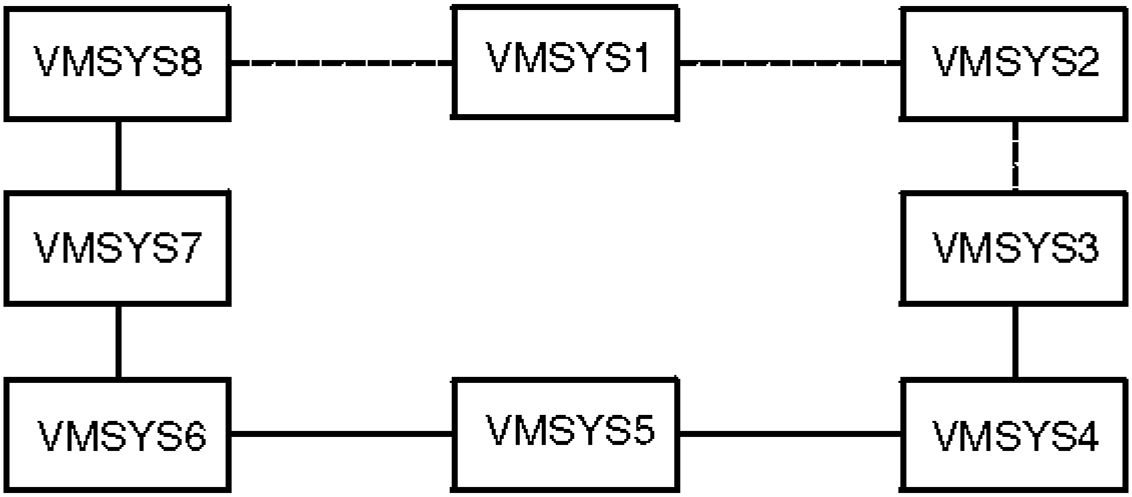


Figure 141. Adjacent TSAF and CS Collections

As Figure 141 on page 307 shows, VMSYS3 and VMSYS8 are now in the CS collection and the adjacent TSAF collection. Now you use the system gateway of VMSYS3 instead of VMSYS2 to communicate between the TSAF and CS collections. Continue using the VMSYS8 system gateway where appropriate.

Avoiding Partitioned TSAF Collections

When deleting TSAF links between two systems, be careful not to create two TSAF collections. If this occurs, transaction programs may not be able to access all resources that have been defined in the TSAF collection.

For example, Figure 142 on page 307 shows a CS collection and a partitioned TSAF collection. VMSYS4 has joined the CS collection with VMSYS1, VMSYS5, and VMSYS8. However, when the TSAF link between VMSYS4 and VMSYS5 was deleted, two TSAF collections were created. One consists of VMSYS5 through VMSYS8 and the other TSAF collection contains VMSYS2, VMSYS3, and VMSYS4. Because the TSAF collection has been partitioned, a transaction program on VMSYS2 can no longer access a resource on VMSYS7.

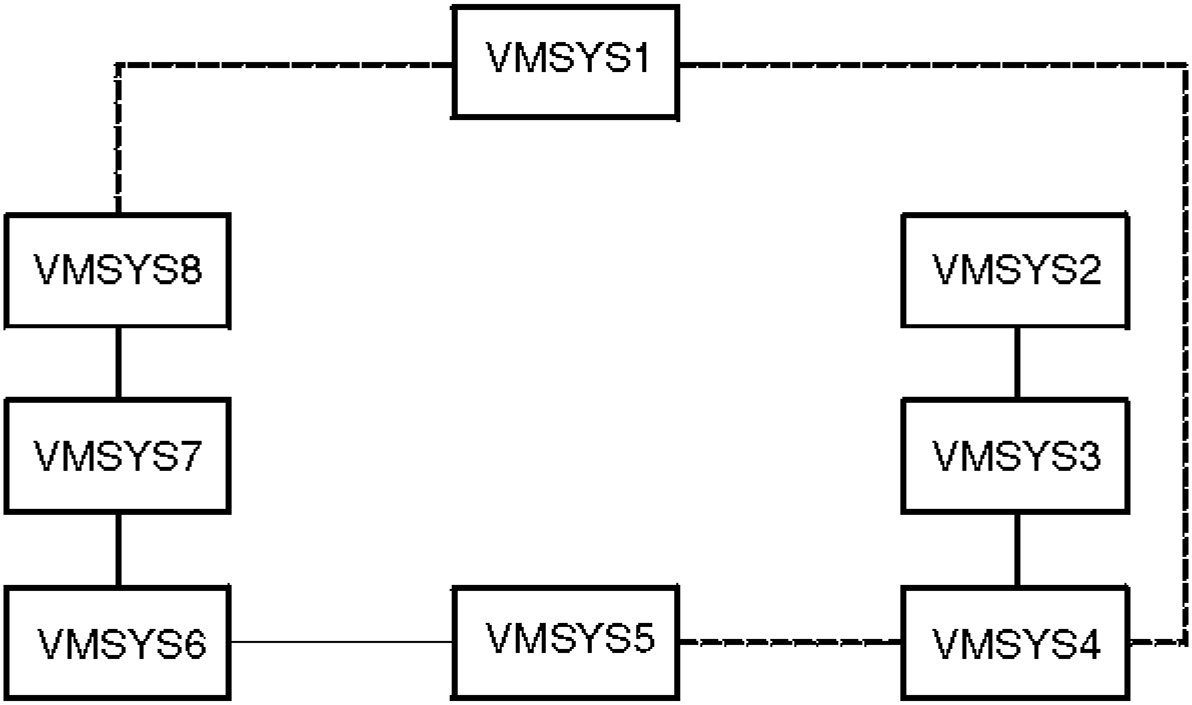


Figure 142. A CS Collection and Partitioned TSAF Collections

Chapter 23. Operating the VM Domain Controller

ISFC, which runs as part of CP, enables the VM domain controller to communicate with other domain controllers in the CS collection. The following sections provide an overview of the commands and facilities for operating the VM domain controller in a CS collection.

Overview of Commands

To operate the VM domain controller and display information about the CS collection, you can enter the following CP commands. For more information about these commands, see the *z/VM: CP Commands and Utilities Reference*.

Command	Function
ACTIVATE ISLINK	Activates an ISFC link between domain controllers.
DEACTIVE ISLINK	Deactivates an ISFC link between domain controllers.
DEACTIVE CONV	Deactivates a conversation between transaction programs communicating in the CS collection.
QUERY COLLECT	Displays information about the domain controllers and routes defined in the CS collection.
QUERY CONV	Displays information about APPC/VM conversations managed by the VM domain controller.
QUERY GATEWAY	Displays information about AVS gateways and system gateways defined in the CS collection.
QUERY ISLINK	Displays information about the IS links in the CS collection.
QUERY RESOURCE	Displays information about the local, global, and system resources defined on the system; it also displays information about the global resources defined in the CS collection.

Generating Accounting Information

CP generates accounting records for ISFC. These records (record type 09) are generated when you enter the CP command ACNT with the ALL operand:

Initialization
Contains information about ISFC initialization

Conversation
Contains information about active and ending conversations in the CS collection

Link statistics
Contains information about link activities within the CS collection

Termination
Contains information about ISFC termination.

For information about the format of the ISFC accounting records, see *z/VM: CP Planning and Administration*. See *z/VM: CP Commands and Utilities Reference* for more information about the ACNT command.

Forwarding ISFC Information to a Central Site

You can centralize system operations so that a central site oversees system operation for the VM domain controller and receives its messages that cannot be automatically handled at the distributed system. You can centralize operations by using the VM Programmable Operator facility.

The Programmable Operator Facility allows remote operation of systems in a distributed data processing environment. It does this by intercepting all messages and requests directed to its virtual machines and by handling them according to preprogrammed actions. These messages and requests are sent to a logical operator.

If operations are centralized at your installation using the IBM NetView family of products or RSCS, the NetView products or the Programmable Operator Facility can forward messages that require operator skills from the local site to a central site virtual machine where a real operator is located.

See *z/VM: CP Programming Services* for more information about the Programmable Operator Facility.

Appendix A. IUCV Directory Control Statement

This appendix describes the IUCV directory control statement. You use the IUCV directory control statement to authorize:

- Requester virtual machines to connect to resources (see "Authorizing Connections to Resources" on page 311)
- Server virtual machines to manage resources (see "Authorizing Resource Management" on page 312)
- AVS virtual machines to manage gateways (see "Authorizing the AVS Virtual Machine to Manage Gateways" on page 314).

The complete IUCV directory control statement is described in *z/VM: CP Planning and Administration*.

Specifying IUCV Statements

The IUCV directory control statement enables a virtual machine to use IUCV or APPC/VM communication paths. A virtual machine can use these communication paths to access or manage resources and gateways. To enable connectivity between virtual machines and systems, you must add IUCV directory control statements to the CP directory entries for the TSAF and AVS virtual machines. You must also add IUCV statements to the CP directory entries of the virtual machines that request or manage resources.

The IUCV directory control statement is specified in the CP directory entry of the virtual machine. You must specify this statement before any CONSOLE, SPOOL, LINK, or MDISK statements. For examples of IUCV statements, see Figure 90 on page 191 and Figure 119 on page 248.

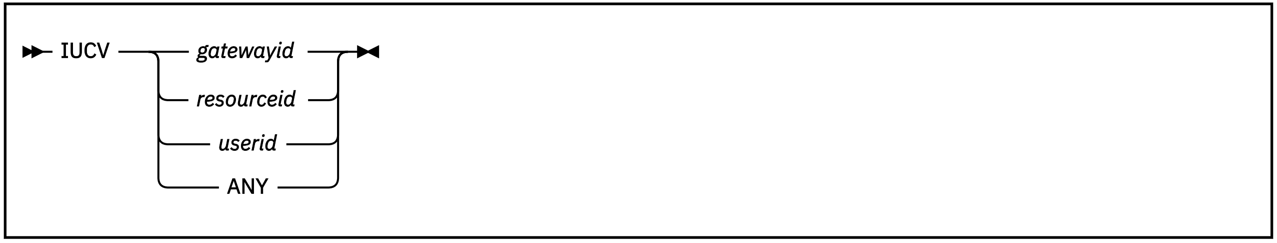
Authorizing Connections to Resources

Use the IUCV directory control statement to explicitly authorize a requester virtual machine to access specific resources, user IDs, or gateways. The resources may be in a TSAF or CS collection or in the SNA network.

You can specify the IUCV directory control statement in the CP directory entry for the requester virtual machine. If the requester virtual machine has an alternate user ID, you can also add this statement to the CP directory entry for the alternate user ID. Alternate user IDs are described in *z/VM: CPI Communications User's Guide*.

IUCV Statement Syntax (Connection Authority)

Use the following syntax of the IUCV statement to authorize a requester virtual machine to connect to a resource or gateway.



gatewayid

is a 1- to 8- character gateway name used to connect to the resources in the SNA network rather than to a specified virtual machine. The first byte of the gateway name must be alphanumeric. (IBM reserves the names beginning with nonalphanumeric characters for its own use.)

Be sure that the gateway name you specify is not the same as a user ID or resource name on the system, ALLOW, ANY, or SYSTEM.

resourceid

is a 1- to 8-character local or global resource name that connects to a resource manager rather than to a specified server virtual machine. The first byte of the resource name must be alphanumeric. (IBM reserves names beginning with nonalphanumeric characters for its own use.) If the resource name is less than 8 characters, you should left justify it and pad on the right with blanks.

Be sure that the resource name you specify is not the same as a user ID on the system, ALLOW, ANY, or SYSTEM.

userid

is the 1- to 8-character user ID of the private resource server virtual machine. The first byte of the user ID must be alphanumeric. (IBM reserves names beginning with nonalphanumeric characters for its own use.)

ANY

enables the requester virtual machine to establish a connection to any resource, user ID, or gateway.

Table 11 on page 312 summarizes the authority a virtual machine obtains when you specify this format of the IUCV statement.

Table 11. IUCV Statements to Authorize Connections to Resources

Authorization	IUCV Statement	Comments
Local or global resource	IUCV *resourceid*	Requester program is not authorized to specify the user ID of the server virtual machine that owns the resource.
Private resource	IUCV *userid*	Requester program is not authorized to specify a resource ID when connecting to a server virtual machine.
Specific gateway	IUCV *gatewayid*	Connection requests to resources in the SNA network are routed through the specified gateway.
Any resource or gateway	IUCV ANY	Requester virtual machine is authorized to connect to any resource, user ID, or gateway.

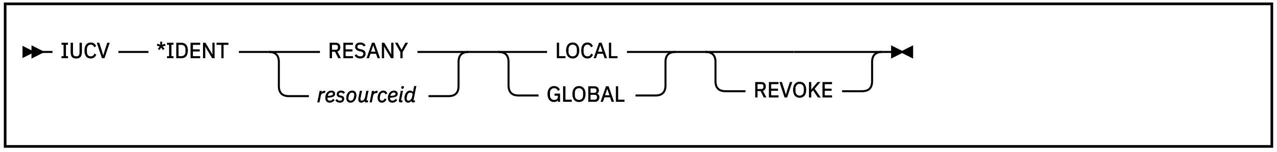
Note: When you authorize each requester virtual machine, you should also give explicit directory authorization to the TSAF virtual machine residing on the same system as the private or global resource or gateway (with IUCV *resourceid*, IUCV *userid*, IUCV *gatewayid*, or IUCV ANY). See "Examples of Authorizing Virtual Machines" on page 315 for examples of explicitly authorized TSAF collections.

Authorizing Resource Management

To authorize a server virtual machine to manage a resource, you must specify an IUCV *IDENT statement in the CP directory for the virtual machine.

IUCV Statement Syntax (Resource Management)

Use the following syntax to specify an IUCV *IDENT statement to authorize a server virtual machine to manage resources.



***IDENT**

lets the virtual machine connect to the Identify System Service.

RESANY

lets the virtual machine identify any resource name; the virtual machine can also identify any number of resources in the TSAF or CS collection.

Be careful when you give authorization for RESANY. A virtual machine that has authorization for RESANY can identify a resource using any resource name, including the name “resany”.

resourceid

is a 1- to 8- character resource name. APPC/VM programs can connect to the resource manager that manages the resource specified by *resourceid*. The first byte of the resource name should be alphanumeric. (IBM reserves names beginning with the nonalphanumeric characters for its own use.) If the resource name is less than 8 characters, you should left justify it and pad on the right with blanks.

The *resourceid* should not be the same as a user ID on the system, ALLOW, ANY, or SYSTEM.

LOCAL

authorizes the virtual machine to identify the resource as a local resource known only to the local system. If you specify LOCAL with RESANY, the virtual machine can identify any resource as a local resource.

GLOBAL

authorizes the virtual machine to identify the resource as a global resource, which will be known to all systems in the TSAF or CS collection. The virtual machine can also identify the resource as a local resource. If you specify GLOBAL with RESANY, the virtual machine can identify any number of local or global resources. This option also allows a virtual machine to identify system resources.

REVOKE

authorizes the virtual machine to revoke the specified resource name. A virtual machine that can revoke resources can also identify them.

If you specify REVOKE with:

- LOCAL, the virtual machine can revoke and identify the specified resource on the local system only.
- GLOBAL, the virtual machine can revoke and identify:
  - The specified global resource
  - A local or system resource on the local system that has the same name as the global resource.

A virtual machine cannot revoke a global and local resource at the same time. The virtual machine must specify which resource to revoke when the connection is made to *IDENT.

- RESANY and LOCAL, the virtual machine can revoke and identify one or more local resources.
- RESANY and GLOBAL, the virtual machine can revoke and identify one or more local, global, or system resource.

Because TSAF and ISFC do not keep track of the local or system resources, a virtual machine cannot revoke a local or system resource on another system.

Identifying Multiple Resources

If a server virtual machine manages more than one resource, its CP directory entry must contain an IUCV *IDENT statement for each resource. Alternatively, the server virtual machine could have RESANY authority. *IDENT checks if the virtual machine is authorized to identify or revoke the resource specified on the IUCV CONNECT function to *IDENT. *IDENT searches the server virtual machine's CP directory entry for IUCV directory control statements in this order:

1. The first *IDENT entry that has the same resource name as specified on the CONNECT request. If *IDENT does not find a match or finds a match but authorization for the LOCAL/GLOBAL and REVOKE parameters does not correspond to those specified in the user data field of the identify request, *IDENT searches for,

2. The first *IDENT entry that has the resource name RESANY. If *IDENT finds a match, it checks that the authorization for the LOCAL/GLOBAL and REVOKE parameters corresponds to those specified in the user data field of the identify request.
3. If it does not find a match, or if the other parameters do not correspond, *IDENT rejects the identify request.

Example 1: The CP directory entry for the server virtual machine RESMGR1, contains the following IUCV directory control statements:

```
IUCV *IDENT RESANY GLOBAL
IUCV *IDENT RESIDX LOCAL REVOKE
```

These statements identify RESMGR1 as a resource manager. RESMGR1 can identify any resource as a local or global resource, because of the first statement. This includes the local resource, RESIDX. However, RESMGR1 can only revoke the resource, RESIDX, when it is defined on the local system as a local resource.

Example 2: A resource manager, RESMGR2, has the following IUCV directory control statements:

```
IUCV *IDENT RESIDX GLOBAL
IUCV *IDENT RESIDY GLOBAL
IUCV *IDENT RESANY LOCAL REVOKE
```

RESMGR2 can identify the resources, RESIDX and RESIDY, as local or global resources. RESMGR2 is not authorized to revoke any global resources. The last directory control statement enables RESMGR2 to identify any resource as a local resource. RESMGR2 can also revoke any resource known on the local system as a local resource.

Example 3: A resource manager, RESMGR3, has the following IUCV directory control statements:

```
IUCV *IDENT RESIDX LOCAL
IUCV *IDENT RESIDY GLOBAL
IUCV *IDENT RESIDX GLOBAL REVOKE
```

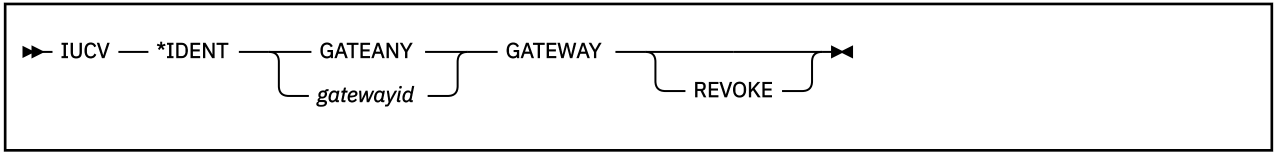
RESMGR3 can identify the resource, RESIDX, as a local resource and the resource, RESIDY, as a local or a global resource. RESMGR3 cannot revoke any resources. The reason for this is that *IDENT searches for the first entry that matches the resource name specified on the connection request. If RESMGR3 tries to connect to *IDENT to identify or revoke the GLOBAL resource, RESIDX, *IDENT severs the connection. In this case, if you want RESMGR3 to identify and revoke the global resource RESIDX, you would delete the first directory control statement.

Authorizing the AVS Virtual Machine to Manage Gateways

The IUCV directory control statement also authorizes the AVS virtual machine to manage a gateway. You must specify the gateway name on an IUCV *IDENT statement in the AVS virtual machine's CP directory entry. (To define the gateway, you specify the gateway name on the "AGW ACTIVATE GATEWAY" on page 271 command.)

IUCV Statement Syntax (Gateway Management)

Use the following syntax to specify an IUCV *IDENT statement to enable the AVS virtual machine to manage a gateway.



***IDENT**
lets the AVS virtual machine connect to the Identify System Service (*IDENT).

GATEANY
lets the virtual machine identify any gateway name. Be careful when you give authorization for GATEANY. A virtual machine that has authorization for GATEANY can identify any gateway name, including the name “gateany”.

gatewayid
is the 1- to 8-character gateway name. The first character of the gateway name should be alphabetic, @, #, or $. (IBM reserves names beginning with the remaining characters for its own use.) Be sure that the gateway name you specify is not ALLOW, ANY, or SYSTEM.

GATEWAY
indicates that this is an authorization for a gateway.

REVOKE
authorizes this virtual machine to revoke and identify a gateway anywhere in the TSAF or CS collection.

Identifying Multiple Gateways

The AVS virtual machine can manage more than one gateway. In this case, the AVS virtual machine's CP directory entry must contain an IUCV *IDENT directory control statement for each gateway. You can also give the AVS virtual machine GATEANY authority.

*IDENT checks if the AVS virtual machine is authorized to identify the gateway specified on the IUCV CONNECT to *IDENT. *IDENT searches the AVS virtual machine's CP directory entry for IUCV directory control statements in this order:

1. The first *IDENT entry that has the same gateway name as specified on the connection request. If *IDENT does not find a match or finds a match whose GATEWAY authorization parameter is not the same as those specified in the CONNECT parameter list, *IDENT searches for,
2. The first *IDENT entry that has the gateway name GATEANY. If *IDENT finds a match, it checks that the authorization for the GATEWAY parameter corresponds to that specified in the CONNECT parameter list. If the authorization is the same, it accepts the connection. If it does not find a match, then it severs the requested connection.

Example: A gateway manager AVS1, has the following IUCV directory control statements:

```
IUCV *IDENT GATEX GATEWAY
IUCV *IDENT GATED GATEWAY
```

AVS1 identifies the gateway GATEX and GATED on an AGW ACTIVATE GATEWAY command.

Examples of Authorizing Virtual Machines

The following examples show how to explicitly authorize server virtual machines, the AVS virtual machine, and requester virtual machines.

Example 1: Figure 143 on page 316 is an example of an explicitly authorized TSAF collection involving two z/VM systems sharing global resources. The entries within each box represent the CP directory entries for each CMS virtual machine.

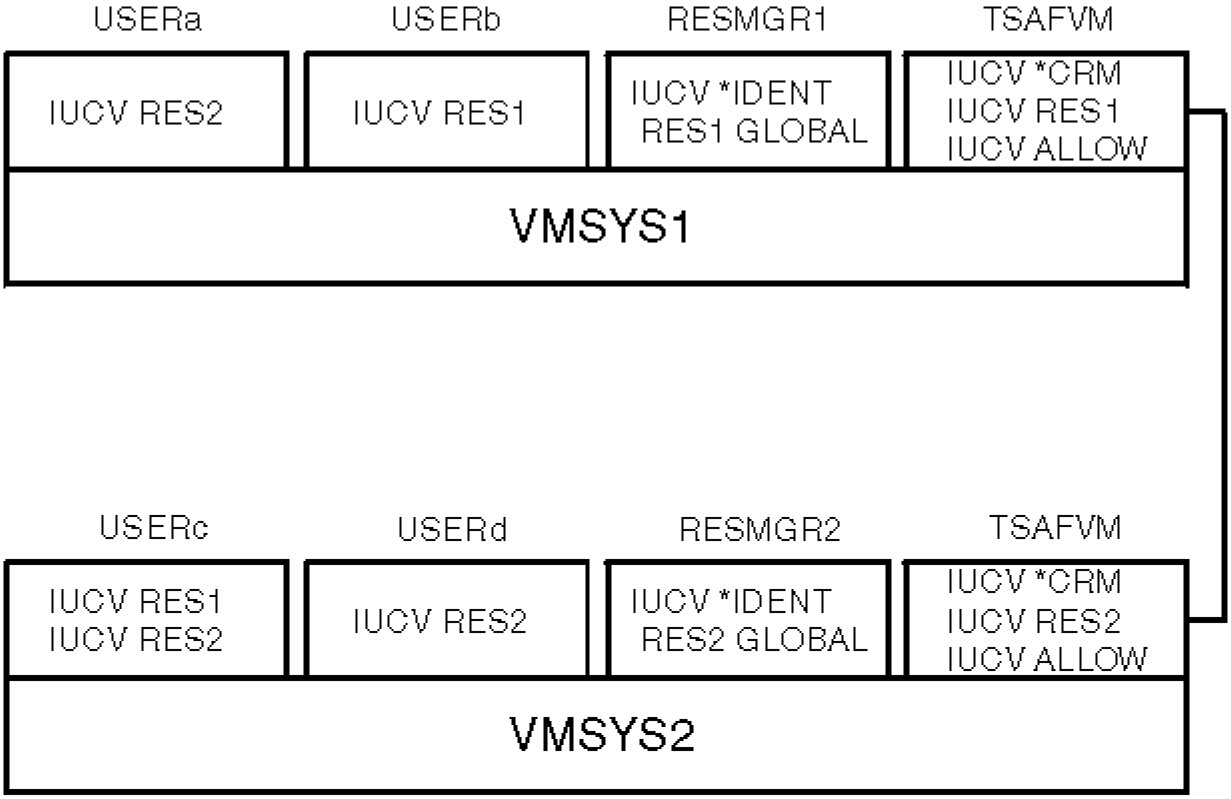


Figure 143. TSAF Collection with Authorized Global Resource Managers and User Programs

In Figure 143 on page 316, users have the following authorization:

- USERa on VMSYS1 can connect only to RES2 on VMSYS2.
- USERb on VMSYS1 can connect only to RES1 on VMSYS1.
- USERc on VMSYS2 can connect to RES1 on VMSYS1 and to RES2 on VMSYS2.
- USERd on VMSYS2 can connect only to RES2 on VMSYS2.

Example 2: Figure 144 on page 317 shows a TSAF collection in which the server and requester virtual machines are explicitly authorized to share local and private resources. The entries within each box represent the CP directory entries of each CMS virtual machine.

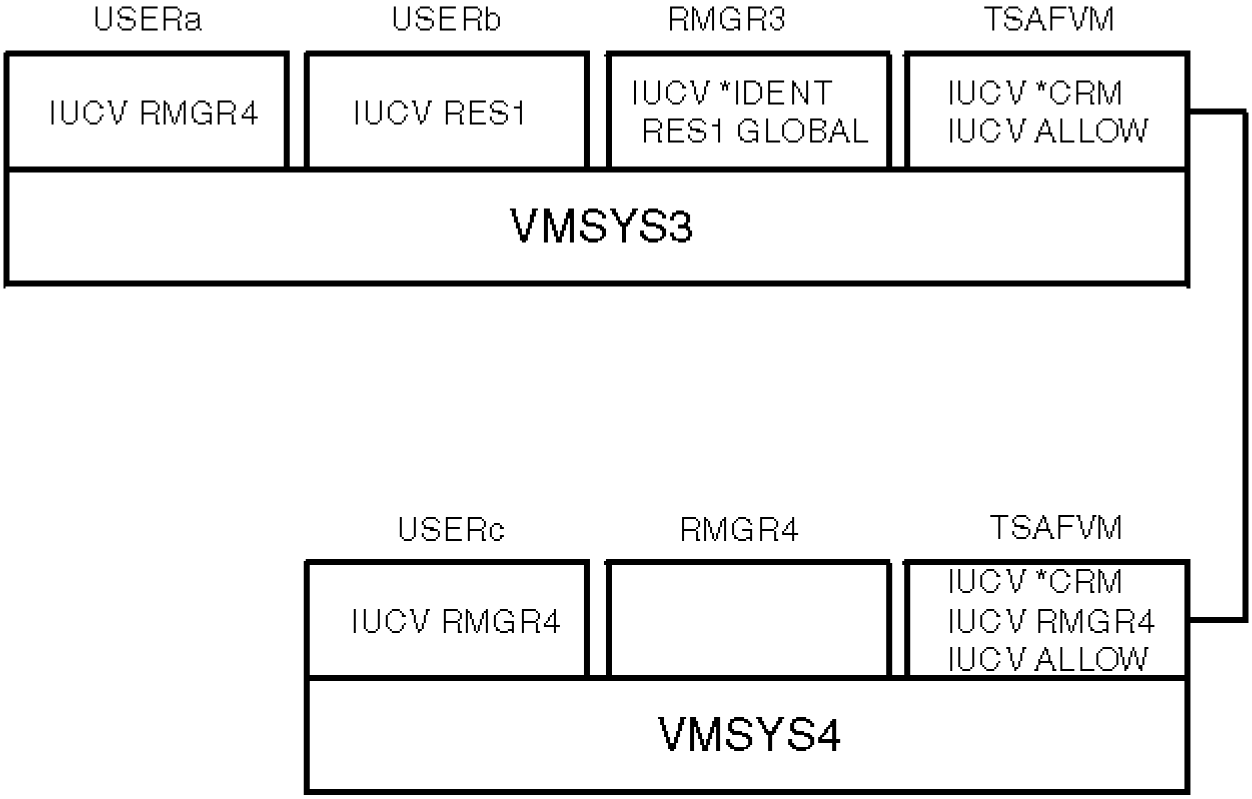


Figure 144. TSAF Collection with Authorized Local and Private Resource Managers and User Programs

In this figure, users have the following authorization:

- USERa on VMSYS3 can connect only to RMGR4 on VMSYS4 to access a private resource managed by RMGR4.
- USERb on VMSYS3 can connect only to RES1 on VMSYS3.
- USERc on VMSYS4 can connect only to RMGR4 on VMSYS4 to access a private resource managed by RMGR4.

Example 3: Figure 145 on page 318 shows an explicitly authorized TSAF collection involving two z/VM systems and one AVS virtual machine. The entries within each box represent the CP directory entries for each CMS virtual machine and the AVS virtual machine.

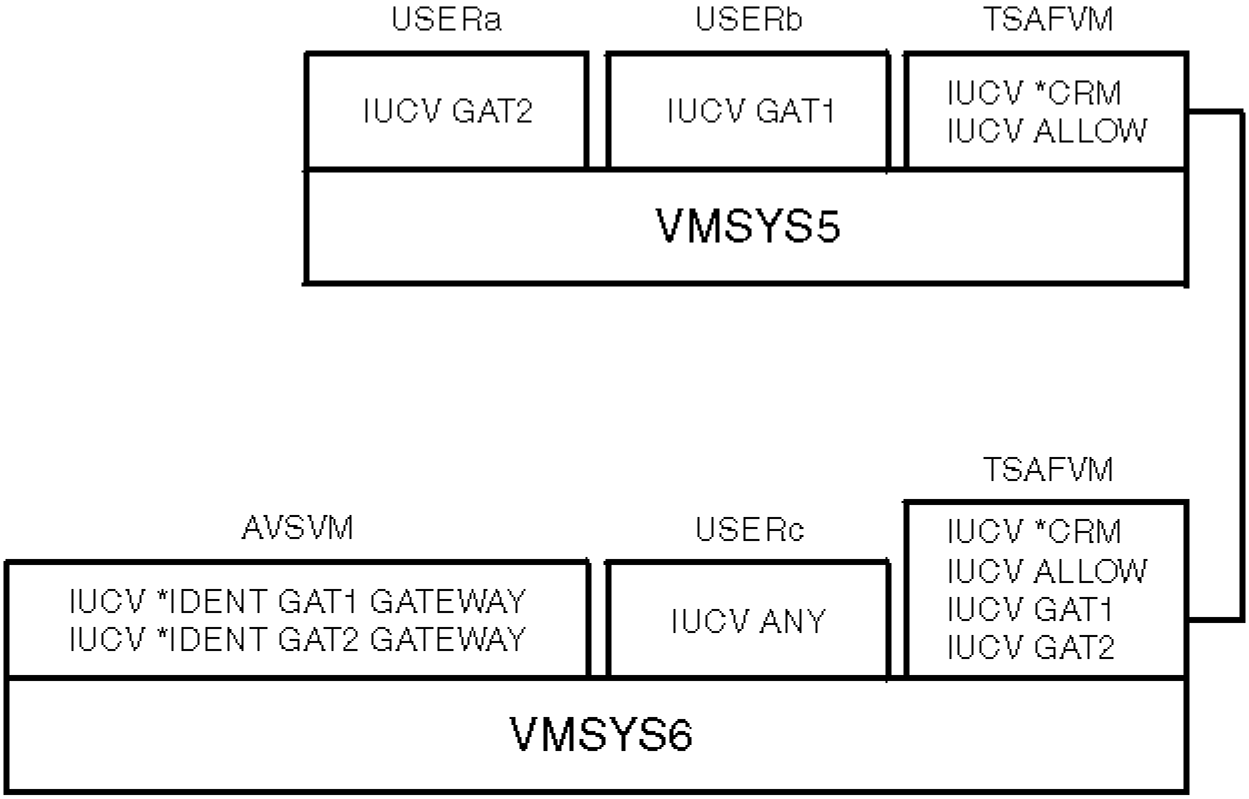


Figure 145. TSAF Collection with an AVS Virtual Machine

In this figure, users have the following authorization:

- USERa on VMSYS5 can only connect out to the SNA network through GAT2 on VMSYS6.
- USERb on VMSYS5 can only connect out to the SNA network through GAT1 on VMSYS6.
- USERc on VMSYS6 can connect out to the SNA network through any gateway defined on VMSYS6 because it is authorized to connect to any virtual machine, resource, or gateway on the local system.

Appendix B. Connectivity Summary

This section summarizes the information a user program must specify when requesting a connection to various types of resources.

- Table 12 on page 320 summarizes connections to resources within the TSAF or CS collection.
- Table 13 on page 322 summarizes outbound connections to resources in the SNA network.
- Table 14 on page 322 summarizes inbound connections to global and private resources in a TSAF or CS collection.

Connections within a TSAF or CS Collection

Table 12 on page 320 describes the information a user program specifies to connect to a resource within a TSAF or CS collection. If a communications directory entry is defined for the resource, the user program can specify a symbolic destination name on the connection request. If a symbolic destination name is not defined, the user program must explicitly specify the location of the resource and security information on the allocation request.

For local, global, and system resources, the connection request is routed by the name of the target resource (transaction program name). The server virtual machine for the resource manager program must be running when the connection request is made.

Connection requests to private resources are routed by the user ID of the server virtual machine. When the request is made, the private server virtual machine does not need to be logged on; CP will autolog the virtual machine.

Table 12. Connection to Resources within the TSAF or CS Collection

Information Specified By	Target Identified By	Access Security Information Specified By
Communications directory	The resource is identified by a symbolic destination name, which is specified on the :nick. tag. The user program must specify the symbolic destination on the allocation request.	The security level is identified on the :security. tag in the communications directory entry. The following security levels determine the information the target LU receives: **NONE**: Target LU does not receive user ID or password information about the requester virtual machine. **SAME**: Target LU receives the user ID of the requester virtual machine, but does not receive the password. **PGM**: Target LU receives user ID and password information about the requester virtual machine; this may be specified in: • The :userid. and :password. tags in the communications directory entry for the specified symbolic destination name • APPCPASS directory statement • The :userid. tag in the communications directory entry and the password specified on the APPCPASS statement
User Program (communications directory not used)	The user program must provide the locally-known LU name, transaction program name, and mode name on the allocation request.	The user program must specify one of the following values on the allocation request: **SECURITY(NONE)**: Target LU does not receive user ID or password information. **SECURITY(SAME)**: Target LU receives the user ID of the requester virtual machine, but does not receive the password. **SECURITY(PGM)**: Target LU receives the user ID and password of the user program. The user program must provide the user ID and password or the information may be obtained from an APPCPASS directory statement.

Access Security Verification

Verification of access security information varies, depending on the security level specified on the allocation request. The user ID is sent to the resource manager, which determines if the user program is authorized to access the resource.

SECURITY(NONE)

When a user program specifies SECURITY(NONE), z/VM does not verify that the program is authorized to connect to the resource. The resource manager program must allow any user to connect to the resource. For a private resource, the :list. field in the $SERVER$ NAMES file entry contains *.

SECURITY(SAME)

When a user program specifies SECURITY(SAME), z/VM does not check if the user program is authorized to connect to the resource.

For private resources, z/VM checks if the specified user ID is the same as the user ID of the server virtual machine. If they are not the same, z/VM checks the :list. field of the $SERVER$ NAMES file entry for the resource to determine if the user ID is authorized to access the resource. If the user ID is the same as the user ID of the server virtual machine, the $SERVER$ NAMES file is not checked.

SECURITY(PGM)

When a user program specifies SECURITY(PGM), z/VM verifies if the specified user ID and password are valid in the TSAF or CS collection. To do so, z/VM requests that each system in the TSAF or CS collection validate the user ID and password against the CP directory entry defined on that system. If a system validates the user ID and password, the user ID is then sent to the resource manager, which determines if the requester is authorized to access the resource.

For private resources, z/VM checks if the specified user ID is the same as the user ID of the server virtual machine. If they are not the same, z/VM checks the :list. field of the $SERVER$ NAMES file entry for the resource to determine if the user ID is authorized to access the resource. If the user ID is the same as the user ID of the server virtual machine, the $SERVER$ NAMES file is not checked.

Outbound Connections to the SNA Network

When a user program in a TSAF or CS collection requests access to a resource in the SNA network, the request is routed through an AVS gateway. The user program specifies the name of this gateway as the locally-known LU name. AVS then passes this request to VTAM, which builds the fully-qualified name that routes the connection to the target LU in the SNA network.

Table 13. Outbound Connection to Resources in the SNA Network

Information Specified By	Target Identified By	Access Security Information Specified By
Communications directory	Resource is identified by a symbolic destination name, which is specified on the :nick. tag. The user program must specify the symbolic destination on the allocation request.	The security level is identified on the :security. tag in the communications directory entry. The following security levels determine the information that is passed to the target LU: **NONE**: Target LU does not receive user ID or password information about the requester virtual machine. **SAME**: Target LU receives the user ID of the requester virtual machine, but does not receive the password. **PGM**: Target LU receives user ID and password information about the requester virtual machine; this may be specified in: • The :userid. and :password. tags in the communications directory entry for the specified symbolic destination name • APPCPASS directory statement • The :userid. tag in the communications directory entry and the password specified on the APPCPASS statement
User Program (communications directory not used)	The user program must provide the locally-known LU name, transaction program name, and mode name on the allocation request.	The user program must specify one of the following values on the allocation request: **SECURITY(NONE)**: Target LU does not receive user ID or password information. **SECURITY(SAME)**: Target LU receives the user ID of the requester virtual machine, but does not receive the password. **SECURITY(PGM)**: Target LU receives the user ID and password of the user program. The user program must provide the user ID and password or the information may be obtained from an APPCPASS directory statement.

Inbound Connections from the SNA Network

User programs within the SNA network can access resources within a TSAF or CS collection. Table 14 on page 322 summarizes the items a user program must specify for an inbound connection request.

To access a global resource, the user program specifies the name of a global gateway. The virtual machine in which the global resource manager resides must be logged on when the allocation request is made.

To access private resources, the user program can specify a dedicated or a nondedicated private gateway. When the allocation request is made, the private server virtual machine does not need to be logged on; CP will autolog the virtual machine.

Table 14. Inbound Connection to Resources in a TSAF or CS Collection

Item	Global Gateway	Dedicated Private Gateway	Nondedicated Private Gateway
Target LU Name	Gateway name	Gateway name	Gateway name

Table 14. Inbound Connection to Resources in a TSAF or CS Collection (continued)

Item	**Global Gateway**	**Dedicated Private Gateway**	**Nondedicated Private Gateway**
Transaction Program Name	Global resource name	Private resource name	Private resource name
Access Security Type	**SECURITY(NONE)**: User ID and password are not sent. **SECURITY(SAME)**: User ID (valid in the collection); no password is sent. **SECURITY(PGM)**: User ID and password (valid in the collection).	**SECURITY(NONE)**: User ID and password are not sent. **SECURITY(SAME)**: User ID (valid in the collection); no password is sent. **SECURITY(PGM)**: User ID and password (valid in the collection).	**SECURITY(SAME)**: User ID (valid in the collection); no password is sent. **SECURITY(PGM)**: User ID and password (valid in the collection).
Connection Routed By	Global resource name (transaction program name)	User ID of the server virtual machine, which is specified on the AGW ACTIVATE GATEWAY command.	User ID of the server virtual machine, which is specified in the connection request. If SECURITY(SAME) is specified, the user ID may be obtained from the AVS user ID table.

Access Security Verification

The verification of the access security information varies, depending on the level and the type of gateway specified on the allocation request.

SECURITY(NONE)

The resource manager must allow any user to connect to the resource. For a connection through a dedicated private gateway, `:list.*` must be specified in the $SERVER$ NAMES file entry for the resource.

SECURITY(SAME)

z/VM verifies that the user ID specified on the request is valid in the TSAF or CS collection. The AVS user ID table maps the specified remote user ID to a local user ID that is used in this system. This mapping is created when you enter the AGW ADD USERID command.

For global resources, the local user ID is sent to the global resource manager, which determines if the requester is authorized to access the resource.

On requests through dedicated private gateways, the specified user ID might not be the same as the user ID of the private server virtual machine. If they are not the same, the private resource manager checks if the user ID is specified in the :list. field of the $SERVER$ NAMES file. If the user ID is the same as the user ID of the server virtual machine, the $SERVER$ NAMES file is not checked.

For requests through nondedicated private gateways, the $SERVER$ NAMES file is not checked because the specified user ID is the same as the user ID of the private server virtual machine.

SECURITY(PGM)

When a user program in the SNA network specifies SECURITY(PGM) on an allocation request, the target z/VM system verifies that the specified user ID and password are valid in the TSAF or CS collection. To do so, z/VM requests that each system in the TSAF or CS collection validate the user ID and password against the CP directory entry defined on that system. If a system validates the user ID and password, the user ID is then sent to the resource manager, which determines if the requester is authorized to access the resource.

For global resources, the user ID specified on the request is sent to the global resource manager. The resource manager must determine if the requester is authorized to access the resource.

On requests through dedicated private gateways, the specified user ID may not be the same as the user ID of the private server virtual machine. If they are not the same, the private resource manager checks if the user ID is specified in the :list. field of the $SERVER$ NAMES file. If the user ID is the same as the user ID of the server virtual machine, the $SERVER$ NAMES file is not checked.

For requests through nondedicated private gateways, the $SERVER$ NAMES file is not checked because the specified user ID is the same as user ID of the private server virtual machine.

Appendix C. Introduction to Service Pool Support

Service Pool Support lets independent application host services run on z/VM; these services can also be accessed from a programmable workstation or another z/VM system in the SNA network. A workstation or another z/VM system connects to the z/VM host using a APPC session.

What Is a Service Pool?

A service pool consists of several identical virtual machines set up to handle transactions. Typically, these transactions would originate outside the z/VM host system, such as from a workstation or from another z/VM system. However, they may originate from within the host system. A service pool environment includes all of the service pool virtual machines and the supporting service programs located in a z/VM host. The configuration of a service pool environment is flexible and dependent only upon the application or product utilizing the service pool. Figure 146 on page 325 shows a simple service pool environment (with resulting path from an APPC connection).

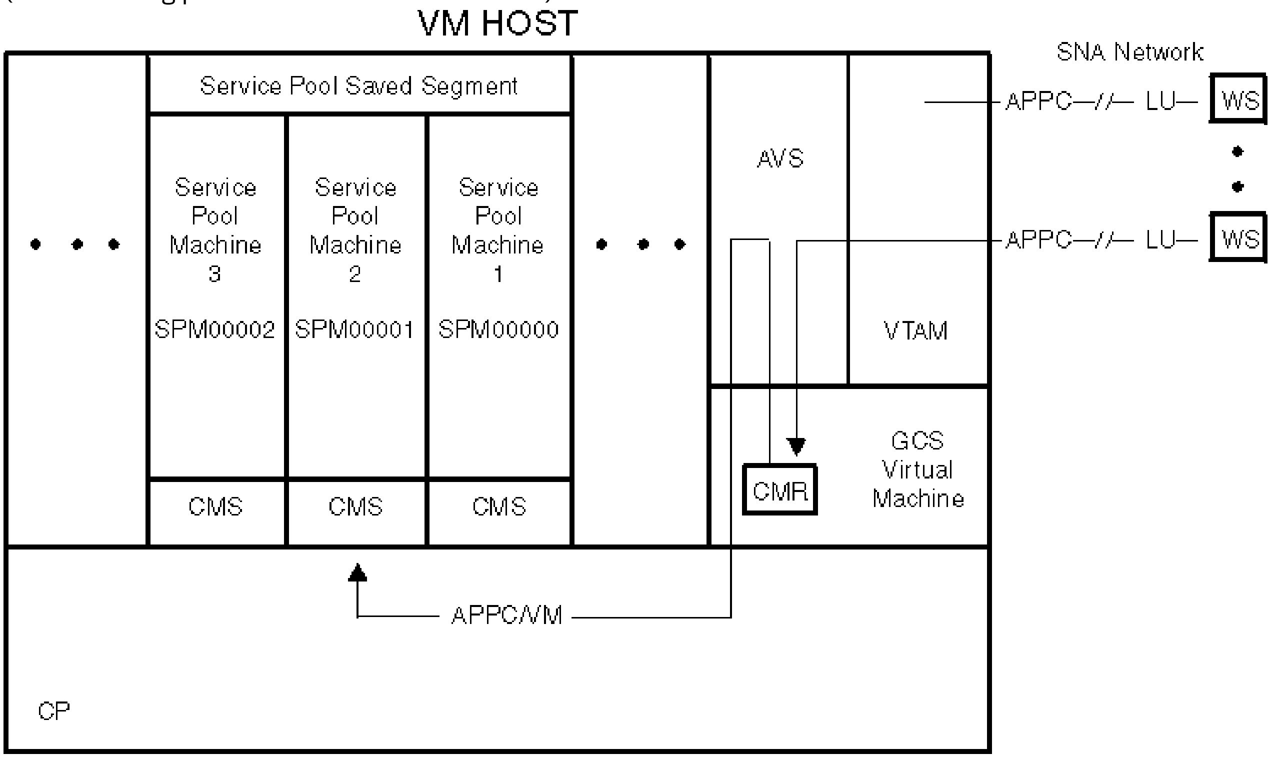


Figure 146. Service Pool Environment

Theoretically, a service pool can be comprised of up to 100,000 service pool virtual machines. In the service pool environment shown in Figure 146 on page 325, the service pool consists of three virtual machines (identified as SPM00000 through SPM00002) running in a z/VM host. In this environment, each service pool virtual machine shares the same service pool saved segment area.

The group of identical service pool virtual machines handles transactions originating from a workstation, another z/VM system, or any system communicating with APPC in the SNA network. Transactions come in through VTAM and are processed by the z/VM host. These transactions correspond to an invocation of a transaction program using APPC/VM private server support. APPC conversation allocations, generated by a workstation request for host services, are routed to an available virtual machine in this service pool. If all of the service pool virtual machines are busy, the allocation request is temporarily queued until a service pool virtual machine becomes available. When a transaction completes, the workstation or service machine deallocates the conversation. The individual service pool virtual machine is then identified as available for the next transaction, which could come from any active workstation.

AVS facilitates the service pool management. A private gateway LU can have a Conversation Management Routine (CMR) associated with it whenever Service Pool Support is required. The CMR is commonly known as the service pool manager and is product dependant. Multiple CMRs and service pool environments may be needed if an installation supports several products that use Service Pool Support. Because only one CMR is allowed per gateway, multiple gateways would be required in this environment.

When the gateway LU is activated, the CMR program is loaded into GCS private storage. As Figure 147 on page 326 shows, certain events cause AVS to interact with the CMR; Table 15 on page 326 describes these events. When an event occurs, AVS links to the CMR with a parameter list appropriate to the event; the CMR then returns the modified parameter list to AVS. The modified parameters typically contain information that lets the CMR control the routing from a workstation to a service pool virtual machine through AVS. See "Interactions between AVS and a CMR" on page 327 for more information on the parameter lists passed between AVS and the CMR.

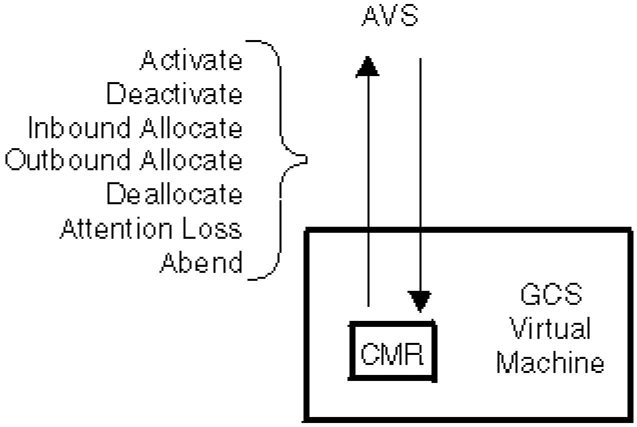


Figure 147. AVS Interface with CMR (Service Pool Manager)

Table 15. AVS Events that Cause Interactions between AVS and the CMR

Event	Event Description
Activate	Gateway LU is activated by the AGW ACTIVATE GATEWAY command.
Deactivate	Gateway LU is deactivated.
Inbound Allocate	An allocate request flows from VTAM to AVS.
Outbound Allocate	An allocate request flows from a virtual machine in the local collection to AVS.
Deallocate	The connection from AVS to a virtual machine in the local collection is severed.
Attention loss	VTAM reports an error on an attention loss.
Abend	AVS encounters an irrecoverable error condition.

An application program provides the CMR to autolog the service pool virtual machines, route conversations to a selected service pool virtual machine, and select the alternate user ID under which a service pool virtual machine runs.

The service pool environment performs a service for an application outside the z/VM host system. Communications are made possible by the assignment of an alternate user ID. The individual service pool virtual machine that is selected to process a transaction uses the user ID specified by the CMR as its alternate user ID.

This alternate user ID assignment is made when a service pool virtual machine accepts a connection and is cleared when the connection is severed. Connection pending interrupts that contain an alternate user ID are not presented to a service pool virtual machine that already has an alternate user ID assigned; such interrupts are deferred until the alternate user ID has been unassigned.

Why Have a Service Pool?

The service pool can increase performance for an environment in which the host transactions are short. The APPC connections are severed between each transaction and the virtual machine configuration remains consistent between the transactions. This design lets a set of virtual machines remain logged on, saving initialization time between transactions with additional virtual machines logged on only on demand. This allows fast transaction initiation without causing excessive real storage contention.

A service pool provides:

- An efficient means (over conventional private servers) for accessing data existing only at one location (in a common saved segment area). As an alternative to one server, if the private server is busy, the system can go to another service pool virtual machine for service.
- A general method for supporting SNA-connected independent workstations that require many short requests for host services.

Considerations for Writing and Installing a CMR

PI

The Conversation Management Routine (CMR) is the service pool manager. When writing a CMR for your installation, you should consider the following:

- AVS uses the GCS LOAD macro to load the CMR into storage and the DELETE macro to remove it.
- The CMR must reside in a load library that was made known to GCS by a previous GLOBAL LOADLIB command.
- The linkage between AVS and the CMR is handled by the GCS LINK macro, which is similar to an assembler language CALL interface. GCS builds a parameter list of addresses pointed to by register 1 and passes it to the CMR. On return from the CMR, register 1 will point to the parameter list as modified by the CMR. See *z/VM: Group Control System* for more information on the LINK macro.
- To access the parameters that AVS passes to it using the GCS LINK macro, the CMR must be able to address any storage that AVS can address. Because it may not be able to address the parameters in the parameter list, a CMR that uses 24-bit addressing cannot be used with an AVS load library that uses 31-bit addressing. If AVS uses 31-bit addressing and the CMR uses 24-bit addressing, AVS will not activate the requested gateway.
- The main AVS task calls the main entry point of the CMR routine when selected AVS events occur. These events are described in "Interactions between AVS and a CMR" on page 327.
- The CMR must attach a new subtask to use *one-per-task* resources. These are referred to as STIMER and ESTAE exits. See *z/VM: Group Control System* for more information on the GCS ATTACH macro.
- The CMR must be able to handle the defined parameter lists that it passes to, and receives from, AVS. The parameter lists are defined in the "Interactions between AVS and a CMR" on page 327.

To install the CMR routine, the system administrator must include the CMR in a load library and ensure that the load library is accessible to the AVS virtual machine. The name of the load library containing the CMR must also be specified on a GCS GLOBAL LOADLIB command. To associate a CMR with a gateway, use the MANAGER operand of the AGW ACTIVATE GATEWAY command. See "Purpose" on page 271 for more information.

Interactions between AVS and a CMR

When AVS communicates with the CMR by the GCS LINK macro, the address of an event-specific parameter list, in OS format, is passed in register 1. This section describes the various events and the parameter lists.

Note that several of the lists have output parameters. The CMR must provide values for these parameters before returning control to the AVS virtual machine.

The LU names (gateway_lu_name and partner_lu_name) that appear in many of the parameter lists are in the following format: *netid.luname*. The *netid* is optional; if specified, it must be from 1 to 8 characters long. If the *netid* is omitted, the connecting period (.) is also omitted. The *luname* is required, and is from 1 to 8 characters long. If the LU name is less than 17 characters long, it is padded on the right with blanks. The gateway_lu_name corresponds with the local LU's fully-qualified LU Name, and the partner_lu_name corresponds with the remote LU's fully qualified LU Name.

The CMR may use the user word that appears in each parameter list in any way that is needed. Information passed to AVS in the user word is stored and passed back to the CMR in the parameter list for the next event.

ACTIVATE Event

The Activate event is initiated when an AGW ACTIVATE GATEWAY command is successfully issued. The CMR is loaded into GCS private storage by the GCS LOAD macro. AVS then issues the GCS LINK macro to call the CMR and send the Activate event parameter list. If the command fails and the CMR was loaded, a GCS DELETE macro is issued to remove the CMR.

Field	Length	Contents
'ACTIVATE'	8 chars	Event name
gateway_name_len	1 byte	Length of gateway LU name
gateway_lu_name	17 chars	Gateway LU name
userword	4 bytes	User data

DEACTIVATE Event

A Deactivate event starts when an AGW DEACTIVE GATEWAY command is successfully issued or if AVS cannot find the conversation specified by the CMR at the completion of the Deallocate event. The latter situation is described under “Queued Allocates” on page 331.

When the AVS gateway is deactivated, AVS links to the CMR with the Deactivate event parameter list. If the deactivation is caused by an error with a queued allocate, the conversation_id contains the VTAM conversation identifier that the CMR passed to AVS at the conclusion of a Deallocate event.

Field	Length	Contents
'DEACTIV '	8 chars	Event name
gateway_name_len	1 byte	Length of gateway LU name
gateway_lu_name	1-17 chars	Gateway being deactivated
conversation_id	4 bytes	VTAM conversation ID causing error deactivate (if 0, a deactivate command was issued)
userword	4 bytes	User data

INBOUND ALLOCATE Event

An Inbound Allocate event occurs when VTAM presents an ATTN(FMH5) to AVS. The allocate originated from a transaction program in the SNA network.

When AVS processes this allocate, it sends the Inbound Allocate event parameter list to the CMR by the GCS LINK macro. On return, the CMR must provide information about the service pool machine to which the request is routed, an optional alternate user ID, and a completion flag, which indicates if AVS should proceed with this allocation.

Field	Length	Contents
'INALLOC '	8 chars	Event name

Field	Length	Contents
gateway_name_len	1 byte	Length of gateway LU name with allocate
gateway_lu_name	1-17 chars	Name of gateway LU with allocate
partner_name_len	1 byte	Length of partner LU name
partner_lu_name	1-17 chars	Name of partner LU
mode_name	8 chars	Logon mode name
TPN_length	1 byte	Length of TPN
TPN	1-64 chars	Transaction program name
access_userid_len	1 byte	Length of access user ID
access_userid	1-10 chars	Access user ID (if length is 0, no user ID was specified)
VTAM_conv_id	4 bytes	VTAM conversation ID
VTAM_session_id	8 bytes	VTAM session ID
LU_name_qualifier	8 bytes	Output: APPC/VM LU name qualifier must be '*USERID' to have connection proceed as a private server connection, but may be changed to *IDENT or CS gateway name
target_LU_name	8 bytes	Output: LU name to use on connect
change_id	8 chars	Output: ID for alternate user ID (if this field contains blanks, an alternate user ID was not specified)
completion_flag	4 bytes	One of the following output values: **00000000** Complete connection to target **00000004** Reject conversation; allocation error TRANS_PGM_NOT_AVAIL_NO_RETRY **00000008** Reject conversation; allocation error TRANS_PGM_NOT_AVAIL_RETRY **0000000C** Queue the allocation request for later retry when indicated by CMR after a deallocate event
userword	4 bytes	User data

Note: Any other completion flags will cause the Inbound Allocate to be rejected. The following error message is also displayed:

```
"INVALID COMPLETION FLAG FROM CONVERSATION MANAGER cmrname".
```

OUTBOUND ALLOCATE Event

An outbound allocate originates from a transaction program in the local TSAF collection; the remote transaction program is on another system in the SNA network. The connection is routed through AVS and sent to the remote transaction program by VTAM.

When AVS is about to issue an outbound allocate, it notifies the CMR by the GCS LINK macro. The Outbound Allocate event parameter list notifies the CMR that a conversation originated in the local collection.

Field	Length	Contents
'OUTALLOC'	8 chars	Event name

Field	Length	Contents
gateway_name_len	1 byte	Length of gateway LU name with allocate
gateway_lu_name	1-17 chars	Name of gateway LU with allocate
partner_name_len	1 byte	Length of partner LU name
partner_lu_name	1-17 chars	Name of partner LU
mode_name	8 chars	Logon mode name
TPN_length	1 byte	Length of TPN
TPN	1-64 chars	Transaction program name
access_userid_len	1 byte	Length of access user ID
access_userid	0-10 chars	Access user ID (if the length is 0, the access user ID was not specified)
vm_conv_id	4 bytes	z/VM conversation ID
userword	4 bytes	User data

DEALLOCATE Event

A deallocate is when a conversation is severed. When a connection from AVS to a service pool machine is severed, AVS sends the Deallocate event parameter list to the CMR by the GCS LINK macro. Upon return, the CMR may specify a different VTAM conversation ID, which refers to a queued allocate that should be retried at this time. See "Queued Allocates" on page 331 for more information.

Field	Length	Contents
'DEALLOC '	8 chars	Event name
vm_conv_id	4 bytes	z/VM conversation ID
vtam_conv_id	4 bytes	VTAM conversation ID
userword	4 bytes	User data

ATTENTION LOSS Event

AVS is notified of an Attention Loss by VTAM when a session between your local LU and the partner LU has ended. When the session is finished, AVS notifies the CMR by sending the Attention Loss event parameter list by the GCS LINK macro.

Field	Length	Contents
'ATTNLOSS'	8 chars	Event name
vtam_exit_data	record	Parameter list passed from VTAM
acb_addr	4 bytes	VTAM ACB address
reserved1	4 bytes	Reserved
reserved2	4 bytes	Reserved
vtam_subexit	4 bytes	Name of VTAM subexit: "LOSS"
rpl_addr	4 bytes	Address of R/O VTAM RPL
reserved3	4 bytes	Reserved
userword	4 bytes	User data

ABEND Event

If an irrecoverable AVS error occurs, the CMR is notified by the GCS LINK macro with the Abend event parameter list.

Field	Length	Contents
'ABEND '	8 chars	Event name
userword	4 bytes	User data

Queued Allocates

If the CMR finds that all the service pool virtual machines are busy on an inbound allocate request, the CMR can request AVS to queue the allocation request for later retry. The sequence of events for a queued allocate is:

1. AVS receives an inbound allocate from VTAM.
2. AVS links to the CMR with the inbound allocate parameter list.
3. The CMR finds all the service pool virtual machines busy and sets the completion_flag of the returned parameter list to X'0000000C'. The CMR requests that AVS queue the allocation request to be retried later.
4. AVS queues the allocation request until a later deallocation event.
5. After a deallocation event, the CMR will have an available service pool virtual machine. Before returning the deallocate parameter list to AVS, the CMR changes the vtam_conv_id to the ID of the conversation the had previously been queued. This indicates to AVS that the allocation should be retried using the vtam_conv_id returned by the CMR.
6. After receiving the returned deallocate parameter list from the CMR, and finding the vtam_conv_id changed, AVS attempts to locate the queued allocation request corresponding to the vtam_conv_id from the CMR.
7. If the allocation fails because AVS was unable to find the queued request:
   - AVS issues an error message that indicates that the gateway deactivation is caused by problems with the CMR
   - The gateway is deactivated
   - The CMR is sent the deactivate event parameters with conversation_id set to the vtam_conv_id that could not be matched with a queued allocation request.
8. If AVS finds the queued request, the allocation is retried.

Note: When an allocation is retried, AVS gives it priority over other inbound events.

PI end

Appendix D. POSIX Security Values and Connectivity

This section describes POSIX security values.

Introduction

Security in a POSIX environment is based on three entities: the effective UID (eUID), the effective GID (eGID), and the supplementary GIDs. These values are used by resource providers in a POSIX environment to verify that a process is authorized to make use of the resource in question. These values, along with the real UID (rUID) and real GID (rGID), make up the POSIX security fields discussed in this section.

In order for a server to be able to use the POSIX security fields for resource access control, there must be a mechanism which allows the server to extract the current values of the POSIX security fields. Specifically, the byte file system server must be able both to verify the POSIX security values in effect for a process which is attempting to access a file, and to establish a new set of POSIX security values for a process which is attempting to issue an *exec()* call to execute a file in a POSIX file system. Because of security considerations, the POSIX security fields must be maintained and updated by CP.

For more information on POSIX security values, their use for resource access control, and other CP control mechanisms for these values, refer to the *z/VM: CP Planning and Administration*.

File Access

Secure file access in a POSIX environment is provided by associating a privilege level with each UID or GID which may access a file. Processes which do not have a matching UID or GID may only access the file if a privilege level has been established for world (that is, any process) access.

In VM, POSIX-compliant file access is provided by the byte file system, implemented by an SFS server. Communications between an SFS client (including byte file system clients) and the SFS server are implemented by a pair of APPC/VM connections. This transparently allows the client and server to be on different VM images. Since the byte file system is a POSIX file system it requires the POSIX security fields to validate a client's access to files. To accomplish this, CP adds the POSIX security fields to the information which is provided on every APPC/VM connect.

However, how the VM systems are connected together makes a difference in determining the source of the security data that is provided to the server. In general, a server that resides on the same system as the requestor, or one accessible via either ISFC or TSAF (a member of the CS or TSAF collection) is provided with the POSIX security values for the user's currently-active process. If there is no such process, or the server is accessible through AVS, the user's database POSIX security values are provided. There are exceptions to this generalization; refer to the APPC/VM CONNECT description in the *z/VM: CP Programming Services* for details on this and its relationship to access security.

The version of the VM system on which the target resource resides, or through which the connection must pass, may also affect the information supplied to the server. For details, refer to the APPC/VM CONNECT description in the *z/VM: CP Programming Services*.

File Execution

POSIX provides a method whereby an executable file in a POSIX file system may be associated with a UID and/or GID so that the eUID and/or eGID of the process attempting to issue an *exec()* call to execute the file are changed to the values associated with the file. Files which have this attribute are known as set_UID/set_GID files.

In VM these files must reside in the byte file system, and a communication method must be defined by which an authorized byte file system server may instruct CP to change the POSIX security values for a client process.

To this end, several requirements need to be satisfied before the application can successfully execute a set_UID/set_GID file. The complete list of requirements can be found in the *z/VM: CP Planning and Administration*; those specific to connectivity are repeated here.

1. The requestor and the server must reside either on the same VM system or in the same CS collection. File execution of set_UID/set_GID files is not supported across TSAF or AVS.
2. The UID and GID values must define the same security authorizations at the server as they do at the client. In order for the byte file system to be able to be accessed by a similar set of users as the current SFS filepools, the scope of UID and GID values is defined as the CS or TSAF collection in which the client and server reside. This implies that the system administrator is responsible for allocating UID and GID values such that there is no UID value which identifies more than one distinct user or set of users in the collection. This is known as having a flat UID space, or more generically, a flat name space. The same requirement for a flat name space is levied on GIDs. This requirement is analogous to the existing requirement to maintain a flat userid space for a CS or TSAF collection. (The need to maintain a flat name space is due to the fact that the byte file system appears to be a local file system to all nodes in the collection.)
3. The server must use extensions to the existing *IDENT interface to notify CP that it wishes to participate in these operations, and to communicate with CP *exec()* processing. Refer to the *z/VM: CP Programming Services* for details on this interface.

Appendix E. Secure APPC Communications

APPC communications can occur between any two APPC-capable systems: mainframe-to-mainframe, mainframe-to-workstation, or workstation-to-workstation. Some of these configurations provide better security than others. If two systems communicate using a physically secure connection, you can be certain of the identity of the system at either end of the connection. Communicating across a Local Area Network (LAN) or Wide Area Network (WAN), on the other hand, may legitimately raise concerns about the identity of the communications partners.

As in all communications, APPC application programmers and network administrators must be sensitive to the security needs of their customers. Both have a responsibility to understand the two primary aspects to APPC security: authentication and authorization.

What Is Authentication?

Authentication verifies the identity of a particular client and is the responsibility of the client LU, the server LU, or both. All identifying information should[2] be authenticated:

- The access security user ID and password on a SECURITY(PGM) conversation must be verified by the server LU.
- The access security user ID sent by the client LU on a SECURITY(SAME) conversation must have been previously verified by the client LU. This is called an **already-verified** user ID. To protect VM from unintentional security exposures, AVS requires explicit permission from the AVS administrator to accept already-verified user IDs from a client LU.
- The identity of the client LU itself. Before accepting already-verified user IDs from a client LU, the two LUs should exchange encrypted session keys to confirm their identities to each other. This process is called **LU-LU verification**.

  LU-LU verification may be used without regard to the conversation security level, though the APPC architecture only requires it for SECURITY(SAME).

Because authentication is the responsibility of the partner LUs, not the application, only the LUs have access to access security passwords or session keys.

Once authentication is complete, the server application program is notified of the client request and can then perform any needed authorization functions.

What Is Authorization?

Authorization verifies that a client may access the requested resource (a file or database, for example) in the requested way or manner (read, write, add, change, delete, and so on). Authorization is the responsibility of the server application program. The application performs this function by using the client user ID and client LU name provided by the server LU. The server application must never depend on any data sent to it by the client application for user identification since application data can easily be corrupted.

Because client authentication is the responsibility of the LUs, the server application should not make any further attempt to authenticate the client.

If a server application needs to determine the NJE node name (for example, RSCS or JES2) of the client, it should do so by using the client LU name and LU name, not by getting the node ID or other identifying information from the application data stream.

[2] "Should" because while the APPC architecture requires it, actual implementation is at installation discretion.

The Importance of Unique User IDs

The access security user ID is most often the only basis for user authentication and authorization, so it is critical that every user has his or her own unique user ID. No two users should have the same user ID.

Even though two users with the same user ID are defined on different systems, server applications cannot differentiate between the two.

Within a TSAF or CS collection, a user may connect to any resource within the collection using SECURITY(SAME). For this reason, it is vital that no two users be assigned the same user ID. If the uniqueness of user IDs is not assured, some users may be able to access information that they have not been given permission to access. For more details on defining us unique user IDs, see “Defining Unique User IDs” on page 173.

Using SECURITY(SAME) Safely

To use SECURITY(SAME), you must trust the client LU. This trust is achieved in one of two ways:

1. A physically secure dedicated connection between the two LUs, or
2. LU-LU verification.

With the widespread use of LANs and WANs, dedicated communication links may not be practical, and in such an environment it is trivial for one workstation to masquerade as or "spoof" another workstation or host. To counter this, LU-LU verification must be used.

If the partner workstation or host LU is not secure from tampering, do not accept SECURITY(SAME) conversations from that LU.

Once you trust the LU's identity, you must make a decision about its administration. It is, again, trivial for the informed workstation user to generate local user IDs (by using User Profile Management on OS/2, for example). Therefore, allow only specific user IDs (probably only one) from a trusted LU. **Do not use** the "all users" form of AGW ADD USERID, except where the client host or workstation is controlled by a trusted party who selects user IDs from the same "pool" as you do.

Applying these few rules will help to ensure that your APPC applications are not compromised by accident or by malicious intent.

Appendix F. Creating a VSwitch Controller

An additional VSwitch controller might be required for QDIO uplinked VSwitches, which utilize OSA-Express adapters. EQDIO uplinked VSwitches, which utilize Network Express adapters, do not use a VSwitch controller.

When an additional VSwitch Controller is required, use one of the existing controllers DTCVSW1, DTCVSW2, DTCVSW3 or DTCVSW4 as an example. The example below depicts the creation of a new VSwitch Controller DTCVSWA, based on the preconfigured controller DTCVSW1.

1. Using DTCVSW1 as a guide, add a directory entry for the new controller (DTCVSWA) and format and allocate a 5 cylinder 191 A disk.
2. Log on to the new controller and copy the PROFILE EXEC from TCPMAINT for the new controller:

   ```
   link tcpmaint 591 591 rr
   acc 591 e
   copyfile tcprofil exec e profile exec a (olddate
   release e
   ```

3. After the PROFILE EXEC is copied to the VSwitch controller, log off controller.
4. Log on to TCPMAINT to set up the server configuration file for the new controller. Ensure minidisk 591 is accessed as the e disk and the 198 minidisk is accessed as the d disk. Both were already set as needed. If not, you need to access them.

   ```
   acc 591 e
   acc 198 d
   ```

5. Copy each of the sample server configuration files:

   ```
   copyfile dtcvsw1 stcpip e dtcvswa tcpip d (olddate
   ```

6. The controller should now be ready. Log off from the TCPMAINT user ID and start the controller with XAUTOLOG.
7. From a user id with class B privilege, issue the QUERY CONTROLLER command to insure the new controller is operational.
8. Add the VSwitch controller to the process you use to start things automatically at IPL. They should be started early in the process and before the VSwitches are defined.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (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

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 jurisdictions 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.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain 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 actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate 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.

Programming Interface Information

This document primarily documents information that is NOT intended to be used as Programming Interfaces of z/VM.

This document also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

PI

<...Programming Interface information...>

PI end

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on IBM Copyright and trademark information (https://www.ibm.com/legal/copytrade).

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268

- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- Open Systems Adapter-Express ICC 3215 Support (https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support), SA23-2247
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- *z/VM: Performance*, SC24-6301. See z/VM Performance Data Pump.
- *z/VM: Other Components Messages and Codes*, GC24-6300. See Data Pump Messages.

RACF Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ickug00_v2r5.pdf), GC35-0033

Environmental Record Editing and Printing Program

- Environmental Record Editing and Printing Program (EREP): Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc2000_v2r5.pdf), GC35-0152
- Environmental Record Editing and Printing Program (EREP): User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc1000_v2r5.pdf), GC35-0151

Related Products

XL C++ for z/VM

- *XL C/C++ for z/VM: Runtime Library Reference*, SC09-7624
- *XL C/C++ for z/VM: User's Guide*, SC09-7625

z/OS

IBM Documentation - z/OS (https://www.ibm.com/docs/en/zos)

Index

Special Characters

*IDENT
    definition 28
*IDENT (Identify System Service)
    identifying
        gateways 246, 314
        multiple connections to 313
    in communications directory 180
    in resource's locally known LU name 170
    specified as LU name qualifier 170
*USERID
    in communications directory 181
    in resource's locally known LU name 170
    specified as LU name qualifier 170
$SERVER$ NAMES file
    description of 175
    overview 167
    sample entries 176, 200
    setting up 175–177
    tags 176

A

abend event parameter list 331
abends
    AVS 295
    TSAF 239
access port
    definition 18
access security
    checking with adjacent TSAF and CS collections 303
    field 26
    for connections outside of collection 177
    identifying who can connect to private resources 176
    information
        description of 27
        in APPCPASS statement 181
        in communications directory 180
        in CP directory 181
        ways to specify 178
    resolution of information in APPCPASS statement 182
    summary of use in connections 319
    tag in communications directory 179
    types
        and gateways 256
        description of 27
        specified by user programs 177
        specified in communications directory entry 180
accessibility of resources 166
ACCOUNT1 parameter, SYSTEM_USERIDS statement 235
accounting records
    AVS
        AGWACI ASSEMBLE 251
        conversation 291
        generating 291
        initialization 291
accounting records *(continued)*
    AVS *(continued)*
        termination 293
    ISFC 309
    SYSTEM_USERIDS statement 235, 291
    TSAF
        generating 235
        initialization 235
        link 237
        session 236
        termination 237
ACCT directory option 189, 235, 247, 291
ACF/VTAM 11
activate event parameter list 328
ACTIVATE GATEWAY command (AVS), AGW 271
ADD LINK command (TSAF) 217
ADD USERID command (AVS), AGW 273
Advanced Program-to-Program Communications (APPC)
    access security 27
    communications
        between collection and network 11
        in single z/VM system 7
        summary 13
        within a collection 8
Advanced Program-to-Program Communications/VM (APPC/VM)
    description of 6
aggregation, isolation and link 61
AGW ACTIVATE GATEWAY command (AVS) 271
AGW ADD USERID Command (AVS) 273
AGW CNOS command (AVS) 274
AGW DEACTIVE CONV command (AVS) 277
AGW DEACTIVE GATEWAY command (AVS) 278
AGW DELETE USERID Command (AVS) 279
AGW prefix 253
AGW QUERY command (AVS) 280
AGW QUIESCE command (AVS) 285
AGW SET ETRACE command (AVS) 286
AGW SET ITRACE command (AVS) 287
AGW START command (AVS) 288
AGW STOP command (AVS) 289
AGWACI ASSEMBLE file 251
AGWPROF GCS 253, 288
AGWTAB ASSEMBLE file 249
AGWTUN ASSEMBLE file 251
AGWTUN TEXT file 252
alternate user ID 326
APPC/VM VTAM Support (AVS)
    accounting file 251
    AGWACI ASSEMBLE 251
    AGWPROF GCS 253
    AGWTAB ASSEMBLE file 249
    AGWTUN ASSEMBLE file 251
    AGWUME message repository 248
    AGWUME REPOS 248
    AGWUMEB message repository 248
    AGWUMEB REPOS 248

- APPC/VM VTAM Support (AVS) *(continued)*
  - autologging 253
  - centralizing operations 194, 253
  - coexisting with CS collection 303
  - gateways
    - authorization to connect to 178, 311
    - authorization to manage 314
    - defining maximum number of sessions for 250
    - defining to VTAM 249
    - definitions with CS collection coexistence 304
    - description 255
    - directory control statements for 315
    - identifying multiple gateways 315
    - identifying to collection 311, 314
    - name space with TSAF and CS collections 303
    - overview 32, 255
  - installation 245
  - interface with CMR 326
  - LOADLIB 248
  - message repositories 248
  - overview of 11
  - performance tuning 251
  - profile for 253
  - setting up 245
  - tuning parameters 251
- APPCCMD statement (VTAM) 275
- APPCPASS directory statement 181, 263
- APPL definition statement 249
- application program major node 249
- assigning
  - domain controller names 301
  - unique resource names 173
  - unique user IDs 173
- asterisk in $SERVER$ NAMES file 177
- ATSLINKS FILE
  - using ADD LINK to manipulate 217
  - using DELETE LINK to manipulate 220
- ATSUME REPOS file 192
- ATTACH command (CP) 86, 193
- attention loss event parameter list 330
- authorization
  - connect to gateways 178, 311
  - connect to resources 178, 311
  - default, to access private resources 177
  - gateway example, explicitly 317
  - manage gateways 314
  - to manage gateways 246, 314
  - to manage resources
    - global, system, local 174
    - global/local 174
    - private 174
  - TSAF collection example, explicitly 315, 316, 318
- AUTHUSER macro 245
- autologging
  - AVS virtual machine 253
  - GCS in the AVS virtual machine 247
  - private resource server virtual machine 167, 175
- automatic OSA device allocation 124
- avoiding partitioned collections 210, 307
- AVS virtual machine
  - accounting file 251
  - accounting records
    - conversation 291
    - generating 291
- AVS virtual machine *(continued)*
  - accounting records *(continued)*
    - initialization 291
    - termination 293
    - types of 291
  - AGWACI ASSEMBLE 251
  - AGWPROF GCS 253, 288
  - AGWTAB ASSEMBLE file 249
  - AGWTUN ASSEMBLE 251
  - AUTHUSER macro 245
  - autologging 253
  - changing
    - contention winner/loser 274
    - session limit 274
  - commands
    - AGW ACTIVATE GATEWAY 271
    - AGW ADD USERID 273
    - AGW CNOS 274
    - AGW DEACTIVE CONV 277
    - AGW DEACTIVE GATEWAY 278
    - AGW DELETE USERID 279
    - AGW QUERY 280
    - AGW QUIESCE 285
    - AGW SET ETRACE 286
    - AGW SET ITRACE 287
    - AGW START 288
    - AGW STOP 289
    - overview 269
    - prefix 252
  - contention winner/loser values 274
  - data on 280
  - defining
    - gateways 246, 314
    - gateways to VTAM 249
    - logon mode table 249
    - to GCS 245
  - definitions with CS collection coexistence 304
  - description of 245
  - diagnosing problems
    - collecting error information 295
    - dumps 295
    - system trace data 296
    - trapping entries (TRSOURCE) 296
    - viewing TRSOURCE data 296
  - directory
    - ACCT option 247, 291
    - COMSRV option 247
    - gateway manager 314
    - IPL GCS PARM AUTOLOG 247
    - IUCV *IDENT GATEANY GATEWAY REVOKE 246
    - IUCV *IDENT gatewayid GATEWAY 315
    - IUCV ALLOW 246
    - MAXCONN option 247
    - modifying 246
    - sample 247
  - gateways
    - deactivating 278
    - defining 271
    - defining maximum number of sessions for 250
    - defining to VTAM 249
  - getting status on AVS 280
  - identifying to VTAM 249
  - installation 245, 248
  - invoking 288

- AVS virtual machine *(continued)*
  - IUCV *IDENT 246, 314
  - OPTION directory control statement 247
  - performance
    - AUTHEXIT parameter on APPL statement 249
    - tracing considerations 286, 287
    - tuning parameters 251
  - profile for 253
  - PROFILE GCS 252
  - querying (AGW QUERY) 280, 296
  - quiescing 285
  - session pacing count parameters 249
  - setting
    - contention winner/loser 274
    - external tracing 286
    - internal tracing 287
    - session limits 274
    - storage for internal trace table 288
    - up 245
    - up for remote operations 194, 253
    - up profiles 252, 253
  - starting 252, 288
  - status on 280
  - stopping
    - communication through gateway 285
    - communications through a gateway 278
    - conversations 277, 278
    - disruptively 289
    - nondisruptively 285
  - tuning parameters 251

B

- binary synchronous lines (BSC) 207
- BSC lines (binary synchronous lines) 207
- building a multi-VSwitch LAG configuration 125

C

- centralizing operations 194, 253, 310
- CETI group, dedicating links 193
- changing
  - contention winner/loser 274
  - session limit 274
- channel-to-channel adapter (CTCA) 207
- characteristics of resources
  - CS collection 28
  - description 167
- CMR (conversation management routine) 325
- CMS commands
  - SET AUTOREAD 175
  - SET COMDIR 183
  - SET COMDIR ON 183
  - SET FULLSCREEN 175
  - SET SERVER ON 175
- CNOS command (AVS), AGW 274
- CNOS verbs 26, 274
- CNOS, determining settings of 282
- coexistence with other communication protocols 303
- collaborative load balance management 133
- collection
  - examples 195, 202
- Collection Resource Management System Service (*CRM)
- Collection Resource Management System Service (*CRM) *(continued)*
  - authorizing a virtual machine to connect to 188
  - IUCV directory entry 189
  - TSAF virtual machine identifying to 196
- collection, TSAF
  - as viewed by SNA network 32
  - avoiding partitioned collections 210, 307
  - coexisting with CS collection 303
  - communicating with AVS virtual machine 317
  - connection to
    - global resource 196
    - private resource 200
    - SFS file pool 198
  - description of 28
  - determining
    - gateways in 223
    - nodes in 225
  - duplicate resource or gateway name resolution 210
  - example of different links 208
  - examples 208
  - explicitly authorized 315, 318
  - formation (example) 195
  - formation of 187
  - fully connected 208
  - initialization 187
  - joining 196, 210, 228
  - link performance 212
  - links definition file 190, 194
  - links for 207
  - merging 196, 210
  - migrating to CS collection 304
  - overview 208
  - program communications with network (overview) 11
  - program communications within (overview) 8
  - querying 223
  - querying (TSAF QUERY) 241
  - reliable routes 208
  - representation to SNA network 255
  - sharing resources within 28
  - size resolution 210
  - SNA network view of 255
  - structure 195
  - system resource table 197
- COMDIR command (CMS), SET 183
- commands
  - AVS
    - AGW ACTIVATE GATEWAY 271
    - AGW ADD USERID 273
    - AGW CNOS 274
    - AGW DEACTIVE CONV 277
    - AGW DEACTIVE GATEWAY 278
    - AGW DELETE USERID 279
    - AGW QUERY 280
    - AGW QUIESCE 285
    - AGW SET ETRACE 286
    - AGW SET ITRACE 287
    - AGW START 288
    - AGW STOP 289
    - prefix 252
    - summary of 269
  - CMS
    - SET AUTOREAD 175
    - SET COMDIR 183
    - SET COMDIR ON 183

commands *(continued)*
  CMS *(continued)*
    SET FULLSCREEN 175
    SET SERVER ON 175
  CP
    to operate VM domain controller 309
    TRSOURCE 241, 296
  GCS
    GLOBAL LOADLIB 252
    LOADCMD 253
  prefix (AVS) 253
  TSAF
    ADD LINK 217
    DELETE LINK 220
    QUERY 222
    RUNTSAF 227
    STOP TSAF 233
    summary of 215
  VTAM
    APPCCMD 275
    APPL 249
    VBUILD 249
communication
  between programs
    in collection 8
    in collection and network 11
    in single z/VM system 7
  overview of 3, 13
  summary 319
  summary of z/VM support for 13
  terminology 15, 35
communications
  Transmission Control Protocol/Internet Protocol (TCP/IP) 4
communications directory (CMS)
  and APPCPASS statement 181
  defining names of 183
  enabling 183
  IBM level 179
  ICOMDIR file 179
  in collection-wide location 200
  mode name setting for communication servers 180
  sample entries 181, 182, 263, 266
  SCOMDIR file 179
  summary of use in connections 319
  symbolic destination name resolution process 181
  SYSPROF EXEC 183
  system level 179
  tags 179
  UCOMDIR file 179
  updating 183
  user level 179
communications gateway
  deactivating 278
  establishing 271
communications partner 31, 165
communications server
  AVS 247
  in a domain 29
  mode name setting in communications directory 180
  TSAF 189
completion flags, allocation event 329
COMSRV directory option 180, 189, 247
CONCEAL directory option 190
conditions, determining for
  AVS 280
  TSAF 222
configure an IVL domain 125
configure shared port group to global VSwitches 127
configure the exclusive port group to a vswitch 121
configure the partner switch 120, 125
configuring a TSAF collection 187, 195
connecting to
  authorization, IUCV statement 311
  gateways, authorization to 178, 311
  private resource server virtual machines 312
  resources, authorization to 178, 311
connections
  identifying who can connect to private resources 176
  inbound
    activating 271
    deactivating 278
    summary 322
    through dedicated private gateway (example) 259, 261
    through global gateway (example) 257, 259
    through nondedicated private gateway (example) 261, 263
    to global resource (example) 257, 259
    to global/local resource 170
    to private resource 170
    to private resource (example) 259, 263
  outbound
    activating 271
    deactivating 278
    locally known name 170
    LU name qualifier 170
    summary 321
    target LU name 170
    through gateway (example) 263, 267
    to resource (example) 263, 267
    to SFS file pool (example) 265, 267
  request
    identifying resource name in 169
    using locally known LU name 169
    using symbolic destination name 180, 200, 263–265
  summary 319
  to file pool from outside the collection 265
  within TSAF collection
    to file pool 198
    to global resource 196
    to private resource 200
connectivity description, z/VM 3
connectivity to the IVL network 139
considerations
  AVS performance 251
  migration from TSAF to CS collection 304
  TSAF performance 212
  writing and installing CMR 327
CONSOLE control statement 194
console log, TSAF 240
contention
  CNOS command 274
  description of 26
  loser
    description of 26
    setting value 275

contention *(continued)*
  winner
    description of 26
    setting value 275
  winner/loser
    changing settings 274
    determining setting of 282
contents of
  $SERVER$ NAMES file 176
  AGWPROF GCS file 253
  AGWTAB ASSEMBLE 249
  AGWTUN ASSEMBLE file 251
  APPCPASS directory statement 181
  ATSLINKS FILE 194
  AVS message repository 248
  AVS virtual machine directory 247
  communications directory file 179
  private resource names file 176
  PROFILE GCS file 252
  TSAF message repository 192
  TSAF virtual machine directory 190
Continuously Executing Transfer Interface (CETI) group, dedicating 193
control statements, directory
  CONSOLE 194
  IUCV
    *CRM 189
    *IDENT 174, 246, 314
    AVS virtual machine 246, 314
    examples 315, 318
    for gateways 314
    gateway 246, 311, 314
    gatewayid 311, 314
    requester virtual machine 177, 178, 311
    resource 312
    resourceid 312
    server virtual machine 174, 175
    syntax for gateways 314
    syntax for global, system, local resources 174
    TSAF virtual machine 188
    userid 312
  multiple IUCV 313, 315
  OPTION 189, 246
  OPTION MAXCONN 174, 175
controller, domain
  defining unique names 301
  overview of 29
CONV command (AVS), AGW DEACTIVE 277
conversation
  accounting record, types 291
  CONV command (AVS), AGW QUERY 282
  deactivating 277
  DEACTIVE CONV command (AVS), AGW 277
  description of 24
  determining ID of 281
  determining status of 282
  new, preventing through a gateway 278
  path status between VTAM and VM 283
  security 26
  stopping 277
conversation management routine
  linkage between AVS and CMR 327
  main entry point 327
  writing a CMR 326
count parameters, session pacing 249
CP command summary for ISFC 309
CPI (Common Programming Interface) Communications 3
CPU ID, specifying 195
create a shared port group 126
create an exclusive port group 120
creating
  dumps 240
  TSAF collection 195
creating a VSwitch controller 337
CS collection
  adjacent TSAF collection
    name space of resources 303
    security authorization checking 303
    system definitions 304
  coexisting with other communications 303
  domain
    overview of 29
  domain controller
    defining unique names 301
    joining with remote domain controller 301
    overview of 29
  joining collections 301
  overview 28
  planning configuration of 301
  planning for 301, 308
  structure 301
CTCA (channel-to-channel adapter) links 207

D

deactivate event parameter list 328
deactivating
  autologged private resource server virtual machine 167
  AVS virtual machine 285, 289
  conversation 277
  gateway 278
  links 220
  tracing
    external (AVS) 286
    internal (AVS) 287
  TSAF virtual machine 233
DEACTIVE CONV command (AVS), AGW 277
DEACTIVE GATEWAY command (AVS), AGW 278
deallocate event parameter list 330
dedicating links to TSAF 193
default authorization to access private resources 177
define a global VSwitch 142
define global VSwitch(s) 125
defined gateway, deactivating 278
defining
  application program major node 249
  AVS internal trace table 288
  AVS load module 248
  AVS virtual machine to GCS 245
  communications directory entries 179
  domain controller names 301
  gateways
    to AVS virtual machine 271
    to collection 314
    to VTAM 249
  global, system, local resources 174
  links
    to domain controllers 302

defining *(continued)*
links *(continued)*
to TSAF virtual machine 192, 194
logon mode table 249
names of communications directories 183
private resources 176
private resources to the collection 174
requester user ID and password in communications directory 180
requester user ID and password in its CP directory 181
resource name in connection request 169
target of connection request 169
unique
domain controller names 301
resource names 173
unique user IDs 173
definition file, TSAF links 194
definition statement, APPL 249
definitions of connectivity terms 15
DELETE LINK command (TSAF) 220
DELETE USERID command (AVS), AGW 279
deploying an IVL network 139
determining
AVS settings 280
conversation status 282
link status 223
nodes in collection 225
QUERY command (AVS), AGW 280
QUERY command (TSAF) 222
TSAF collection status 222
when autologged private resource server virtual machine are logged off 167
DIAGNOSE code X'4C' 237
DIAGNOSE code X'98' 189
diagnosing problems
AVS 295
dumps 241, 295
TSAF 239
difference between a port based and user based virtual switch
definition 21
directory (virtual machine)
APPCPASS statement 181
AVS 246
CONSOLE control statement 194
entry for private resource server virtual machine 175
gateway owner 311, 314
global, system, local resource managers 174
multiple global/local resource owner 313
private resource server virtual machine 175
requester user ID and password information 181
resource owner 312
TSAF 188
user program user ID and password information 181
directory network authorization (DNA)
definition 22
disabling
autologged private resource server virtual machine 167
AVS virtual machine 285, 289
communications directory 183
conversation through a gateway 277
external tracing 230, 286
gateway 278
internal tracing 287
disabling *(continued)*
links 220
TSAF virtual machine 233
displaying
AVS information 280
dump information 240, 295
ISFC information 309
trace records 241
TSAF link information 222
TSAF routing information 223
domain
overview of 29
domain controller
defining unique names 301
overview of 29
Dump Viewing Facility 240, 295
dumps
AVS 295
TSAF 239
duplicate resource or gateways name resolution 210
dynamic configuration
CS collections 28
inoperative TSAF links 210
route failure 210

E

egress rules
definition 19
egress rules, virtual switch 65
enabling
AVS virtual machine 288
communications directories 183
external tracing 230, 286
gateways 271
internal tracing 287
private resource processing 175
TSAF virtual machine 227
entries in
$SERVER$ NAMES file 176
ATSLINKS file 194
in communications directories 179
system resource table 197
EQDIO 5, 16
error rate of lines 213
establishing communications gateway 271
Ethernet 207
ETRACE command
AGW QUERY (AVS) 281
AGW SET (AVS) 286
QUERY (TSAF) 222
event parameter list, AVS
abend event 331
activate event 328
attention loss event 330
deactivate event 328
deallocate event 330
inbound allocate event 328
outbound allocate event 329
example of
$SERVER$ NAMES file entries 176, 200
ADD LINK command (TSAF) 217
AGW ACTIVATE GATEWAY command 253, 257, 259, 261, 263

example of *(continued)*
- AGW CNOS command 253, 258, 259, 261, 263
- AGWPROF GCS 253
- APPCPASS directory statement 182, 263
- APPL definition statement 249
- avoiding partitioned collections 307
- AVS virtual machine directory 247
- communications directory entries 181, 182, 200, 263, 266
- connection to SFS file pool
  - from the SNA network 265, 267
  - from within collection 198, 200
- connection within collection
  - to global resource 196
  - to private resource 200
  - to SFS file pool 198
- control statements (IUCV) for
  - to access resources 313, 315
  - to use gateways 315
- DELETE LINK command (TSAF) 220
- explicitly authorized
  - gateways 317
  - TSAF collection 315, 316
- inbound connection to
  - global resource 257, 259
  - private resource 259, 263
- links in a collection 208
- logon mode table 249
- migration from TSAF to CS collection 305
- outbound connection
  - to global resource 263, 265
  - to SFS file pool 265, 267
- PROFILE GCS 252
- QUERY commands (TSAF) 223
- SCOMDIR NAMES file entries 181, 182, 263, 266
- symbolic destination name usage
  - for connection outside of collection 263, 266
  - for connection within collection 200
- SYSTEM_IDENTIFIER statements 302
- TSAF console log 240
- TSAF virtual machine activities 195
- TSAF virtual machine directory 190
- UCOMDIR NAMES file entries 181
- VBUILD statement 249

example of sharing a LAG within the same z/VM system 129
exchanging information between TSAF virtual machines 196
exclusive LAG configuration 119
exclusive link aggregation (default) 117
exclusive link aggregation vs multi-VSwitch link aggregation 116
exclusive port group 115
Exclusive Port Group
- definition 20

execs
- AGWPROF GCS 253
- PROFILE EXEC, TSAF 191
- SYSPROF EXEC 183

explicitly authorized
- gateways 317
- private resource users 175
- TSAF collection 315, 316

external tracing
- AVS 296
- displaying setting 223, 281

external tracing *(continued)*
- during AVS application initialization 288
- setting 286
- TSAF 241

F

failover, virtual switch 67
failure of routes 210
features of resources, summary 168
field, access security 26
file pool (SFS)
- sample connection outside collection 265
- sample connection within collection 198

files
- AGWACI ASSEMBLE 251
- AGWPROF GCS 288
- AGWTAB ASSEMBLE 249
- AGWTUN ASSEMBLE 251
- AGWTUN TEXT 252
- ATSLINKS FILE 190, 192
- ATSUME REPOS 192
- message repositories 192, 248
- SYSTEM NETID 195

formatting trace records 241
forming a CS collection 301
forming a TSAF collection 187, 195
fully-connected CS collection 304
fully-connected TSAF collection 208
fully-qualified LU name 169

G

gateway
- activating 271
- authorization to
  - connect to 178, 311
  - manage 314
- deactivating 278
- deactivating conversation on 277
- defining
  - contention winner/loser values 274
  - external tracing 286
  - internal tracing 287
  - maximum number of sessions for 250
  - session limit values 274
  - to VTAM 249
  - using AGW ACTIVATE GATEWAY command 271
- description of 32, 35, 255
- determining
  - status 280
  - those in a collection 223
- directory control statements for 315
- displaying information 280
- examples of
  - defining to VTAM 249
  - inbound connections through 257, 263
  - IUCV *IDENT statement 246, 314
  - IUCV gatewayid statement 311, 314
  - outbound connections through 263, 267
- global
  - example of 257
  - general description 33, 255

- gateway *(continued)*
  - global *(continued)*
    - inbound connection through (example) 257, 259
    - outbound connection through (example) 263, 267
  - identifying
    - multiple gateways 315
    - to collection 311, 314
    - to VTAM 249
  - IUCV *IDENT 246
  - name
    - defining 278
    - in APPL statement 249
    - in communications directory 179
    - in locally known LU name 169
  - on APPL statement 249
  - overview 255
  - private
    - determining which to use 256
    - general description 33, 255
    - inbound connection through dedicated (example) 259, 261
    - inbound connection through nondedicated (example) 261, 263
  - revoking 278
  - setting session limits for 274
  - status of 280
  - types 32
- GATEWAY command
  - AGW ACTIVATE (AVS) 271
  - AGW DEACTIVE (AVS) 278
  - AGW QUERY (AVS) 280
  - QUERY (TSAF) 223
- GCS (Group Control System)
  - AGWPROF GCS 253
  - AUTHUSER macro 245
  - autologging 253
  - autologging in AVS virtual machine 247, 253
  - commands to load AVS 252
  - defining AVS virtual machine to 245
  - description of 11
  - GLOBAL LOADLIB 252
  - LOADCMD 248
  - PROFILE GCS 252
- getting
  - AVS virtual machine information 280
  - gateway information 280
  - status on AVS 280
  - TSAF link information 222
  - TSAF routing information 223
- global gateways
  - activating 271
  - deactivating 278
  - example of 257
  - general description 33, 255
  - inbound connection through (example) 257, 259
  - outbound connection through (example) 263, 267
- global resource
  - access to 29
  - accessibility 166
  - characteristics of 167
  - connection to within collection (example) 196, 200
  - description of 29
  - determining ones in collection 225
  - directory control statements for 313
- global resource *(continued)*
  - identifying 313
  - inbound connection to (example) 257, 259
  - IUCV statements for 313
  - locally known LU name for 170
  - LU name qualifier for 170
  - name space 166, 256
  - outbound connection to (example) 263, 267
  - registration 167
  - revoking 313
  - same name as local 30
  - sample communications directory entry 180
  - setting up server virtual machine 174
  - summary of features 168
  - target LU name for 170
- global TSAF functions 212
- global virtual switch 135
- global VLAN ID
  - definition 19
- Global VSwitch
  - definition 18
- gracefully stopping AVS 285
- Guest LAN
  - advantages 41
  - attributes
    - changing 81
    - viewing 80
  - configuration considerations 49
  - configuring
    - attributes, changing 81
    - attributes, viewing 80
    - automating on IPL 81
    - example 80
    - guest LAN segment, creating 78
    - lifetime, establishing 78
    - network interface, configuring 79
    - owner, establishing 78
    - simulated NIC, coupling to guest LAN 79
    - simulated NIC, creating on each virtual machine 79
  - defining in SYSTEM CONFIG 82
  - definition 17
  - environment 45
  - in virtual networks 6
  - persistent guest LAN 79
  - planning for 45
  - transient guest LAN 79
  - Troubleshooting 147
  - using physical hardware components 44
  - versus Virtual Switch 43
- Guest LAN sniffing
  - methods 147
- Guest Port Traffic Forwarding Strategies
  - VEPA mode 60
  - virtual switch 57
- Guest Promiscuous Mode 149

H

- HELP for AVS commands 269
- HELP for TSAF commands 215
- HiperSockets
  - defining IOCP 86
  - definition 17
  - iQDIO protocol 85

HiperSockets *(continued)*
maximum transmission unit (MTU) 86
network characteristics 40
setting up 85
using in z/VM environment 85
verifying connectivity 87
HiperSockets bridge capable port
definition 21
HiperSockets bridge port
definition 20
HiperSockets guest port
definition 21
host definitions
defining for OSA-Express 84

I

IBM level communications directory 179
ICOMDIR NAMES communications directory 179
identifying
gateway name in connection request 170
links
in a TSAF collection 217
in CS collections 302
links in a TSAF collection 194
resource name in connection request 169
TSAF virtual machine 195
idle private resource server virtual machine 167
IEEE 802.3 207
immediately stopping AVS 289
improving performance of
AVS 251
TSAF 212
inbound allocate event parameter list 328
inbound communications
activating 271
deactivating 278
summary 322
through dedicated private gateway (example) 259, 261
through global gateway (example) 257, 259
through nondedicated private gateway (example) 261, 263
to global resource (example) 257, 259
to private resource (example) 259, 263
independent load balance management 133
ingress rules
definition 19
ingress rules, virtual switch 65
initialization accounting record 235, 291
installing
AVS 248
CMR routines 327
TSAF 187
Inter-System Facility for Communications (ISFC)
accounting records 309
APPC communications 9
command summary 309
migrating from TSAF collection 304
operating with VM PWSCF 301
interface takeover 84
intermediate communications server 180
intermediate server 27
internal tracing
internal tracing *(continued)*
AVS 296
displaying setting 281
during AVS application initialization 288
setting 287
introduction to service pool support 325
invoking
AVS 288
TSAF 227
IOCP
defining for HiperSockets 86
IP address, using the same in different VLANS 66
IP layer networks 66
IP mode considerations, virtual switch 66
IPL GCS PARM AUTOLOG statement 247
IPv6
considerations 48
Isolation configuration strategy
virtual switch 57
ITRACE command (AVS)
AGW QUERY 281
AGW SET 287
IUCV (Inter-User Communication Vehicle) statements
*CRM directory entry 189
*IDENT
manage gateways 314
manage global, system, local resources 174
*IDENT GATEANY GATEWAY REVOKE directory entry 246
ALLOW directory entry 188, 246
example 188, 246, 315, 318
for gateways 178, 311, 314
for global, system, local resources 174
gatewayid 178
manage gateways 246
resourceid 178, 312
userid 178, 312
IVL (Inter-VSwitch Link) overview 136
IVL controls and diagnostic aids 138
IVL Domain
definition 23
IVL network
definition 23
IVL network configuration domain 140
IVL virtual switch
definition 23
IVL virtual switch priority queuing 55

J

joining CS collections 301
joining TSAF collections 195, 210, 228

L

LACP
definition 20
LAG Port Controller
definition 20
LAN (local area network) links 207
levels of
communications directories (CMS) 179
limit, session, changing 274

- limits, idle private resource server virtual machine 167
- lines, communication
  - error rate 213
  - performance 212
  - speed 213
  - types 207
- link accounting record 237
- link aggregation
  - configuration considerations 113
- Link aggregation control protocol 113
- link aggregation control protocol enhancements 124
- Link Aggregation Port Group (LAG)
  - definition 19
- link layer networks 66
- links
  - adding (ADD LINK) 217
  - ATTACH command 193
  - DEDICATE statement 193
  - defining to
    - domain controllers 302
    - TSAF virtual machine 194
  - definition file 190, 192
  - deleting (DELETE LINK) 220
  - determining status of 223
  - example of 208
  - fully-connected 208
  - inoperative 210
  - ISFC supported 302
  - performance of 212
  - querying 223, 241
  - reliable 208
  - TSAF supported 207, 212
- list tag in $SERVER$ NAMES file 176
- listing private resources 175
- Live guest relocation
  - networking considerations 89
- load balancing within the virtual switch 131
- load library (AVS) 252
- load map, creating 241
- loading
  - AVS load module 248
  - dumps 241
  - TSAF 188
- local area network (LAN)
  - links 207
- local resource
  - access to 29
  - accessibility 166
  - characteristics 167
  - description of 29
  - directory control statements for 313
  - identifying 313
  - IUCV statements for 313
  - locally known LU name for 170
  - LU name qualifier for 170
  - name space 166
  - registration 167
  - revoking 313
  - same name as global 30
  - setting up server virtual machine 174
  - summary of features 168
  - target LU name for 170
- locally known LU name
  - ADD USERID command 273
- locally known LU name *(continued)*
  - AGW QUERY USERID command 280
  - APPCPASS directory statement 181
  - for connection to resource in SNA network 170
  - for outbound connection 170
  - for resource in same collection 170
  - in requester program's CP directory entry 181
  - LU name qualifier
    - *IDENT 170
    - *USERID 170
    - for global/local resource 170
    - for resource in same collection 170
    - for resource outside of collection 170
  - purpose 169
  - specified in communications directory entry 180
  - summary of 170
  - summary of use in connections 319
  - target LU 169
- log, TSAF console 240
- logging off autologged private resource server virtual machine 167
- logon mode name 274
- logon mode table 249, 274
- loser, contention 26, 275
- LU (logical unit)
  - description of 24
  - gateway 32
  - name
    - fully qualified 169
    - locally known LU 169
    - LU name qualifier 169
    - tag in communications directory 179
    - target LU 169
  - name qualifier
    - *IDENT 170
    - *USERID 170
    - for gateway 170
    - for global/local resource 170
    - for outbound connection 170
    - for private resource 170
    - in communications directory entry 180
    - IUCV directory control statement 178
  - representation of collection 255
- LU 6.2 protocol 4, 24
- luname tag in communications directory 179

M

- MAC address protection 53
- MAC address usage
  - link aggregation configuration 52
- MAC prefix
  - configuring 51
- macprefix 52
- management scope of a load balance operation 132
- managing
  - gateways 314
  - global, system, local resources 174
  - private resource 174
- map, TSAF load 241
- MAXCONN default 175, 189, 247
- MAXCONN directory option 175, 189, 247
- maximum number of sessions for gateway 250
- media access control 50

merging collections
- CS and TSAF collections 301
- resource management in 210

message
- forwarding for ISFC 310
- processing operations, centralizing 310
- repository (AVS) 248
- repository (TSAF) 192

message examples, notation used in xxiv
migrating to a CS collection 304
mode name 25
mode name entry in communications directory 180
mode name group 250
mode table, logon 249, 274
module tag in $SERVER$ NAMES file 176
multi-VSwitch LAG configuration 118, 123
Multi-VSwitch LAG
- Configuration
- definition 20

multiple gateways 315
multiple resources 313

N

name
- access global or local with same 30
- defining communications directory 183
- space (resources) 166, 256
- unique resource 173

NAMES files
- $SERVER$
  - description of 175, 176
  - overview 167
  - sample entries 176
  - setting up 175
- $SERVER$ tags 176
- communications directories
  - description of 178
  - IBM level 179
  - ICOMDIR 179
  - sample entries 181
  - SCOMDIR 179
  - system level 179
  - UCOMDIR 179
  - user level 179
- communications directory tags 179

native VLAN ID
- definition 19

NAU (network addressable unit) 169
negotiation 27, 274
network
- addressable unit (NAU) 169

Network Express
- definition 17

network interface card (NIC)
- definition 17

Network-Express
- network characteristics 40

new conversations through a gateway, preventing 278
nick tag in
- $SERVER$ NAMES file 176
- communications directory 179

node 24
NOSYSGAT, system gateway name 302
notation used in message and response examples xxiv

O

online HELP for AVS commands 269
online HELP for TSAF commands 215
Open Systems Adapter 5, 16
operations, centralizing
- centralizing 194, 253
- for VM domain controllers 310

OPTION directory statement
- AVS virtual machine 246
- COMSRV and mode name setting 180
- requester virtual machine 175
- server virtual machine 174
- TSAF virtual machine 189

OSA adapter 5
OSA configuration, virtual switch 63
OSA-Express
- configuring modes 84
- defining host definitions 84
- definition 16
- multiple ports function 56
- network characteristics 39
- setting up 84
- using in z/VM environment 83

OSA-Express (Open System Adapter-Express)
- QDIO protocol 5

OSA-Express3
- multiple ports function 83

OSD CHPID
- definition 102

OSH CHPID
- definition 102

outbound allocate event parameter list 329
outbound communications
- activating 271
- deactivating 278
- sample communications directory entry 181, 263, 266
- summary 321
- through gateway (example) 263, 267
- to resource (example) 263, 267
- to SFS file pool (example) 265, 267

overview of
- AVS commands 269
- gateways 255
- ISFC commands 309
- TSAF commands 215
- z/VM connectivity
  - between collection and network 11
  - concepts 3
  - in single system 7
  - summary of 13
  - terms 28, 35
  - within collection 8

owning
- gateways 314
- global, system, local resources 174
- multiple gateways 315
- multiple global/local resources 313
- private resource 174

P

- pacing count parameters, session 249
- parallel sessions 24
- partitioned collections, avoiding 210, 307
- partner, communications 31, 165
- password
  - entry in communications directory for requester 180
  - specifying requester program's in CP directory 181
  - tag in communications directory 179
- path
  - speed of 212
  - status between VTAM and VM 283
- performance
  - AGWTUN ASSEMBLE file 251
  - AUTHEXIT parameter on APPL statement 249
  - AVS 286, 287
  - commands to improve 212
  - considerations
    - AVS 249, 251
    - TSAF 212
  - degradation 213
  - line 212
  - of remote paths 212
  - programs 212
  - session pacing count parameters 251
- persistent guest LAN 79
- planning
  - CS collection 301
  - CScollection 308
  - VM domain controller 309, 310
- PMTUD (path MTU discovery protocol) 53
- PMUTD
  - definition 21
- port isolation
  - routing considerations 60
- port type
  - definition 18
- Port VLAN ID (pvid)
  - definition 19
- preparing
  - AVS virtual machine 245, 254
  - communications directory 178, 183
  - global, system, local resource manager 174
  - global/local resource manager 174
  - private resource server virtual machine 174, 178
  - server virtual machine 173, 177
  - SYSPROF EXEC for use of communications directories 183
  - TSAF virtual machine 187, 194
  - user programs to connect to resources 177
  - VM domain controller virtual machine 309, 310
- preventing conversations through a gateway 277, 278
- printing TSAF dumps 240
- private gateways
  - activating 271
  - deactivating 278
  - description 33
  - determining which to use 256
  - general description 33, 255
  - inbound connection through dedicated (example) 259, 261
  - inbound connection through nondedicated (example) 261, 263
- private resource
  - access security 177
  - access to 30
  - accessibility 166
  - characteristics 167
  - connection to within collection (example) 200, 202
  - default authorization to access 177
  - description of 30
  - determining gateway to use 256
  - enabling processing for 175
  - identifying authorized users 176
  - inbound connection to (example) 259, 263
  - locally known LU name for 170
  - LU name qualifier for 170
  - name space 166, 256
  - registration 167
  - sample communications directory entry 181
  - sample communications directory entry for 200
  - server virtual machine
    - autologging 175
    - CP directory entry 175
    - environment 175
    - IUCV authorization to connect to 312
    - logging off autologged 167
    - PROFILE EXEC 175
  - summary of features 168
  - target LU name for 170
  - using dedicated/nondedicated private gateway 256
- problem diagnosis
  - dumps
    - creating (AVS) 295
    - TSAF 240
  - system trace data
    - AVS external tracing 296
    - trapping entries (TRSOURCE) 241, 296
    - TSAF external tracing 241
    - viewing TRSOURCE data 241, 296
- processing for private resources, enabling 175
- processor ID, specifying 195
- profile
  - AGWPROF GCS 253
  - AVS virtual machine, setting up 252, 253
  - private resource server virtual machine 175
  - PROFILE EXEC
    - for user level communications directory 183
    - private resource server virtual machine 175
    - SET AUTOREAD 175
    - SET FULLSCREEN 175
    - SET SERVER ON 175
    - TSAF 191
  - PROFILE GCS 252
- program
  - communications
    - APPC 4
    - between programs in a collection 8, 9
    - between programs on one z/VM system 7
    - between z/VM programs 6
    - CPI Communications 3
    - LU 6.2 protocol 4
    - SNA network 4
    - summary of z/VM support 13
    - Systems Application Architecture 3
    - terminology 15, 35
    - with programs in SNA network 11

program *(continued)*
transaction 24
Programmable Operator Facility 194, 254, 310
programming interface information 340
Promiscuous mode
activating 151
protected application environment 190
protocol, LU 6.2 24

Q

QDIO 5, 16
QDIO protocol 16
QEBSM
definition 21
QUERY command (AVS), AGW 280
QUERY command (TSAF) 222
QUERY TRSOURCE command 241, 296
queued allocates 331
Queued Direct I/O Hardware Facility (iQDIO) simulated channels 85
quickly stopping AVS
commands 269
QUIESCE command (AVS), AGW 285

R

RDEVICE statements 302
real hardware network
advantages 40
options 39
Real networking devices 89
registering private resources 176
reliable routes in TSAF collection 208
relocation information for priority queuing 56
remote LU
ADD USERID command 273
AGW QUERY USERID command 280
CNOS settings 282
CNOS value negotiation 274
DELETE USERID command 279
remote operations 194, 253
repository, message
AVS 248
TSAF 192
requester virtual machine
accessing communications directories 179
CP directory entry
APPCPASS 181
explicit authorization to connect to resources 178, 311
specifying target system security information in 181
description of 32
password, specifying in
communications directory 180
CP directory 181
preparing to connect to resources 177
setting up 165, 177, 183
user ID, specifying in
communications directory 180
in CP directory 181
resolution of
resolution of *(continued)*
access security information from APPCPASS statement 182
collection size limits 210
duplicate resource or gateways names 210
symbolic destination name 181, 182
resource
accessibility 166
authorization to
connect to 178, 311
manage global, system, local 174
manage private 174
characteristics 167
communications directory, CMS 178
connection to
inbound from the SNA network 257, 267
within TSAF collection 196, 202
description of 29
determining global, in collection 225
differences between types of 166
global 29
global and local resource with same name 30
ID
IUCV directory control statement 178
identifying
in $SERVER$ NAMES file 175
in connection request 169
in private resource server virtual machine directory 175
multiple global/local resources 313
target of request 169
to collection 312
using symbolic destination name 178
in SNA network
overview of communications with 11
sample communications directory entry for 181, 182, 263, 266
specifying locally known LU name 170
inbound connections to 257, 267
local 29
locally known LU name for 170
LU name qualifier for 170
name
for resource in SNA network 169
in IUCV directory control statement 178
unique 173
name space 166, 303
NAMES file entry
in $SERVER$ NAMES file (private resources) 175
preparing user programs to connect to 177
private 29
QUERY command (TSAF) 225
registration 167
summary of type features 168
symbolic destination name for 178
system 29
table, system 197
transaction program name 29, 169
types 29
unique 166, 173
with CS and TSAF collections 303
resource manager
description of 32
server virtual machine

resource manager *(continued)*
server virtual machine *(continued)*
environment for private 175
setting up 173, 177
response examples, notation used in xxiv
revoking
conversation 277
gateway 278
global resource 313
local resource 313
routing
dynamic configuration 209
failure 210
querying 225
selection 209
setting up reliable 208
running
AVS 269
TSAF 215
RUNTSAF command (TSAF) 227
RUNTSAF MODULE 188

S

sample of
$SERVER$ NAMES file entries 176, 200
ADD LINK command (TSAF) 217
AGW ACTIVATE GATEWAY command 253, 257, 259, 261, 263
AGW CNOS command 253, 258, 259, 261, 263
AGWPROF GCS 253
APPCPASS directory statement 182, 263
APPL definition statement 249
AVS virtual machine directory 247
communications directory entries 181, 182, 200, 263, 266
connection to SFS file pool
from the SNA network 265, 267
from within collection 198, 200
connection within collection
to global resource 196
to private resource 200
to SFS file pool 198
control statements (IUCV) for
to access resources 313, 315
to use gateways 315
DELETE LINK command (TSAF) 220
explicitly authorized
gateways 317
TSAF collection 315, 316
inbound connection to
global resource 257, 259
private resource 259, 263
links in a collection 208
logon mode table 249
outbound connection
to global resource 263, 265
to SFS file pool 265, 267
PROFILE GCS 252
QUERY commands (TSAF) 223
SCOMDIR NAMES file entries 181, 182
symbolic destination name usage
for connection outside of collection 263, 266
for connection within collection 200
sample of *(continued)*
SYSTEM_IDENTIFIER statements 302
TSAF console log 240
TSAF virtual machine activities 195
TSAF virtual machine directory 190
UCOMDIR NAMES file entries 181
VBUILD statement 249
SCIF (Single Console Image Facility) 194, 254
SCOMDIR NAMES communications directory 179
security
authorizing to connect to private resources 176
checking with adjacent TSAF and CS collections 303
conversation 26
field, access 26
level entry in communications directory 180
session 24
specified by requester 177
summary of use in connection requests 319
tag in communications directory 179
SECURITY(NONE) 27
SECURITY(PGM) 27
SECURITY(SAME) 27
SERVER command (CMS), SET 175
server names file ($SERVER$ NAMES) 177
server virtual machine
CP directory entry
for private resources 175
identifying multiple global/local resources 313
IUCV *IDENT 174
IUCV ALLOW 174, 175
OPTION MAXCONN 174, 175
description of 32
directory option 189, 247
environments 167, 175
IUCV authorization to connect to 175
logging off autologged private resource 167
setting up 165, 183
server, communications 189, 247
service
TSAF, preparing to 188
service pool environment 325
service pool parameter lists 327
servicing AVS 295
servicing TSAF 191
session
characteristics 25
defining maximum number for gateway 250
description of 24
limit, changing 274
limit, description 26
limit, determining 280
pacing parameters, send and receive 249
record 236
security 24
SET AUTOREAD command, CMS 175
SET COMDIR command (CMS) 183
SET COMDIR ON command (CMS) 183
SET ETRACE command (AVS), AGW 286
SET FULLSCREEN command, CMS 175
SET ITRACE command (AVS), AGW 287
SET SERVER ON command, CMS 175
setting
amount of storage for AVS internal trace table 288
contention winner/loser 274

setting *(continued)*
number of connections to a resource allowed 174, 175
session limit 274
up
$SERVER$ NAMES file 175, 177
AVS virtual machine profiles 252, 253
communications directory 178, 183
communications gateway 271
HiperSockets network 85
links in TSAF collection 194
logon mode table 249
OSA-Express network 84
reliable links in collection 208
requester virtual machines 165, 183
server virtual machines 165, 183
SYSPROF EXEC 183
settings in AVS, determining 280
Shared File System (SFS)
example in collection 198
example in network 265
shared port group 115
Shared Port Group
definition 20
Single Console Image Facility (SCIF) 194, 254
single session connection session limits 275
single system image MAC address
considerations 52
SNA (Systems Network Architecture)
communications with programs in collection 11
definition of 23
negotiation 27
resource located in
collection (inbound connection) example 257, 259, 261
sample communications directory entry for 181, 182, 263, 266
SNA network (outbound from collection) example 263
summary of communications between programs 13
terms
contention 26
conversation 24
conversation security types 26
logical unit 24
LU 6.2 protocol 24
LU name 169
mode name 25
nodes 24
parallel session 24
session 24
session limit 26
session security 24
transaction program 24
view of collection 32, 255
Spanning Tree Protocol 72
speed of paths 212
ssi cluster 52
START command (AVS), AGW 288
starting
AVS 252, 288
TSAF 227
statements
APPCCMD (VTAM) 275
APPCPASS 181
statements *(continued)*
APPL definition 249
CP directory
IPL GCS PARM AUTOLOG 247
OPTION (AVS) 246
OPTION (TSAF) 189
VBUILD 249
status of
AVS, getting 280
conversation paths between VTAM and VM 283
TSAF collection 222
STOP command (AVS), AGW 289
STOP TSAF command (TSAF) 233
stopping
autologged private resource server virtual machine 167
AVS disruptively 289
AVS nondisruptively 285
communications in collection over a link 220
communications over a gateway 278
conversation on a gateway 277
conversations through a gateway 278
trace record writing to storage 287
TSAF 233
summary of
$SERVER$ NAMES file fields 176
AVS commands 269
communications directory file fields 179
connectivity 319
ISFC commands 309
locally known LU name 170
resource characteristics 168
SCOMDIR NAMES file fields 179
TSAF commands 215
UCOMDIR NAMES file fields 179
z/VMsupport for connectivity 13
symbolic destination name
example of use
connection outside of collection 263, 264
connection within collection 200
resolution of 181
symptom records 241
syntax
APPCPASS directory statement 181
IUCV *IDENT (gateway) 311, 314
IUCV *IDENT (resource) 312
IUCV gatewayid 311, 314
IUCV resourceid 312
IUCV userid 312
syntax diagrams, how to read xxii
SYSPROF EXEC
SET COMDIR command 183
SET COMDIR ON command 183
setting up 183
SYSTEM CONFIG file
guest LAN, defining 82
virtual switch, defining 77
system gateways 33
system level communications directory 179
SYSTEM NETID file 195
system operations, centralizing 194, 253
system resource
access to 30
description of 30
system resource table 197

system trace data
external tracing
AVS 296
TSAF 241
internal tracing
AVS 287
query external trace setting 241, 281
trapping entries (TRSOURCE) 241
viewing TRSOURCE data 241
SYSTEM_IDENTIFIER statement 35, 301
SYSTEM_IDENTIFIER_DEFAULT statement 35, 301
SYSTEM_USERIDS statement 235, 240, 291
Systems Application Architecture 3

T

table
logon mode 274
logon mode, defining 249
system resource 197
tagged frames
definition 18
tags
in $SERVER$ NAMES file 176
in communications directory 179
take-back operations 143
take-back recovery for a Multi-VSwitch LAG configuration 131
take-over operations 143
take-over recovery for a Multi-VSwitch LAG configuration 131
take-over sequence 144
target LU
name
for global/local resources 170
for outbound connections 170
for private resources 170
in communications directory entry 180
outbound connection to (example) 263
user ID and password 177
target of connection request, identifying 169
TCP/IP
defining for a z/VM guest system in a HiperSockets network 86
QDIO 5, 16
TCP/IP profile
sample profile 85
tcpdump output 153
terminating
AVS 285, 289
TSAF 233
termination accounting records 237, 293
terminology
virtual network 15
thrashing 213
time limit, idle private resource server virtual machine 167
Token Ring links 207
topology, virtual switch 70, 71
tpn tag in communications directory 179
trace table, AVS internal, defining 288
tracing
external
AVS 296
displaying setting 281
tracing *(continued)*
external *(continued)*
during AVS application initialization 288
setting 286
TSAF 241
internal
AVS 296
displaying setting 281
during AVS application initialization 288
setting 287
trademarks 340
transaction program
in a domain 29
transaction program name
description of 24
for global/local resource 170
for outbound connection 170
for private resource 170
in communications directory entry 180
transient guest LAN 79
Transmission Control Protocol/Internet Protocol (TCP/IP)
communications 4
transmission error rate 213
Transparent Services Access Facility (TSAF)
accounting record
generating 235
initialization 235
link 237
session 236
termination 237
commands
ADD LINK 217
DELETE LINK 220
QUERY 222
RUNTSAF 227
STOP TSAF 233
summary of 215
defining links to 194
description 187
diagnosing problems
dumps 240
system trace data 241
trapping entries (TRSOURCE) 241
viewing TRSOURCE data 241
directory
ACCT option 189, 235
COMSRV option 189
CONCEAL option 190
entry 188
IUCV *CRM 189
IUCV ALLOW 188
MAXCONN option 189
sample TSAF 190
exchanging information between 196
global functions 212
installation 187
links 192
performance 212
preparing to use 187
profile exec 191
servicing 191
setting up 187
transport mode
configuring 84

transport mode, virtual switch 50
transport type
  definition 18
TRSOURCE
  trace a Guest LAN 154
  trace a virtual switch 154
TRSOURCE command 241, 296
TRSOURCE Output 158
TRSOURCE trace 148
trunk port
  definition 19
TSAF collection
  adjacent ISFC collection
    name space of resources 303
    security authorization checking 303
    system definitions 304
  as viewed by SNA network 32
  avoiding partitioned collections 210, 307
  coexisting with CS collection 303
  communicating with AVS virtual machine 317
  connection to
    global resource 196
    private resource 200
    SFS file pool 198
  description of 28
  determining
    gateways in 223
    nodes in 225
  duplicate resource or gateway name resolution 210
  example of different links 208
  examples 208
  explicitly authorized 315, 318
  formation (example) 195
  formation of 187
  fully connected 208
  initialization 187
  joining 196, 210, 228
  link performance 212
  links definition file 190, 194
  links for 207
  merging 196, 210
  migrating to CS collection 304
  overview 208
  program communications with network (overview) 11
  program communications within (overview) 8
  querying 223
  querying (TSAF QUERY) 241
  reliable routes 208
  representation to SNA network 255
  sharing resources within 28
  size resolution 210
  SNA network view of 255
  structure 195
  system resource table 197
TSAF PROFILE EXEC 191
TSAF service
  preparing to 188
TSAF virtual machine
  accounting record
    generating 235
    initialization 235
    link 237
    session 236
    termination 237

TSAF virtual machine *(continued)*
  commands
    ADD LINK 217
    DELETE LINK 220
    QUERY 222
    RUNTSAF 227
    STOP TSAF 233
    summary of 215
  defining links to 194
  description 187
  diagnosing problems
    collecting error information 239, 241
    dumps 240
    system trace data 241
    trapping entries (TRSOURCE) 241
    viewing TRSOURCE data 241
  directory
    ACCT option 189, 235
    COMSRV option 189
    CONCEAL option 190
    entry 188
    IUCV *CRM 189
    IUCV ALLOW 188
    MAXCONN option 189
    sample TSAF 190
  exchanging information between 196
  global functions 212
  installation 187
  links 192
  performance 212
  preparing to use 187
  profile exec 191
  servicing 191
  setting up 187
tuning parameters for AVS 251
types of
  access security 27
  AVS tuning parameters 251
  communications directories (CMS) 179
  CS collections 301
  domain controllers 29
  gateways 32
  links in collection 207
  resources 29

U

UCOMDIR NAMES communications directory 179
uniqueness
  resource and gateway names 210
  resource names 166
untagged frames
  definition 18
updating communications directories 183
UPLINK NIC 48
UPLINK port
  definition 20
user ID
  ADD USERID command 273
  AGW QUERY USERID command 280
  DELETE USERID command 279
  entry in communications directory 180
  entry in requester virtual machine's CP directory 181
  in $SERVER$ NAMES file 176

user ID *(continued)*
in APPCPASS statement 182
IUCV control statement 178
requester, in communications directory 180
unique 173
used in access security 26
user level communications directory 179
user program
defining target of request 169
description of 32
specifying access security type 177
userid tag in communications directory 179
userprefix 52

V

values for AVS, displaying 280
VBUILD statement 249
VEPA mode 60
verification of IVL network connectivity 140
verify exclusive LAG configuration 121
verify of shared LAG configuration 127
view of collection, SNA network 255
Virtual LAN (VLAN)
definition 18
Virtual Network
security 148
virtual network, z/VM
advantages 42
guest LAN intercommunication 6
options 41
virtual switch
OSA configuration considerations 54
virtual Switch
configuration considerations 49
Virtual Switch
advantages 41
attributes
changing 77
viewing 77
BRIDGEPORT 48
configuration considerations
egress rules 65
failover 67
guest configuration considerations in IP mode 66
ingress rules 65
OSA configuration 63
topology 70
transport mode 50
using the same IP address in different VLANs 66
virtual machine routing for QDIO 71
VLAN awareness 62
VLAN support 62
configuring
access to the virtual switch, configuring 75
attributes, changing 77
attributes, viewing 77
automating on IPL 77
example 76
network interface, configuring 76
simulated NIC, coupling to the virtual machine 76
simulated NIC, creating 75
defining 74
defining in SYSTEM CONFIG 77

Virtual Switch *(continued)*
definition 17
environment 46
link aggregation 49
OSA configuration considerations 54
planning for 45
Troubleshooting 147
versus guest LAN 43
Virtual Switch Controller
definition 18
virtual switch controller configuration strategy: OSA-Express 54
virtual switch link aggregation 113
virtual switch management 57
Virtual Switch Priority Queuing
definition 23
virtual switch priority queuing function 55
Virtual Telecommunications Access Method (VTAM)
AGW CNOS command effects 274
APPCCMD statement 275
APPL definition statement 249
autologging 253
conversation, deactivating 277
conversation, status and information 281
defining gateways to 249
logon mode table 249
path status between VM 283
session pacing count parameters 249
VBUILD statement 249
VLAN awareness, virtual switch 62
VLAN ID 1
considerations 62
VLAN support, virtual switch 62
VLAN unaware
definition 18
VM domain controller
centralizing operations 310
defining
links to 302
name of 301
description of 309
link definitions 302
planning for 309, 310
planning links connected to 302
system gateway name 302
VM Personal Workstation Communication Facility (VM PWSCF)
required support with CS collection 301
VMBAT mode name setting 180
VMINT mode name setting 180

W

winner, contention 26, 275
writing a CMR 327
writing trace records
stopping 287
to storage 286, 287

Z

z/VM connectivity
communications partners 31

z/VM connectivity *(continued)*
overview of
APPC/VM 6
between programs in collection 8
between programs in collection and network 11
between programs on one z/VM system 7
between z/VM programs 6
summary of 13
terms
collection 28
gateway 32
requester virtual machines 32
resource 29
resource manager 32
server virtual machine 32
TSAF collection 28
user program 32
z/VM guest system
defining for TCP/IP in a HiperSockets network 86
zSeries HiperSockets
defining TCP/IP 86

Product Number: 5741-A09

Printed in USA

SC24-6267-73