OS/390®



SecureWay Security Server Firewall Technologies Guide and Reference

SC24-5835-05

OS/390®



SecureWay Security Server Firewall Technologies Guide and Reference

SC24-5835-05

Note

Before using this information and the product it supports, be sure to read the information under Appendix I, “Notices” on page 469.

Fifth Edition (September 2000)

This edition, SC24-5835-05, applies to Version 2 Release 10 of OS/390 SecureWay Security Server (program number 5647-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC24-5835-04.

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

IBM welcomes your comments. A form for reader's comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Information Development, Dept. G60
1701 North Street
Endicott, NY, 13760-5553
United States of America

FAX (United States & Canada): 1+607+752-2327
FAX (Other Countries):
Your International Access Code +1+607+752-2327

IBMLink (United States customers only): GDLVME(PUBRCF)
Internet e-mail: pubrcf@vnet.ibm.com
World Wide Web: http://www.ibm.com/s390/os390/

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

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

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

© **Copyright International Business Machines Corporation 1997, 2000. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

About This Book . . . xiii
OS/390 Firewall Technologies Overview . . . xiii
Who Should Use This Book . . . xiii
How to Use This Book . . . xiii
Prerequisite Knowledge . . . xiv
| OS/390 Version 2 Release 10 Enhancements . . . xiv
Entering IP Addresses . . . xiv
Conventions Used in This Book . . . xiv
| Softcopy Publications . . . xv
| Internet Sources . . . xv

| **Summary of Changes** . . . xvii

Part 1. Introduction and Configuration . . . 1

Chapter 1. Introducing the OS/390 Firewall Technologies . . . 3
Firewall Concepts . . . 3
  Protecting a Secure Network . . . 3
Filters and Address Translation . . . 4
  IP Filters . . . 4
  Network Address Translation . . . 5
  Real Audio . . . 5
Network Servers . . . 5
  ISAKMP Server . . . 5
  Configuration Server . . . 5
  fwstackd Server . . . 6
  Socks Server . . . 6
  Proxy FTP Server . . . 7
  Domain Name Server (DNS) . . . 7
Virtual Private Networks and Internet Protocol Security . . . 8
Making Your Firewall Choice . . . 9
  OS/390 Firewall Technologies Functions . . . 9
Combining the Technologies . . . 9
  Guidelines for Implementing Firewall Technology . . . 9
Security Considerations . . . 10
  Isolation of the Firewall . . . 10
  External Security Manager . . . 10
  Disabling Non-Secure Connections . . . 10
  Monitoring HFS Usage . . . 11

Chapter 2. Pre-installation Planning . . . 13
  Network Configuration Planning Worksheet . . . 13
  Prerequisite Software and Hardware . . . 14

| **Chapter 3. Migration Considerations** . . . 15
| Common Migration Actions . . . 15
| Firewall Configuration Files . . . 15
| Migration for fwtrace . . . 16
| Firewall Configuration File Migration . . . 16
| Migration from Release to Release . . . 16

© Copyright IBM Corp. 1997, 2000

| Migration from Release 4 and 5 - Starting Point . . . 16
| Migration from Release 6 - Starting Point . . . 17
| Migration from Release 7 - Starting Point . . . 18
| Migration from Release 8 - Starting Point . . . 18
| Migration from Release 9 - Starting Point . . . 18
| Release 10 Migration Information . . . 18
| Summary of Interface Changes . . . 20

Chapter 4. Configuring the OS/390 Firewall . . . 21
Step 1. Set Appropriate UNIX Services System Parameters . . . 22
Step 2. Authorize the Firewall to the External Security Manager . . . 23
Step 3. Authorize the Firewall to ICSF/MVS (Optional) . . . 27
Step 4. Configure TCP/IP on the Firewall Host . . . 28
Step 5. Copying Shell Scripts . . . 29
Step 6. Activate Sample Configuration Files . . . 30
Step 7. Define Firewall Stacks through FWSTACK Command . . . 31
Step 8. Define the Secure Interface to the Firewall . . . 31
Step 9. Configure Firewall Servers . . . 31
How Servers are Controlled . . . 31
Step 10. Enable Firewall Services and Features . . . 32
Step 11. Activate System Configuration Changes . . . 33
Step 12. Start the Firewall Kernel . . . 34
Step 13. Define Firewall Logging . . . 34

Chapter 5. Setting Up the Configuration Server and the Configuration Client (GUI) . . . 35
Configuring SSL . . . 35
Configuring the Configuration Server (CFGSRV) . . . 35
Setting Up the Configuration Client on AIX . . . 36
Moving the Client Code for AIX . . . 36
Setting Up the Configuration Client on Windows . . . 36
Log On to the Configuration Client (GUI) . . . 37
The Navigation Tree . . . 38
General Features on the Main Panel . . . 39
The Log Viewer . . . 40
Other Features of the Configuration Client Panels . . . 41
Common Fields . . . 42

Part 2. Firewall Services . . . 43

Chapter 6. Managing Firewall Logging Activity . . . 45
Logging Performed by the Firewall . . . 45
Using Logging . . . 47
Extracting Log Messages . . . 47

Chapter 7. Tracing Firewall Daemon Activity . . . 49
Trace Data Files . . . 49

Chapter 8. Controlling Traffic Through the Firewall with Filters . . . 51
Filtering Components . . . 52
Predefined Services . . . 53
Network Objects . . . 53
Defining Network Objects . . . 54
Network Object Groups . . . 54

Filter Rules . . . 55
Creating the Rule Base . . . 55
Services . . . 57
Creating a Service . . . 57
Configuring Socks Rules . . . 58
Socks with Filters . . . 58
Connections . . . 58
Ordering Connections . . . 59
Managing the Rule Base . . . 59
Filter Rule Activation States . . . 60

Chapter 9. Configuring and Using the Socks Server . . . 61
Overview . . . 61
How the Socks Server is Used . . . 61
How the Socks Server Controls Access . . . 62
Socks and Secure and Nonsecure Domains . . . 62
Starting the Socks Server . . . 62
Determining Socks Server Connections Settings . . . 63

Chapter 10. Configuring and Using the FTP Proxy Server . . . 65
Overview . . . 65
Configuring and Starting FTP . . . 65
Determining Proxy FTP Server Connections Setting . . . 66
Defining Users for Proxy FTP . . . 66
Changing User Passwords . . . 67
FTP Process . . . 67

Chapter 11. Configuring and Using the ISAKMP Server . . . 69
Overview . . . 69
Configuring and Starting the ISAKMP Server . . . 69
Example . . . 70

Chapter 12. Configuring and Using the Domain Name Service (DNS) . . . 71
Overview . . . 72
DNS on the Firewall . . . 73
How It Works . . . 73
The Recommended Configuration . . . 74
Setting Up DNS on the Firewall . . . 74
Changes to DNS Settings . . . 75
Starting the UNIX Services Name Server . . . 76

Chapter 13. Internet Protocol (IP) Security . . . 77
Benefits of a Virtual Private Network (VPN) . . . 77
IPSec Architecture . . . 77
Security Associations . . . 78
Tunnels and Key Management . . . 78
Manual Tunnels . . . 79
Dynamic Tunnels (ISAKMP) . . . 79
Choosing a Tunnel Type . . . 80
Defining Tunnel Modes . . . 80
Tunnel Policy . . . 80
VPN Customer Scenarios . . . 81
Branch Office Connection Network . . . 81
Business Partner/Supplier Network . . . 82

Remote Access Network . . . 83

Chapter 14. Dynamic Virtual Private Network (VPN) . . . 85
ISAKMP Overview . . . 85
Phases of ISAKMP . . . 85
Authentication in ISAKMP . . . 86
Steps to Defining Dynamic VPNs . . . 87
Policy . . . 88
Key Server Information . . . 92
Anchor Filter Rules . . . 95
Local Activation Information . . . 98
Summary . . . 100
Miscellaneous . . . 100
ISAKMP Main Mode Limitations . . . 101

Chapter 15. Network Address Translation . . . 103
NAT Configuration File . . . 103
NAT Configuration File Entry Interactions . . . 104
NAT Logging . . . 106

Chapter 16. Supporting the RealAudio Protocol . . . 107
How Firewall Supports RealAudio . . . 107
Configuring RealAudio . . . 107
RealAudio Web Site . . . 108

Part 3. Configuration Client Graphical User Interface (GUI) . . . 109

Chapter 17. Using the Configuration Client Graphical User Interface (GUI) . . . 111
Defining a Security Policy . . . 111
Network Objects . . . 112
Defining Network Objects . . . 112
Network Object Groups . . . 113
Determining the Rule States . . . 114
Creating Filter Rules . . . 115
Changing IP Rule Configuration Entry . . . 119
Deleting Rule Configuration Entry . . . 119
Defining Services . . . 119
Creating a Service . . . 120
Configuring the Socks Server . . . 122
Adding a New Socks Rule . . . 122
Modifying a Socks Rule . . . 124
Deleting a Socks Rule . . . 124
Building Connections Using Predefined Services . . . 125
Ordering Connections . . . 126
Activating Connections . . . 126
Configuring Archiving . . . 128
Configuring the Domain Name Server . . . 129
Configuring Tunnels . . . 130
Configuring Manual Tunnels . . . 130
Adding a Tunnel . . . 131
Changing a Tunnel . . . 138
Deleting a Tunnel . . . 139
Exporting Tunnel Definition Files . . . 139

Importing Tunnel Definition Files . . . 140
Activating a Tunnel . . . 140
Deactivating a Tunnel . . . 140
Shutting Down a Tunnel . . . 141
Building Dynamic Virtual Private Network (VPN) Connections . . . 141
Defining Key Management Information . . . 141
Defining a Key Transform . . . 141
Changing a Key Transform Configuration Entry . . . 143
Copying a Key Transform Configuration Entry . . . 143
Deleting a Key Transform Configuration Entry . . . 144
Defining a Key Proposal . . . 144
Changing a Key Proposal Configuration Entry . . . 145
Copying a Key Proposal Configuration Entry . . . 145
Deleting a Key Proposal Configuration Entry . . . 145
Defining a Key Policy . . . 145
Changing a Key Policy Configuration Entry . . . 146
Copying a Key Policy Configuration Entry . . . 147
Deleting a Key Policy Configuration Entry . . . 147
Defining Data Management Policies . . . 147
Defining an ESP transform . . . 147
Changing an ESP Transform Configuration Entry . . . 149
Copying an ESP Transform Configuration Entry . . . 149
Deleting an ESP Transform Configuration Entry . . . 149
Defining an AH transform . . . 150
Changing an AH Transform Configuration Entry . . . 151
Copying an AH Transform Configuration Entry . . . 151
Deleting an AH Transform Configuration Entry . . . 151
Defining a Data Proposal . . . 152
Changing a Data Proposal Configuration Entry . . . 153
Copying a Data Proposal Configuration Entry . . . 153
Deleting a Data Proposal Configuration Entry . . . 153
Defining a Data Policy . . . 153
Changing a Data Policy Configuration Entry . . . 155
Copying a Data Policy Configuration Entry . . . 155
Deleting a Data Policy Configuration Entry . . . 155
Dynamic Tunnel Policy . . . 155
Defining Dynamic Tunnel Policy . . . 155
Changing a Dynamic Tunnel Policy Configuration Entry . . . 156
Copying a Dynamic Tunnel Policy Configuration Entry . . . 157
Deleting a Dynamic Tunnel Policy Configuration Entry . . . 157
Defining Key Server Information . . . 157
Defining a Key Ring . . . 157
Defining a Certificate Authority . . . 158
Changing a Certificate Authority Configuration Entry . . . 158
Copying a Certificate Authority Configuration Entry . . . 159
Deleting a Certificate Authority Configuration Entry . . . 159
Defining a Key Server . . . 159
Changing a Key Server Configuration Entry . . . 160
Copying a Key Server Configuration Entry . . . 160
Deleting a Key Server Configuration Entry . . . 160
Defining Authentication Information . . . 161
Changing an Authentication Information Configuration Entry . . . 162
Copying an Authentication Information Configuration Entry . . . 162
Deleting an Authentication Information Configuration Entry . . . 162

Defining a Key Server Group . . . 162
Ordering Key Server Group Objects . . . 164
Changing a Key Server Group Configuration Entry . . . 164
Copying a Key Server Group Configuration Entry . . . 164
Deleting a Key Server Group Configuration Entry . . . 164
Defining Anchor Rules . . . 164
| Defining an On-demand VPN Connection . . . 164
Defining Locally Activated Dynamic Connections . . . 165
Defining a Dynamic VPN connection . . . 166
Changing a Dynamic VPN Connection Configuration Entry . . . 167
Copying a Dynamic VPN Connection Configuration Entry . . . 168
Deleting a Dynamic VPN Connection Configuration Entry . . . 168
Activating a Dynamic VPN connection . . . 168
Managing Activate Dynamic VPN Connections . . . 168
Configuring Network Address Translation . . . 169
Adding a NAT Entry . . . 169
Changing NAT Entry . . . 171
Deleting NAT Entry . . . 171
Activating NAT . . . 171
Creating Filter Rules for NAT . . . 173
Configuring RealAudio . . . 173

Part 4. Commands and Messages . . . 175

Chapter 18. Commands . . . 177
Command Syntax . . . 177
Syntax Notes . . . 177
Common Parameters . . . 177
Common Return Codes . . . 178
fwadapter . . . 179
fwahtran . . . 180
| fwarchive . . . 184
fwaudio . . . 187
fwauthinfo . . . 188
fwcertauth . . . 192
fwconns . . . 195
fwdaemon . . . 201
fwdatapol . . . 205
fwdataprop . . . 210
fwdns . . . 216
fwdynconns . . . 220
fwdyntun . . . 228
fwesptran . . . 231
fwfilter . . . 235
fwfrule . . . 238
fwkern . . . 247
fwkeypol . . . 251
fwkeyprop . . . 254
fwkeyring . . . 258
fwkeysrvgrp . . . 259
fwkeysrv . . . 264
fwkeytran . . . 268
fwlogmgmt . . . 272

fwlogtxt . . . 274
fwmigrate . . . 275
fwnat . . . 279
fwnwgrp . . . 284
fwnwobj . . . 288
| fwondemand . . . 292
fwsecpolicy . . . 296
fwservice . . . 298
fwsrule . . . 305
fwstack . . . 311
fwtrace . . . 314
fwtunnl . . . 316

Chapter 19. Messages . . . 329
Message Format . . . 329
Messages and Explanations . . . 330

Appendix A. Firewall Configuration Example . . . 421
Part A - Defining Network Objects . . . 422
Part B - Allow LANs in Company A to Ping Internet Addresses . . . 423
Step 1 - Create a Connection . . . 424
Step 2 - Update Filters to Activate Connection . . . 424
Part C - Configure FTP . . . 425
Step 1 - Create Connections . . . 426
Step 2 - Update Filters to Activate Connection . . . 426
Part D - Configure HTTP . . . 428
Step 1 - Create a Service . . . 429
Step 2 - Create Connections . . . 430
Step 3 - Update Filters to Activate Connections . . . 431

Appendix B. IP Security Example (Using Manual Tunnels) . . . 435

Appendix C. Dynamic Virtual Private Networking (VPN) Tunnel (ISAKMP) Examples . . . 437
Branch Office Network Using Pre-Shared Key Authentication . . . 437
Branch Office Network Using RSA Signature Authentication . . . 443

Appendix D. Alternative DNS Configuration . . . 451
Configuration Without a Secure Name Server . . . 451

Appendix E. Problem Determination . . . 453
General Problems . . . 453
Routing Problems . . . 453
DNS Problems . . . 455
Socks Problems . . . 456
FTP Problems . . . 456

Appendix F. FWTUNNL Export File Formats . . . 457
Export File format for OS/390 Version 2 Release 6 and Prior Releases . . . 457
Export File format for OS/390 Version 2 Release 7 and Release 8 . . . 459

Appendix G. SMF Record Type 109 . . . 461
Format . . . 461

Appendix H. Initializing Security Associations with ISAKMP and IKE . . . 463

Phase 1: 463
Aggressive Mode 465
Phase 2: 466

Appendix I. Notices 469
Trademarks 470

Bibliography 473
IBM Product Libraries 473
Information on the Internet 473
Redbooks 473
Other References 473

Index 475

Figures

1. A Firewall Protecting a Secure Network . . . 4
2. Firewall with IP Filtering . . . 4
3. Firewall with a Socks Server . . . 6
4. Firewall with a Proxy Server . . . 7
5. Tunnel, All IP Traffic between Two Secure Networks . . . 8
6. Network Configuration Planning Worksheet . . . 13
7. Summary of OS/390 Firewall Technologies Updates for OS/390 V2R10 . . . 19
8. Removal of Firewall Syslog Server . . . 19
9. On-demand Dynamic VPN Connections . . . 20
10. Configuration Client Logon Panel . . . 38
11. Configuration Client Navigation Tree . . . 39
12. Command Viewer Results . . . 40
13. Log Viewer . . . 41
14. Logging Performed by the Firewall . . . 45
15. Example of Log Records (Condensed Form) . . . 46
16. Example of Log Records (Formatted Form) . . . 46
17. Building Connections . . . 52
18. Name Resolution Flows . . . 72
19. DNS on the Firewall . . . 73
20. Tunnel, All IP Traffic between Two Secure Networks . . . 77
21. Branch Office Connection Network . . . 81
22. Business Partner/Supplier Network . . . 82
23. Remote Access Network . . . 83
24. Hierarchy of Key Policy Templates . . . 88
25. Relationship Between Various Key Management Templates . . . 89
26. Hierarchy of Data Management Templates . . . 90
27. Relationship Between Data Management Templates . . . 90
28. Dynamic Tunnel Example . . . 91
29. Hierarchy of a Dynamic Tunnel Template and Data Management Templates . . . 92
30. Dynamic Connection Scheme . . . 92
31. Relationship Between Key Server Groups, Key Servers and Key Management Templates . . . 93
32. Key Servers, Authentication Information, Key Ring and Certificate Authority Definitions . . . 94
33. Relationship Between Anchor Filter Rules and Dynamic Filter Rules . . . 96
34. Dynamic Tunnel Policy, Filter Rules, Services Network Objects and Connections . . . 97
35. Dynamic Connections, Network Objects, Key Servers and Key Server Groups . . . 98
36. Dynamic VPN Scheme . . . 100
37. NAT Configuration File Entry Interactions . . . 104
38. RealAudio connections through the Firewall . . . 107
39. Security Policy . . . 111
40. Add a Network Object . . . 113
41. Add a Network Object Group . . . 114
42. Add IP Rule . . . 116
43. Add a Service . . . 120
44. Add a Socks Rule . . . 122
45. Add a Connection . . . 125
46. Connection Activation . . . 128
47. Add Log Archive Definition . . . 129
48. Domain Name Service . . . 130
49. VPN Tunnel List . . . 131
50. Add a Tunnel - General Information . . . 132

© Copyright IBM Corp. 1997, 2000

51. Add a Tunnel - Encryption Information . . . 134
52. Add a Tunnel - Authentication Information . . . 137
53. Export properties . . . 139
54. Import properties . . . 140
55. Add Key Transform . . . 142
56. Add Key Proposal . . . 144
57. Add Key Policy . . . 146
58. Add ESP Transform . . . 148
59. Add AH Transform . . . 150
60. Add Data Proposal . . . 152
61. Add Data Policy . . . 154
62. Add Dynamic Tunnel Policy . . . 156
63. Configure Key Ring Parameters . . . 157
64. Add Certificate Authority . . . 158
65. Add Key Server . . . 159
66. Add Authentication Information . . . 161
67. Add Key Server Group . . . 163
68. Add On-demand . . . 165
69. Dynamic VPN Connection . . . 166
70. Dynamic VPN Connection Activation . . . 168
71. Network Address Translation List . . . 169
72. Add NAT Configuration . . . 169
73. NAT Activation . . . 172
74. Configuring RealAudio . . . 173
75. . . . 265
76. Internet Access of Company A via 2 LANs . . . 421
77. Predefined Ping Service 18 . . . 423
78. Completed Ping Filter Rules . . . 424
79. Predefined FTP Service 13 . . . 425
80. Predefined FTP Services 14 . . . 426
81. Completed Ping and FTP Filter Rules . . . 427
82. Predefined HTTP Service 2 . . . 429
83. Completed Ping, FTP and HTTP Filter Rules . . . 431
84. Branch Office Connection Network . . . 437
85. Export fwexpmctx.manual File Example (V2 R6 and Prior) . . . 457
86. Export fwexppolicy File Example (V2 R6 and Prior) . . . 458
87. Export fwexpmctx.manual File Example (V2 R7) . . . 459
88. Export fwexppolicy File Example (V2 R7) . . . 460
89. SMF Record 109 Layout . . . 461
90. Security Policy . . . 464
91. Security Policy . . . 465
92. Security Policy . . . 466

About This Book

This book describes how to configure and administer the OS/390 Firewall Technologies to prevent unwanted and unauthorized communication into or out of your secure network.

This document contains the following steps which are required to implement the OS/390 Firewall Technologies:

1. Pre-installation and planning
2. Installation
3. System and firewall configuration
4. Implementing firewall services

This book also includes configuration, commands, messages, examples and problem determination information for the OS/390 Firewall Technologies.

OS/390 Firewall Technologies Overview

| The OS/390 Firewall Technologies functions are provided to a customer in both the SecureWay Communi-
| cation Server for OS/390 and the OS/390 Security Server.

| • Functions that are included with the SecureWay Communication Server for OS/390 are:
  - IP Packet Filtering
  - IP Security (tunnels)
  - Network Address Translation (NAT)

  These functions are included as part of the IP Security feature of OS/390 and are always available.

- Functions that are included with the OS/390 Security Server are:
  - Internet Security Association Key Management Protocol (ISAKMP) support
  - FTP proxy support
  - Socks server (daemon) support
  - Command Line Configuration/Administration
  - GUI Configuration/Administration

  These functions (with the exception of Command Line Configuration/Administration) are not included as part of the OS/390 base and are only available when the OS/390 Security Server has been enabled. The Command Line Configuration/Administration support only requires that the OS/390 Security Server is installed before it becomes available.

Who Should Use This Book

This book is intended for network or system security administrators who install, administer, and use the OS/390 Firewall Technologies. It also describes how to access the firewall using client programs, such as FTP.

How to Use This Book

Read this book before you install the OS/390 Firewall Technologies on your system. It explains how to install and configure your firewall in a secure manner.

© Copyright IBM Corp. 1997, 2000

Prerequisite Knowledge

It is important that you have a sound knowledge of TCP/IP and network administration before you install and configure the OS/390 Firewall Technologies. Because you will set up and configure a firewall that controls the access in and out of your network, you must first understand some of the basics of network operation; specifically, IP addresses, fully qualified names, and subnet masks.

OS/390 Version 2 Release 10 Enhancements

In Version 2 Release 10, OS/390 Firewall Technologies has added support for On-Demand Dynamic Virtual Private Networks (VPNs). On-Demand VPNs allow an outbound Security Association (SA) to be set up automatically when the designated network traffic requires that it be transmitted securely through a VPN. Prior to Version 2 Release 10, a Dynamic VPN required manual activation to initiate the creation of a Security Association (SA) for a non-autostarted dynamic connection. The On-Demand enhancement provides an easier-to-use IKE implementation for the case where the OS/390 host is initiating the communication with another host.

Also in Release 10, the OS/390 Firewall Technologies syslog server has been removed. The SecureWay Communication Server syslog daemon is being used. The change to use the SecureWay syslog daemon will allow Firewall Administrators to utilize security enhancements made to the central syslog daemon as well as a single syslog daemon for all applications on the system.

Entering IP Addresses

When you configure your firewall, you will be asked to enter IP addresses. Enter these addresses in complete dotted-decimal format, with all 4 values:

```
n.n.n.n
```

Where each `n` is a decimal number in the range 0–255.

Note: Numbers with leading zeroes are not allowed.

Conventions Used in This Book

This book uses the following typographic conventions:

Bold **Bold** words or characters represent API names, attributes, status codes, environment variables, parameter values, and system elements that you must enter into the system literally, such as commands, options, or path names.

Italic *Italic* words or characters represent values for variables that you must supply.

`Example font`
Examples and information displayed by the system appear in `constant width type style`.

[ ] Brackets enclose optional items in format and syntax descriptions.

{ } Braces enclose a list from which you must choose an item in format and syntax descriptions.

| A vertical bar separates items in a list of choices.

... Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

**** A backslash is used as a continuation character when entering commands from the shell that exceed one line (255 characters). If the command exceeds one line, use the backslash character **** as the last nonblank character on the line to be continued, and continue the command on the next line.

Softcopy Publications

The OS/390 SecureWay Security Server library is available on the following CD-ROMs. The CD-ROM online library collections include the IBM Library Reader, which is a program that enables you to view the softcopy books.

SK2T-6718 *OS/390 PDF Library Collection*

This collection contains the set of unlicensed books for the current release of OS/390 in Portable Document Format (PDF) files. You can view or print these files with the Adobe Acrobat Reader.

SK2T-6700 *Online Library Omnibus Edition OS/390 Collection*

This softcopy collection contains a set of unlicensed books for OS/390 and related products. The collection contains the publications for multiple releases of these products.

Internet Sources

- **OS/390 Online Library**

  Softcopy OS/390 publications are also available for web-browsing and PDF versions of the OS/390 publications for viewing or printing using Adobe Acrobat Reader at the following URL:

  `http://www.ibm.com/s390/os390`

  You can also provide comments about this book and any other OS/390 documentation by visiting that URL. Your feedback is important in helping to provide the most accurate and high-quality information.

- Firewall Home Page

  You can access the latest OS/390 Firewall Technologies information and related publications at the following URL:

  `http://www.s390.ibm.com/firewall/Welcome.html`

Summary of Changes

Summary of Changes
SC24-5835-05
OS/390 V2 R10

This book contains information previously presented in the *OS/390 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835-04, which supports OS/390 V2 R8.

New Information - OS/390 V2 R10

- Enhancements to the configuration of a dynamic VPN, receiving notification that an on-demand request has been received and initiating a VPN connection.
- Removal of Security Server (Firewall) syslogd
- **New Configuration Client panels:**
  - Add On-demand
  - Add Log Archive Definition

New Commands - OS/390 V2 R10

- **fwarchive** - configures archive attributes for Firewall syslog records
- **fwondemand** - configures on-demand object to define connection granularity and key server information

Changed Information - OS/390 V2 R10

- Migration information (**fwmigrate**)

Changed Commands - OS/390 V2 R10

- **fwlog** - this command was removed
- **fwconns** - updated to allow specifying an on-demand object to be associated with a connection
- **fwdaemon** - removed support to configure the syslog server
- **fwtrace** - removed ability to trace syslog server
- **fwkern** - removed support to start, stop and query the syslog server

Changed Configuration Client - OS/390 V2 R10

- Removal of Log Facilities

This publication includes terminology, maintenance, and editorial changes, including the following:

- The OS/390 Security Server, of which OS/390 Firewall Technologies is a component, has joined the IBM SecureWay family of products. As such, occurrences of OS/390 Security Server have been changed to SecureWay Security Server for OS/390, or its abbreviated name, Security Server. OS/390 Security Server may continue to appear in messages, panel text, and other code as SecureWay Security Server for OS/390.

Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Changes
SC24-5835-04
OS/390 V2 R8

This book contains information previously presented in the *OS/390 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835-03, which supports OS/390 V2 R7.

© Copyright IBM Corp. 1997, 2000

New Information - OS/390 V2 R8

- ISAKMP support
- New Configuration Client graphical user interface (GUI) panels for building dynamic tunnels
  - AH Transform
  - Authentication Information
  - Certificate Authority
  - Data Policy
  - Data Proposal
  - Dynamic Tunnel Policy
  - Dynamic VPN Connection
  - Dynamic VPN Connection Activation
  - ESP Transform
  - Key Policy
  - Key Proposal
  - Key Ring
  - Key Server
  - Key Server Group
  - Key Transform

New Commands to Support ISAKMP

- **fwahtran** - configures AH transforms
- **fwauthinfo** - configures authentication information
- **fwcertauth** - configures certificate authority
- **fwdatapol** - configures data policies
- **fwdataprop** - configures data proposals
- **fwdynconns** - configures dynamic connections
- **fwdyntun** - configures dynamic tunnels
- **fwesptran** - configures ESP transforms
- **fwkeypol** - configures key policies
- **fwkeyprop** - configures key proposals
- **fwkeyring** - configures key rings
- **fwkeysrv** - configures key servers
- **fwkeysrvgrp** - configures key server groups
- **fwkeytran** - configures key transforms

Changed Information - OS/390 V2 R8

- Migration information (**fwmigrate**)
- Configuration information

Changed Commands - OS/390 V2 R8

- **fwdaemon** command - supports the new ISAKMP daemon
- **fwfilter** command - supports the new *predecap* option and **listactive** command
- **fwfrule** command - supports new anchor filter type
- **fwnwobj** command - supports address ranges
- **fwservice** command - supports anchor filter rules; also removed packet fragment option
- **fwtrace** command - supports debug for the ISAKMP server

- **fwtunnl** command - supports auto-activate of manual tunnels

Summary of Changes
SC24-5835-03
OS/390 V2 R7

This book contains information previously presented in the *OS/390 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835-02, which supports OS/390 V2 R6.

New Information - OS/390 V2 R7

- Supports new Configuration Client graphical user interface (GUI) and Configuration Server
- Supports more than one firewall stack
- Supports scalability for daemons
- Supports FWSTACKD daemon
- Upgraded IP Security to support RFCs 2401 - 2406 and 2410
  - HMAC_SHA - authentication algorithm
  - HMAC_MD5 - authentication algorithm
  - Triple DES (encryption)
  - Null Encryption
  - Replay Protection

New Commands - OS/390 V2 R7

- **fwdaemon** - allows the Firewall administrator to list or change daemon configuration attributes, query daemon status and start and stop daemons.
- **fwmigrate** - handles generic migration and defines an initial TCP/IP firewall stack.
- **fwstack** - manages TCP/IP Firewall stacks.

Changed Information - OS/390 V2 R7

- Migration information
- New output for **modify fwkern** command

Changed Commands - OS/390 V2 R7

- **fwadapter** command - displays data to include stack information
- **fwdns** command - supports the new stack parameter
- **fwfilter** command - supports the new stack parameter
- **fwfrule** command - fragment parameter is no longer supported
- **fwlog** command - handles new and changed facility names
- **fwnat** command - supports the new stack parameter
- **fwsrule** command - supports new values for the **identd** parameter
- **fwtrace** command - supports debug for the configuration server
- **fwtunnl** command - supports upgraded IP Security

Summary of Changes
SC24-5835-02
OS/390 V2 R6

This book contains information previously presented in the *OS/390 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835-01, which supports OS/390 V2 R5.

New Information - OS/390 V2 R6

- A new section was added that includes a table defining the `/var` file system for V2 R6, which is used for dynamic data created by various products.

- A new section, “Security Considerations” on page 10, was added that describes the actions that can strengthen the security policy.
- A new section, Chapter 13, “Internet Protocol (IP) Security” on page 77, which describes the IPSec framework for securing data traffic at the IP layer.
- A new example, Appendix B, “IP Security Example (Using Manual Tunnels)” on page 435, which shows how the IPSec support in OS/390 Firewall Technologies can be used in a non-Firewall environment.

Changed Information - OS/390 V2 R6

- An additional configuration step, “Step 5. Copying Shell Scripts” on page 29, to run shell scripts from locales that are not generated from code page IBM-1047.
- Changes to the command “fwtunnl” on page 316 to specify the operational mode of the tunnel, **transport** or **tunnel** mode.

Summary of Changes
SC24-5835-01
OS/390 V2 R5

This book contains information previously presented in the *OS/390 SecureWay Security Server Firewall Technologies Guide and Reference*, SC24-5835-00, which supports OS/390 V2 R4.

Part 1. Introduction and Configuration

© Copyright IBM Corp. 1997, 2000

Chapter 1. Introducing the OS/390 Firewall Technologies

The OS/390 Firewall Technologies is a network security "firewall" program for OS/390. In essence, the OS/390 "firewall" consists of traditional firewall functions as well as support for Virtual Private Networks. The traditional "firewall" functions act as a blockade between your intranet (a secure, internal private network) and another (nonsecure) network or the Internet. The purpose of a firewall is to prevent unwanted or unauthorized communication into or out of the secure network. The firewall has two jobs:

- The firewall lets users in your own network use authorized resources from the outside network without compromising your network's data and other resources.
- The firewall keeps users who are outside your network from coming in to compromise or attack your network.

The Virtual Private Network further enhances the level of security of the system by providing a mechanism for data to be encrypted and/or authenticated between endpoints. The OS/390 Firewall Technologies provides the facilities to manually or dynamically define a Virtual Private Network according to standards defined by the Internet Engineering Task Force (IETF).

Firewall Concepts

At one time, institutions such as hospitals, universities, banks, and insurance companies had their own computer networks. Each institution maintained an isolated network of computers and workstations wholly detached from any other computers. Then, security issues mostly revolved around preventing unauthorized access to computer files from the workstations in the network. Today, with so many networks, subnetworks, and users all connected through the Internet, the new security threat is unwanted intrusion from outside a private network over the communications lines. However, the many and varied services and repositories of information in the Internet make connection advantageous.

Normally, hosts in a secure network cannot access the outside network. This reduces the risk of being intruded upon by unauthorized users from the Internet, but denies Internet accessibility to users within the secure network. Without Internet accessibility, users in the secure network cannot avail themselves of important tools such as Telnet, FTP (File Transfer Protocol), Gopher, and WWW clients/servers to access the resources available in the Internet. Every connection invites intrusion from outside.

Even when you have taken steps to secure your network, these outside “hackers” pose a much greater threat because they have special skills and tools that they can use to break in. They will try to exploit operating system bugs and administrative errors (maybe you forgot to set a password or delete a maintenance account) that often go unnoticed within your network.

Protecting a Secure Network

Despite this continuing threat from outside, you probably still want to have limited exchanges with users outside, in the Internet, while being very selective about who you let in. The OS/390 Firewall Technologies provides a “firewall” program that does just that. It isolates your internal network from other networks in the Internet, while at the same time letting typical TCP/IP applications (such as FTP) access hosts outside in the Internet without allowing intruders access to the secure network. The OS/390 Firewall Technologies acts as a barrier between your network and the rest of the Internet, as shown in Figure 1. It makes your host a “dual-homed bastion host.” It is dual homed, because it has connections to at least two networks: the secure network and the nonsecure network. It is a bastion host because it defends your network against intrusion.

© Copyright IBM Corp. 1997, 2000

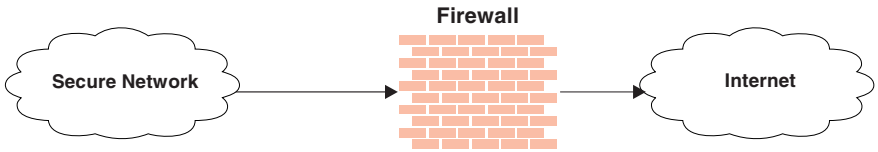


Figure 1. A Firewall Protecting a Secure Network

There are several ways a firewall can protect your network. A firewall can provide screening services that deny or grant access based on such things as user name, host name, and TCP/IP protocol. A firewall can also provide a variety of services that let authorized users through while keeping unauthorized users out, and at the same time ensure that all communications between your network and the Internet appear to end at the firewall, denying the outside world to see the structure of your network.

To control access between your intranet and the Internet, and at the same time permit authorized transactions, OS/390 Firewall Technologies provides three key technologies: network servers, filters and address translation, and Virtual Private Networks.

Filters and Address Translation

This section provides an overview and objectives of IP filters and Network Address Translation.

IP Filters

IP filters provide intranet/Internet access control by using a set of administrator-defined rules to determine if a packet arriving at or departing from the TCP/IP stack should be allowed to pass.

IP filters are tools to filter packets at the IP level, based on IP address, direction, protocol, and port. The filter rules work with the IP gateway function so the machine is required to have two or more network interfaces, each in a separate IP network or subnetwork. One interface is declared nonsecure and the other(s) declared secure. The filter acts between these two interfaces, as illustrated in Figure 2.

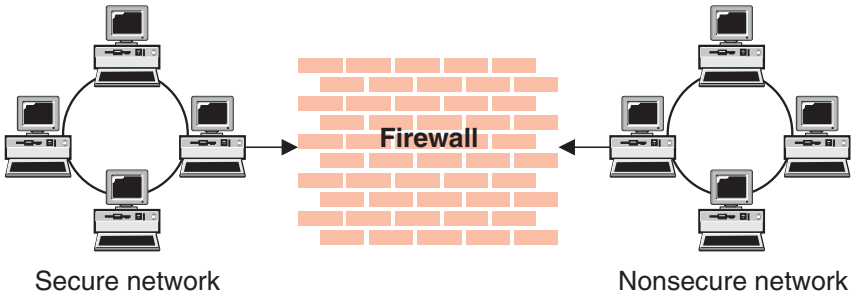


Figure 2. Firewall with IP Filtering

Objectives of the Filters: IP filtering provides the basic protection mechanism for the firewall, allowing you to determine what traffic passes across it based on IP session details, thereby protecting the secure network from external threats such as scanning for secure servers or IP address spoofing. Think of the filtering facility as the base on which the other tools are constructed.

Network Address Translation

Allows internal IP addresses to be hidden from the non-secure network, either for security reasons or because an intranet is using non-registered IP addresses. NAT maps the internal IP addresses to registered addresses, allowing UDP and TCP traffic to flow freely. IBM provides NAT for any routed TCP or UDP application packets. In addition, FTP is supported by providing translation of the IP address in the **port** command.

With the growth of the Internet, the pool of available IP addresses has become depleted. Network address translation (NAT) solves this problem by translating secure IP addresses to temporary external registered addresses in order to provide sufficient connections.

Objectives of Network Address Translation (NAT): IBM provides network address translation for any routed TCP or UDP application packets. Other applications, for example proxy FTP, use the firewall address itself as the source address. In addition, non-proxy FTP is supported by providing translation of the IP address in the **port** command. NAT is not applied to packets that are neither TCP nor UDP; for example, any routed ICMP packets that filters permit to pass will not have their source address translated.

Real Audio

RealAudio is a special protocol developed by Progressive Networks which supports live and on-demand audio from the Internet. The OS/390 Firewall Technologies is able to identify RealAudio connections and support them in a secure way without exposing the security of the private network.

Network Servers

ISAKMP Server

The ISAKMP Server, new in OS/390 V2R8, provides the capability to dynamically establish VPNs, negotiate VPN attributes, and dynamically manage VPN encryption keys. The protocols implemented by the ISAKMP Server are defined by the IETF, and include RFC 2408 (the Internet Security Association and Key Management Protocol - ISAKMP), RFC 2407 (the Internet Domain of Interpretation for ISAKMP), and RFC 2409 (the Internet Key Exchange - IKE).

Objectives of the ISAKMP Server: The ISAKMP Server offers a more automated, dynamic alternative to manual VPNs. Managing many manual VPNs can be time consuming and error prone. The ISAKMP server also automates the most important part of the VPN process: the secure and frequent exchange of encryption keys. The protocols used by the ISAKMP server are industry-standard and should interoperate with similar servers from other vendors.

Configuration Server

The configuration server is the configuration client's graphical user interface (GUI) to the Firewall. This server processes requests from the AIX, Windows 95 or Windows NT configuration clients. Once it is set up, it is considered part of the Firewall machine. The configuration client allows an authorized user to log on to the configuration server and perform specific configuration functions.

fwstackd Server

The **fwstackd** server monitors and communicates with firewall TCP/IP stacks. This includes IPSec initialization and packet filter logging. The **fwstackd** server is configured through “fwdaemon” on page 201 and started by **fwkern**. For more information, see the command “fwkern” on page 247.

Note: **fwkern** will still wait for **at least one** of the firewall stacks to initialize before starting any of the configured firewall servers (or daemons).

Socks Server

Intercepts all TCP/IP requests from `socksified` clients that may cross between your network and the Internet. However, a Socks server provides a remote application program interface so that the functions executed by client programs in secure domains are piped through secure servers at the firewall.

As with the proxy server, the Socks server intercepts the request, checks for authorization to come into or go out of the secure network, and either denies the request or pipes it through the firewall.

Socks is a standard for circuit-level gateways that does not require the overhead of a more conventional proxy server. OS/390 Firewall Technologies supports socks protocol version 4.

The Socks server results in a similar bastion configuration, since the session is broken at the firewall. The difference is that the user does not need to perform the double login manually. Instead of directly starting a session with the FTP server, the session goes to the Socks server on the firewall host. The Socks server then validates that the source address and user ID are permitted to establish onward connection into the nonsecure network and then creates the second session. Figure 3 illustrates a firewall with a socks server.

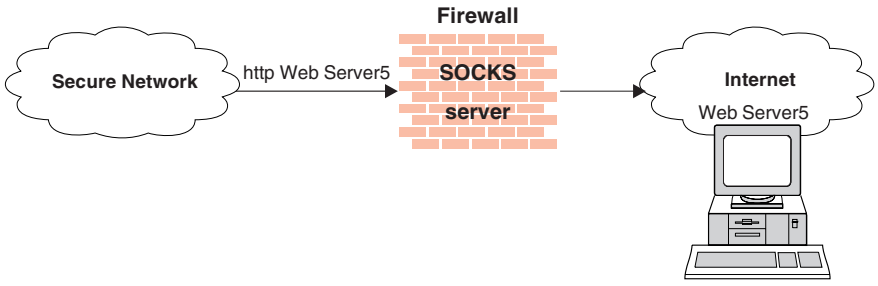


Figure 3. Firewall with a Socks Server

Socks needs to have special versions of the application code (called `socksified` clients) and a separate set of configuration profiles on the firewall. However, the server machine does not need modification; it is unaware that the session is being relayed by a Socks server.

Objectives of the Socks Server: For outbound sessions (from a secure client to a nonsecure server) the socks server has the same objectives as a proxy server, that is, to break the session at the firewall and provide a secure door where users must prove their identity in order to pass. Because the socks server eliminates the need to authenticate at the firewall host and then connect to the remote server with minimal extra administrative work, the process is simpler for the user.

Proxy FTP Server

A proxy server is a secure server that runs on the firewall and performs a specific TCP/IP function as a "proxy" for a network user. A proxy server is, in essence, an application gateway -- a gateway from one network to another for a specific network application. The FTP proxy server intercepts all file transfers and information requests made via the FTP protocol that would cross between your network and the Internet. The user contacts the FTP proxy server using FTP and provides a valid user ID and password. The FTP proxy server then asks the user for the name of the remote host to be accessed. The FTP proxy server makes contact with that remote host as a proxy for the actual user.

This tool allows an internal user, using normal client commands such as FTP, to access the nonsecure network. For example, users use FTP to access the firewall. They have to be authenticated using a password in the normal way. Having successfully accessed the firewall (and thus authenticated themselves) they now have to issue the **site** command to reach the desired machine on the nonsecure network. Figure 4 illustrates a firewall with a proxy server.

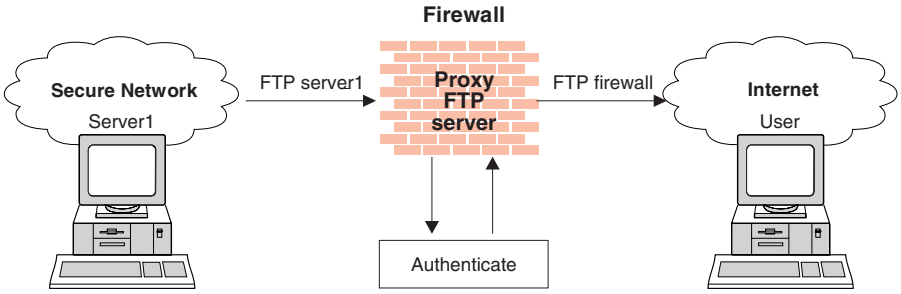


Figure 4. Firewall with a Proxy Server

This method is usable also from the nonsecure network to the secure network but raises other important security problems. If you use the Internet to connect to the firewall, you have to enter your login name and password to be identified. But you cannot know what machines your session may pass through and some intruder may be looking for login names and passwords, by wire tapping or using IP trace commands. If the intruder catches your ID, he or she could impersonate you and thus get into the organization with your identity. (This would only get the intruder to the firewall; they would need to trace the second FTP command to get past it.)

Objectives of a Proxy Server: When you connect through a proxy server, the TCP/IP connections are broken at the firewall, so the potential for compromising the secure network is reduced. Users also have to authenticate themselves.

Once connected, the appearance and the behavior are unchanged. But, the proxy server method needs a double connection: one from the client machine and one from the firewall machine. This double connection may have an impact on performance.

A major advantage of the proxy server is that you do not need a special version of the client program on the client machine. Therefore, once you have installed your firewall, every user defined in the firewall can have access to the nonsecure network without any additional software installation.

Domain Name Server (DNS)

The OS/390 domain name server can be configured to prevent users outside the secure network from seeing addresses of secure hosts, while assisting secure hosts in resolving addresses of hosts in the nonsecure network.

Objectives of the DNS Server: The objectives of running the DNS server on the firewall are to prevent name resolution requests flowing across the firewall and also to hide secure network hosts from the non-secure world.

Virtual Private Networks and Internet Protocol Security

The Internet has become a popular, low-cost backbone infrastructure. Its universal reach has led many companies to consider constructing a secure virtual private network (VPN) over the public Internet. The challenge in designing a VPN for today's global business environment is exploiting the public Internet backbone for both intra-company and inter-company communication while still providing the security of the traditional private self-administered corporate network.

Today, the internet potential is limitless and the focus has shifted to e-business, using the global reach of the Internet for easy access to key business applications and data residing in traditional I/T systems. Companies can now securely and cost effectively extend implementation of secure VPNs.

A VPN is two or more networks connected by one or more tunnels; an extension of an enterprise's private networks across a public network such as the Internet, creating a secure private connection through private tunnels. VPNs securely convey information across the Internet connecting remote users, branch offices, and business partners into an extended corporate network. In Figure 5, two private networks are each protected by a firewall. The two firewalls establish a connection between them and they encrypt and authenticate traffic passing between the private networks. Secure IP tunnels can also exist between non-IBM firewalls, an OS/390 Firewall Technologies and a non-OS/390 Firewall Technologies firewall, or between two hosts.

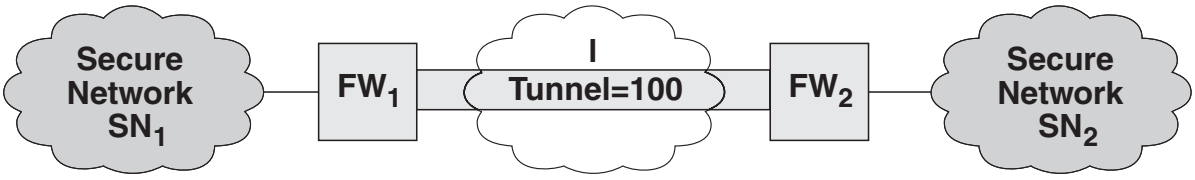


Figure 5. Tunnel, All IP Traffic between Two Secure Networks. FW_1 and FW_2 represent nonsecure interface IP address and mask. SN_1 and SN_2 represent any host in the secure network. The shaded area of the picture represents a VPN.

Standardizing this technology by the Internet Engineering Task Force (IETF) provides support for the full range of VPN scenarios with more interoperability and expansion capabilities. OS/390 Firewall Technologies Version 2 Release 8 adds dynamic tunneling, which is support for the dynamic negotiation of keys and algorithms used to create IPSec Security Associations (SAs) between systems in a VPN. This support is based on a set of open standards defined by the IETF's IP Security (IPSec) working group. As a result, OS/390 Firewall Technologies has added a new daemon to the Security Server called ISAKMPD.

Objectives of the Virtual Private Network: The virtual private network allows you to obscure the real data being sent between two private networks and to be assured of the identity of the session partners and the authenticity of the messages.

Making Your Firewall Choice

The choice of your firewall architecture depends on your security requirements and also on the size of your organization. Your firewall architecture also depends on the software you choose, and OS/390 Firewall Technologies are an excellent solution. As you read further through this book you should be able to decide in more detail which of the OS/390 Firewall Technologies are appropriate for your environment. Before going into the detail, however, it is worth stating some basic rules:

- Anything that is not explicitly permitted is, by default, denied.

  When you set up your firewall you should be able to state exactly what traffic you want to pass through it. It should not be possible for any other traffic to pass.

- Keep outside users out of your internal network whenever possible.
- Do thorough auditing and logging.

OS/390 Firewall Technologies Functions

OS/390 Firewall Technologies are tools you use to implement different firewall architectures: both screening filter and bastion. Once you choose your architecture and your security strategy, you select the necessary firewall tools to implement them.

Because OS/390 Firewall Technologies act as an IP gateway, they divide the world into two or more networks, one or more nonsecure networks and one or more secure networks. The nonsecure network is, for instance, the Internet. The secure networks are usually your corporate IP networks.

Combining the Technologies

You have now been introduced to the technologies provided by OS/390 Firewall Technologies. You may not need to use all of these technologies, but it is likely that you will use a combination of these technologies.

Guidelines for Implementing Firewall Technology

Now you need to assess your installation to define a strategy to protect your assets for allowing access to the Internet, conducting business with other companies, and performing functions within your own enterprise that warrant additional security.

Some general guidelines for implementing firewall technologies are:

- Define a policy for how you want your firewall to function to cooperate with your security group or security advisors. Before you start connecting your internal network to the Internet, decide what type of traffic is allowed through the firewall and under what conditions.
- When you configure your firewall, start by disabling everything and then enable those services you defined in your security policy. Everything that is not specifically allowed is prohibited.
- If you establish more than a single gateway between your internal network and the Internet, make sure that all gateways implement the same level of security.

  If you build a perfect firewall on one end of your network and other Internet users dial in from their LAN-attached PCs, enabling those PCs to act as IP routers between your internal network and the Internet, a hacker can exploit that back door into your network instead of breaking through your firewall.

- One of the most important features of a firewall is the ability to log both successful and rejected access events. However, daily administrative procedures must be set up to analyze and react to the information from these logs to make them useful. By analyzing the firewall logs, you can detect whether an unauthorized access was attempted and if your firewall protection was successful in rejecting such an attempt.

Although these guidelines are not conclusive, they point out some important aspects of implementing firewall technologies in your network.

Security Considerations

OS/390 Firewall Technologies is designed to act as a tool to help protect a private network. However, it alone is not enough to completely protect the network from attack. The following policies are examples of actions recommended to strengthen the security policy.

Isolation of the Firewall

The most secure method of putting up a Firewall is to isolate it on its own system or in its own logical partition (LPAR). This allows the Firewall installation to remove any services that are not required by the Firewall. The next best alternative, and possibly most reasonable approach, is to direct all incoming traffic (from the Internet) through the stacks running the Firewall. This allows applications such as Telnet and FTP to be activated, but becomes subject to the filtering rules defined on the Firewall.

External Security Manager

In many External Security Managers (ESM), user IDs can be created without specifying a password. This introduces the capability to create user IDs with a default password such as the name of the owning group, or a non-trivial installation-defined password. Even though these passwords may be initially set as expired, there is no method for preventing another user ID (that is not the intended owner) from logging in and resetting the password.

Check Installed Program Products: There are several products that create user IDs in the ESM with default passwords. We advise that the system programmers installing these products add another level of security by choosing non-trivial passwords for any user IDs required by the product.

Disabling Non-Secure Connections

Since the most common source of attacks on secure systems is the Internet, we recommend limiting the amount of access from the Internet to the secure network whenever possible. For example, we recommend that the proxy FTP server is not started with the ***-ns*** option. By doing this, the proxy FTP server will not allow connections originating on the non-secure adapter on the Firewall machine.

Similarly, limit or disallow Telnet connections from the non-secure network. This can be done by tailoring the TCP/IP profile to not allow Telnet connections or by defining the necessary filter rules in the Firewall.

Monitoring HFS Usage

If the SYSLOG server is configured to store log records in the HFS, the HFS usage must be carefully monitored. The nature of the SYSLOG server is to write data without really checking to see if the data was written. If the HFS becomes full, the log records are essentially thrown away. This can become a serious security problem in an installation that considers logging to be a critical component of their security policy. Without the logging of Firewall activity, it is possible for an intruder to either attack the system or flood the Firewall with requests.

Chapter 2. Pre-installation Planning

Network Configuration Planning Worksheet

Using the following template, specify the information as part of the planning for your OS/390 Firewall Technologies configuration. Create a worksheet for each TCP/IP stack planned to be enabled as a firewall.

Host name of firewall _____

TCP/IP stack name _____

Secure network interface(s) (connected to internal secure network):
IP address _____ Subnet Mask _____

IP address _____ Subnet Mask _____

IP address _____ Subnet Mask _____

IP address _____ Subnet Mask _____

Nonsecure network interface(s) (connected to untrusted nonsecure network):
IP address _____ Subnet Mask _____

IP address _____ Subnet Mask _____

IP address _____ Subnet Mask _____

IP address _____ Subnet Mask _____

Secure domain name _____

IP address(es) of secure domain name server(s) (DNS):
IP address _____

IP address _____

IP address _____

IP address _____

OpenEdition resolver file name (optional) _____

IP address(es) of non-secure domain name server(s) (DNS):
IP address _____

IP address _____

IP address _____

IP address _____

Registered IP addresses for NAT _____
_____

List route information _____
_____

DEFAULTNET: _____

Figure 6. Network Configuration Planning Worksheet

© Copyright IBM Corp. 1997, 2000

Prerequisite Software and Hardware

This section describes the software, hardware, and system resources needed by the OS/390 Firewall Technologies.

Software: To install and use OS/390 Firewall Technologies, you need to have the following IBM products installed:

- OS/390 V2 R10 with
  - OS/390 V2 R10 SecureWay Communications Server (For IPSec Triple DES Support the IP Security Feature is also required)
  - OS/390 V2 R10 Security Server (for firewall commands and servers)
  - OS/390 V2 R10 Security Server or other External Security Manager on OS/390 (for security management)
  - OS/390 V2 R10 Open Cryptographic Services Facilities (OCSF) (only required for ISAKMP)
  - OS/390 V2 R10 Open Cryptographic Enhanced Plug-Ins (OCEP) (only required for ISAKMP RSA Signature Mode Authentication)

Prerequisites for the Configuration Client on AIX

- Java.rte 1.1.6 or 1.1.8
- AIX 4.2 or higher (as long as Java.rte 1.1.6 or 1.1.8 is available and supported with that level of AIX)
- Netscape with Frames support

  Note: Java.rte is available on the AIX bonus pack. The level of Java.rte is a key part of the requirements.

Prerequisites for the Configuration Client on Windows 95 and Windows NT

- Windows NT 4.0 or Windows 95
- Netscape web browser with Java and frames support
- Zip tool that handles long file names such as the WinZip32 tool in WinZip

  Information about WinZip can be found at http://www.winzip.com.

Hardware: You can use any communication hardware interface supported by the TCP/IP protocol stack to make the network connections.

To be effective as a firewall program, the OS/390 Firewall Technologies should have at least two network interfaces:

- One network interface to connect to the secure, internal network that the firewall protects.
- The other network interface to connect to the nonsecure, outside network or internet.

You will want to identify these now and make note of them on the planning worksheet, because you will need these addresses during configuration.

Chapter 3. Migration Considerations

This chapter discusses migration issues. Your plan for migrating to a new level of OS/390 Firewall Technologies should include information from a variety of sources. These sources of information describe topics such as coexistence service, hardware and software requirements, migration actions, and interface changes.

The following documentation, which is supplied with your product order, provides information about installing your OS/390 system. In addition to specific information about OS/390 Firewall Technologies, this documentation contains information about all of the OS/390 elements.

- *OS/390 Up and Running!*

  This book describes the installation requirements for OS/390 at a system and element level. It includes hardware, software, and service requirements for both the driving and target systems. It also describes any coexistence considerations and actions.

- *OS/390 Program Directory*

  This document, which is provided with your OS/390 product order, leads you through the specific installation steps for OS/390 Firewall Technologies and the other OS/390 elements.

- *ServerPac Installing Your Order*

  This is the order-customized, installation book for using the ServerPac Installation method. Be sure to review "Appendix A. Product Information," which describes data sets supplied, jobs or procedures that have been completed for you, and product status. IBM may have run jobs or made updates to PARMLIB or other system control data sets. These updates could affect your migration.

Within this chapter, you can find information about the specific updates and considerations that apply to this release of OS/390 Firewall Technologies.

- "Common Migration Actions"

  This section identifies tasks that must be done for any migration. These tasks are not associated with a specific new function, but must be completed to ensure successful migration to the new release.

- "Migration from Release to Release" on page 16

  This section describes the specific updates that were made to OS/390 Firewall Technologies for each release.

Common Migration Actions

Firewall Configuration Files

All of the OS/390 Firewall Technologies sample configuration files are located in `/usr/lpp/fw/etc` and `/usr/lpp/fw/samples` directories. This is true for all releases. Do not re-copy any files into their target locations (`/etc` or `/etc/security`), because any previously changed configuration(s) will be lost. Copy only those files that do not exist in the target location. For more information see "Step 6. Activate Sample Configuration Files" on page 30.

© Copyright IBM Corp. 1997, 2000

Migration for fwtrace

- Default trace file changed from **/usr/lpp/fw/fwdata/fwtrace.data** to **/var/fw/fwdata/fwtrace.data** in OS/390 V2 R6.
- New components
  - OS/390 V2 R7 - Configuration Server (CFGSRV)
  - OS/390 V2 R8 - Internet Security Association Key Management Protocol (ISAKMP)
- Removed components
  - OS/390 V2 R10 - SYSLOGD

  The **fwtrace** command will automatically recognize any new components and new default trace file. However, if trace was active before the migration, use the following command sequence:

```
fwtrace -l off -c all
fwtrace -l on -c component_list
```

Firewall Configuration File Migration

The **fwmigrate** command handles the migration for the individual configuration files. The migration routines handle backing up the files as well as whatever migration is needed. The **fwmigrate** command converts old formats for configuration files to new formats, for example, manual tunnels.

This command is designed to be executed multiple times. As errors are found, reported, and fixed, the command will skip all previously completed steps.

fwmigrate backup files (files that existed in the previous release) are stored in `/etc/security/fwbackup`. If restoring the previous release, these files can be copied into their original location.

Migration from Release to Release

Before completing the installation, configuration, and initialization of the latest level of the OS/390 Firewall Technologies product, the following migration considerations from this section must be met. The migration considerations are sensitive to each release of the OS/390 Firewall Technologies. Migration involves starting at your current release and proceeding through changes involved with **each** subsequent release as well as the general migration items. Each section involves changes that are specific to that release.

Migration from Release 4 and 5 - Starting Point

HFS name changes: OS/390 Firewall Technologies writes dynamic data created by the various products to the `/var` file system.

The following table defines those directories and the impact on the installation.

Old Directory Name	New Directory Name
/usr/lpp/fw/home/fwkern	/u/fwkern **Note:** As of OS/390 V2 R6 of the OS/390 Firewall Technologies, this data resides in `/var/fw/fwkern`.
Impact: A home directory is required during the definition of the FWKERN user ID in the External Security Manager (ESM). Delete the FWKERN user ID and re-define it as outlined in “Step 2. Authorize the Firewall to the External Security Manager” on page 23.	
/usr/lpp/fw	/var/fw
Impact: All of the OS/390 Firewall Technologies **dynamic** data previously recorded in `/usr/lpp/fw` directories are stored in `/var/fw` directories. **Note:** All other directories in `/usr/lpp/fw` that the install jobs create during the installation process remain as they are, read-only data.	
/usr/lpp/fw/fwdata	/var/fw/fwdata
Impact: In earlier versions of the OS/390 Firewall Technologies, the `/usr/lpp/fw/fwdata` directory was defined as a separate HFS that was used (by default) to store data created using the **fwtrace** command and the SYSLOG server (or daemon). This data has been moved into `/var/fw/fwdata` (as the default). If there is data in `/usr/lpp/fw/fwdata` that is to be maintained in this release, copy it into the `/var/fw/fwdata` file system. In all future releases, the `/usr/lpp/fw/fwdata` will not be mounted as part of the installation process. **Note:** Use the **fwtrace -q** command to see if **fwtrace** data is being recorded in the `/usr/lpp/fw/fwdata` directory structure. If trace is not active, the file name recorded by the **fwtrace** command will specify ***NULL***. If this is the case, check the `/usr/lpp/fw/fwdata` directory structure to see if there is any data that should be copied into `/var/fw/fwdata`.	
/usr/lpp/fw/fwkern	/var/fw/fwkern
Impact: During the start up of the FWKERN control program, some data is written before *normal* paths are available (DDDEF's, etc.). Prior to this release, this data was stored in `/usr/lpp/fw/fwkern`. The following files are located in this directory: • `stdin` • `stdout` • `stderr` • `icapk.soc`	

Removal of FWADM User ID: Prior to OS/390 V2 R6 of the OS/390 Firewall Technologies, initializing the necessary objects in the External Security Manager included defining a user ID, FWADM. Creating this user ID is no longer necessary. An installation that wants to remove this user ID from the ESM can do so with no side effects on the OS/390 Firewall Technologies capability.

Migration from Release 6 - Starting Point

Migration for fwkern: Prior to OS/390 V2 R7 of the OS/390 Firewall Technologies, the only configuration for the Firewall servers was controlled through a limited set of variables defined in the **/etc/fwkern.conf** file.

OS/390 V2 R7 introduced a new file to store basic configuration information for each of the servers, start up options and environment variables that control the behavior of the servers.

The name of this file is **/etc/security/fwdaemon.cfg.** Copy the installed sample configuration file from **/usr/lpp/fw/etc/security/fwdaemon.cfg** into **/etc/security/fwdaemon.cfg**

If the file **/etc/fwkern.conf** exists, the firewall administrator must use the **fwdaemon cmd=change** configuration command once for each server to reconfigure the SYSLOGD, PFTPD, and SOCKD servers (daemons). The input to the **fwdaemon** command consists of data from the **/etc/fwkern.conf** configuration file.

Migration for PROCLIB:

- ICAPTNAT PROC is no longer used by FWKERN and should be removed.
- ICAPFLOG PROC is no longer used by FWKERN and should be removed.
- Two PROCs are added to the PROCLIB production environment:
  - ICAPSTAK
  - ICAPCFGS

Migration for CFGSRV and FWSTACKD: Two new servers were added in OS/390 V2 R7, CFGSRV and FWSTACKD, which have external security manager considerations. Refer to the External Security Manager section in Chapter 4, “Configuring the OS/390 Firewall” on page 21.

Migration from Release 7 - Starting Point

Migration for PROCLIB:

- A new PROC was added to the PROCLIB production environment.
  - ICAPIKED

Migration for ISAKMPD: A new server was added in OS/390 V2 R8, ISAKMPD, which have external security manager considerations. Refer to the External Security Manager section in Chapter 4, “Configuring the OS/390 Firewall” on page 21.

Migration from Release 8 - Starting Point

Migration for PROCLIB: ICAPSLOG PROCLIB is no longer used by FWKERN and should be removed.

Migration from Release 9 - Starting Point

If you are migrating from Release 9, start with the section “Migration from Release 8 - Starting Point.”

Release 10 Migration Information

This section describes the new and changed OS/390 Firewall Technologies functions that are introduced for OS/390 Version 2 Release 10 (V2R10). The information about each item includes:

- Description
- Summary of the OS/390 Firewall Technologies tasks or interfaces that may be affected
- Coexistence considerations, if any, that are associated with the item
- Migration procedures, if any, that are associated with the item
- References to other publications that contain additional detailed information .

Release Summary: Figure 7 on page 19 summarizes the updates that have been introduced to OS/390 Firewall Technologies in OS/390 Version 2 Release 10 (OS/390 V2R10). For more information, refer to the detailed section for each item.

Figure 7. Summary of OS/390 Firewall Technologies Updates for OS/390 V2R10

For Information About:	Refer to Page:
Removal of Firewall Syslog Server	Not Applicable

Removal of Firewall Syslog Server

Description: In Release 10, the OS/390 Firewall Technologies Syslog server has been removed. The SecureWay Communication Server (CS) syslog daemon is being used. The change to use the SecureWay syslog daemon will allow Firewall Administrators to utilize security enhancements made to the central syslog daemon as well as a single syslog daemon for all applications on the system.

What This Change Affects: This support affects the following areas of OS/390 Firewall Technologies processing.

Figure 8. Removal of Firewall Syslog Server

Area	Considerations
Administration	Single syslog server for all OS/390 applications
Auditing	Firewall audit records recorded by the SecureWay CS syslog daemon
Interfaces	• fwlog command removed • fwarchive command introduced. For more information see Summary of Interface Changes

Dependencies: With the removal of the Firewall Syslog server, to ensure proper auditing of Firewall activities, the administrator must start the SecureWay Communication Server syslog daemon prior to starting any firewall servers or issuing any firewall commands.

Migration Tasks: No special migration tasks are associated with this enhancement. The execution of the fwmigrate command with update the configuration information.

For More Information: For more detailed information about using the SecureWay Communication Server syslog daemon, see *OS/390 IBM Communications Server: IP Configuration Guide*.

On-demand Dynamic Virtual Private Network (VPN) Connections

Description: In Version 2 Release 10, OS/390 Firewall Technologies enhanced the Dynamic VPN function to support on-demand connections. On-Demand VPNs allow an outbound Security Association (SA) to be set up automatically when the designated network traffic requires that it be transmitted securely through a VPN. On-demand VPN connections:

- eliminate the need for manual intervention when initiating a dynamic VPN connection
- allow the VPN administrator to select which outbound traffic can result in the automatic activation of a dynamic VPN connection
- reduce the amount of configuration required for locally initiated dynamic VPN connections.

What This Change Affects: This support affects the following areas of OS/390 Firewall Technologies processing.

Figure 9. On-demand Dynamic VPN Connections

Area	Considerations
Administration	Eases administration activities for locally initiated dynamic VPN connections
Interfaces	• fwconns command enhanced • fwondemand command introduced For more information see Summary of Interface Changes

Migration Tasks: No special migration tasks are associated with this enhancement. The execution of the fwmigrate command will update the appropriate configuration information.

Summary of Interface Changes

This section summarizes the new and changed interface components of OS/390 Firewall Technologies.

- Commands
  - New Commands
    - fwarchive - Configures information needed for log archiving.
    - fwondemand - Configures information to describe an on-demand connection
  - Changed Commands
    - fwdaemon - support for configuring Firewall syslog server removed
    - fwkern - start, stop, query support for Firewall syslog server removed
    - fwmigrate - updated to handle changes needed to the configuration files
    - fwtrace - debug support for Firewall syslog server removed

Chapter 4. Configuring the OS/390 Firewall

This section explains how to configure the basic functions of the firewall on an OS/390 host. This involves the following general steps:

- Set OS/390 UNIX services system parameters that affect the firewall (“Step 1. Set Appropriate UNIX Services System Parameters” on page 22).
- Authorize the firewall to your OS/390 External Security Manager (“Step 2. Authorize the Firewall to the External Security Manager” on page 23).
- Authorize the firewall to ICSF/MVS (optional) (“Step 3. Authorize the Firewall to ICSF/MVS (Optional)” on page 27).
- Set up TCP/IP to use the firewall (“Step 4. Configure TCP/IP on the Firewall Host” on page 28).
- Copy shell scripts, if necessary (“Step 5. Copying Shell Scripts” on page 29).
- Activate sample configuration files, if desired (“Step 6. Activate Sample Configuration Files” on page 30).
- Define firewall stacks through the command “fwstack” on page 311 (“Step 7. Define Firewall Stacks through FWSTACK Command” on page 31).
- Configure the network adapter that attaches the hosts that are in your secure network (“Step 8. Define the Secure Interface to the Firewall” on page 31).
- Configure and start firewall servers, (“Step 9. Configure Firewall Servers” on page 31).
- Enable firewall services features (“Step 10. Enable Firewall Services and Features” on page 32).
- Activate system configuration changes (“Step 11. Activate System Configuration Changes” on page 33 and “Step 12. Start the Firewall Kernel” on page 34).
- Define firewall logging (“Step 13. Define Firewall Logging” on page 34).

© Copyright IBM Corp. 1997, 2000

Step 1. Set Appropriate UNIX Services System Parameters

OS/390 Firewall Technologies servers require the use of system resources such as threads, files, and sockets. The system parameters that control these resources may need to be changed to allow the servers to operate properly. Examine the following values and update them if necessary; they reside in member BPXPRMxx in SYS1.PARMLIB.

MAXPROCSYS
: is the maximum number of processes that UNIX system services will allow to be active at the same time. The default is 200. The firewall requires 11 processes to start its servers.

MAXPROCUSER
: is the maximum number of processes that a single UNIX system services user ID (UID) is allowed to have active at the same time. The default is 25. As with the MAXPROCSYS setting, the firewall requires 11 processes to start its servers.

MAXFILEPROC
: is the maximum number of files that a single UNIX system services user ID (UID) is allowed to have concurrently active or open. The default is 64. The firewall requires approximately 25 open file descriptors for firewall servers, 2 for each concurrent connection to the Socks server, and 4 for each concurrent connection to the FTP proxy server.

MAXTHREADTASKS
: is the number of OS/390 tasks created with **pthread_create** that a single user may have concurrently active in a process. The default is 50. The firewall requires approximately 10 threads for firewall servers, 1 thread for each concurrent connection to the Socks server, and 1 thread for each concurrent connection to the proxy FTP server.

MAXTHREAD
: is the maximum number of threads created with **pthread_create**, including those that are running, queued, and exited but not detached, that a single server can have currently active. The default is 200. The Socks and proxy FTP servers require 1 thread for each concurrent connection.

MAXSOCKETS
: is the maximum number of sockets that can be obtained for the given file system type. The default is 64. The firewall requires approximately 25 sockets for firewall servers, 2 for each concurrent connection to the Socks server, and 4 for each concurrent connection to the FTP proxy server.

Verify that the **AF_UNIX** and **AF_INET** are defined in the BPXPRMxx member of SYS1.PARMLIB.

Note: These file systems should be defined during the initial product installation. See the Program Directory for more information.

If the domains are not defined, add the following to the BPXPRMxx member to use OS/390 Firewall Technologies.

```
NETWORK DOMAINNAME(AF_UNIX)
         DOMAINNUMBER(1)
         MAXSOCKETS(100)
         TYPE(IBMUDS)

NETWORK DOMAINNAME(AF_INET)
         DOMAINNUMBER(2)
         MAXSOCKETS(n)
         TYPE(IBMINET)
```

where: **n** is the value determined for MAXSOCKETS (see above)

For more information about setting these parameters, see *OS/390 UNIX System Services Planning*, SC28-1890, and the *OS/390 MVS Initialization and Tuning Reference*, SC28-1752.

Step 2. Authorize the Firewall to the External Security Manager

This is a two-step process. First, define the appropriate users and groups to your ESM (RACF is used as an example in the following steps). Second, grant permissions for the users and groups you have created to the firewall objects.

The following procedures assume that you are performing them from a RACF administration ID.

Planning for Group Definition

1. Determine the current group definitions:

   Command:

   User: **LISTGRP * OMVS NORACF**

   Note: Bear in mind that two groups, SYS1 (or equivalent) and FWGRP, are required for OS/390 Firewall Technologies configuration. If these groups already exist, note their group identifiers. If not, note available group identifiers for use in the definition of the required group(s).

2. Determine whether or not the SYS1 group (or a logically equivalent group to contain UID=0 users) exists:

   Command:

   User: **LISTGRP SYS1 OMVS**

3. If the SYS1 (or equivalent) group does not exist, create it:

   Command:

   User: **ADDGROUP SYS1 OMVS(GID(*nnn*))**

   Note: *nnn* is the installation-defined group ID for the SYS1 group that was identified in the previous steps.

Defining Users and Groups

1. Define the firewall start up address space to the ESM:

   This includes:

   - Define FWGRP group
   - Define FWKERN user
   - Define the firewall start up program as a started task.

   Commands:

   User: **ADDGROUP FWGRP SUPGROUP(SYS1) OMVS(GID(*nnn*))**
   User: **MKDIR'/u/fwkern' MODE(7,5,5)**
   User: **ADDUSER FWKERN OMVS(HOME(/u/fwkern/) UID(0))**
   DFLTGRP(FWGRP) AUTHORITY(CREATE) UACC(ALTER) PASSWORD(pw)
   User: **RDEFINE STARTED FWKERN STDATA(USER(FWKERN))**
   User: **SETROPTS RACLIST(STARTED) REFRESH**

   Note: *nnn* is the installation-defined group ID for the FWGRP group that was identified in the previous steps.

   Note: 'pw' is the password for the FWKERN user ID. Choose this password with extreme care to avoid potential security exposure.

Granting Users and Groups Authority to Firewall Objects:

1. Create the FWKERN.START.REQUEST resource profile

   This includes:

   - Define the FACILITY class profile FWKERN.START.REQUEST
   - Ensure the FACILITY is active
   - Permit FWKERN update access to FWKERN.START.REQUEST

   Commands:

   ```
   User:  RDEFINE FACILITY FWKERN.START.REQUEST UACC(NONE)
   User:  PERMIT FWKERN.START.REQUEST CLASS(FACILITY) ID(FWKERN) ACCESS(UPDATE)
   User:  SETROPTS CLASSACT(FACILITY)
   ```

2. Permit FWKERN access to start the servers

   This includes:

   - Permit access to the FWKERN procedure (JCL)
   - Permit access to each of the daemon procedures (JCL)

   Commands:

   ```
   User:  RDEFINE STARTED ICAPSOCK.**  STDATA(USER(FWKERN) GROUP(FWGRP))
   User:  RDEFINE STARTED ICAPPFTP.**  STDATA(USER(FWKERN) GROUP(FWGRP))
   User:  RDEFINE STARTED ICAPCFGS.**  STDATA(USER(FWKERN) GROUP(FWGRP))
   User:  RDEFINE STARTED ICAPSTAK.**  STDATA(USER(FWKERN) GROUP(FWGRP))
   User:  RDEFINE STARTED ICAPIKED.**  STDATA(USER(FWKERN) GROUP(FWGRP))
   User:  SETROPTS RACLIST(STARTED) REFRESH
   ```

3. Permit FWKERN access to READ the TCP/IP data sets:

   Command:

   ```
   User:  PERMIT 'TCPIP.**' ID(FWKERN) ACCESS(READ)
   ```

4. Permit the PFTP server to the BPX.DAEMON facility:

   This includes:

   - Verify that the BPX.DAEMON facility exists
   - If it does not exist, create it
   - Set the permission

   Commands:

   ```
   User:  RLIST FACILITY BPX.DAEMON
   User:  RDEFINE FACILITY BPX.DAEMON UACC(NONE)
   User:  PERMIT BPX.DAEMON CLASS(FACILITY) ID(FWKERN) ACCESS(READ)
   ```

5. If program control was turned on (using SETROPTS WHEN(PROGRAM)), the firewall daemon programs must be marked as program controlled. It is also necessary to mark the SSL library SGSKLOAD as program controlled. This can be done by using the following commands:

   Commands:

   ```
   User:  :  RALTER PROGRAM * ADDMEM('ICA.SICALMOD'/'******'/NOPADCHK) UACC(READ)
   User:  :  RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'/'******'/NOPADCHK) UACC(READ)
   User:  :  SETROPTS WHEN(PROGRAM) REFRESH
   ```

6. To use the configuration server, some setup is required. This includes:

   - Define the FACILITY class profile ICA.CFGSRV
   - Permit update access to ICA.CFGSRV

Note: All user IDs that are specified on the firewall configuration GUI must be explicitly given permission to update the configuration through the configuration server. This includes user IDs that have superuser privileges or are members of the firewall group.

- Ensure that the FACILITY is active and refreshed.

  Commands:

  ```
  User:   RDEFINE FACILITY ICA.CFGSRV UACC(NONE)
  User:   PERMIT ICA.CFGSRV CLASS(FACILITY) ID(userid) ACCESS(UPDATE)
  User:   SETROPTS CLASSACT(FACILITY)
  User:   SETROPTS RACLIST(FACILITY) REFRESH
  ```

Adding Firewall Administrators to FWGRP

If the user IDs that will administer the Firewall are not superusers (UID=0), add them to the FWGRP group as follows:

Command:

```
User:   CONNECT <userid> GROUP(FWGRP)
```

7. To start the ISAKMP server the following setup is required:

   - Install and configure the Open Cryptographic Services Facility (OCSF) code as documented in *OS/390 Open Cryptographic Services Facility Application Developer's Guide and Reference*, (SC24-5875). Ensure that installation process enables the OCSF code for Program Control.
   - Install and configure the Open Cryptographic Enhanced Plug-ins (OCEP) code as documented in Chapter 2 of the *OS/390 SecureWay Security Server Open Cryptographic Enhanced Plug-ins Guide and Reference* (SA22-7429). Ensure that installation process enables the OCEP code for Program Control.
   - Permit FWKERN access to the CDS.CSSM, CDS.CSSM.CRYPTO and CDS.CSSM.DATALIB facilities in order for the ISAKMP server to use the OCSF services.

     Commands:

     ```
     User:   PERMIT CDS.CSSM CLASS(FACILITY) ID(FWKERN) ACC(READ)
     User:   PERMIT CDS.CSSM.CRYPTO CLASS(FACILITY) ID(FWKERN) ACC(READ)
     User:   PERMIT CDS.CSSM.DATALIB CLASS(FACILITY) ID(FWKERN) ACC(READ)
     ```

   - If the system operates with OS/390 UNIX security, then permit FWKERN access to the BPX.SERVER facility so the ISAKMP server can use the OCSF services.

     Commands:

     ```
     User:   PERMIT BPX.SERVER CLASS(FACILITY) ID(FWKERN) ACC(READ)
     ```

   - Permit FWKERN access to the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING facilities in order for the ISAKMP server to use the OCEP services. If the Facilities are not already defined, you must add them first.

     Commands:

     ```
     User:   RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
     User:   RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
     User:   PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(FWKERN) ACC(READ)
     User:   PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(FWKERN) ACC(READ)
     ```

   - The ISAKMP server runs as an APF authorized application and requires that the APF-authorized extended attribute be turned on for the OCSF and OCEP Dynamically Loaded Libraries (DLLs). The DLLs (.dll and .so files) in the /usr/lpp/ocsf/lib and /usr/lpp/ocsf/addins directories must have their APF-authorized extended attribute turned on by using the extattr +a command.

Commands:

```
User:   RDEFINE FACILITY BPX.FILEATTR.APF UACC(NONE)
User:   PERMIT BPX.FILEATTR.APF CLASS(FACILITY) ID(userid) ACCESS(UPDATE)
```

where *userid* is ID of the person executing the extattr command

```
User:   SETROPTS RACLIST(FACILITY) REFRESH
```

From an OMVS command prompt:

```
User:   $    cd /usr/lpp/ocsf/lib
User:   $    extattr +a *.dll
User:   $    cd /usr/lpp/ocsf/addins
User:   $    extattr +a *.so
```

Refer to the OS/390 UNIX System Services Planning book (SC28-1890) for more details.

- The ISAKMP daemon supports the ability to perform peer authentication using RSA Signature mode. RSA Signature mode requires that digital certificates be stored in RACF and connected onto a key ring. RACF provides digital certificate and key ring support via the RACDCERT command. The authorizations necessary to perform the basic RACDCERT actions are shown below.

  Commands:

  ```
  User:   RDEFINE FACILITY IRR.DIGTCERT.ADD UACC(NONE)
  User:   RDEFINE FACILITY IRR.DIGTCERT.ADDRING UACC(NONE)
  User:   RDEFINE FACILITY IRR.DIGTCERT.CONNECT UACC(NONE)
  User:   RDEFINE FACILITY IRR.DIGTCERT.GENCERT UACC(NONE)
  User:   RDEFINE FACILITY IRR.DIGTCERT.GENREQ UACC(NONE)

  User:   PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(userid) ACC(CONTROL)
  User:   PERMIT IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(userid) ACC(UPDATE)
  User:   PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(userid) ACC(CONTROL)
  User:   PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(userid) ACC(CONTROL)
  User:   PERMIT IRR.DIGTCERT.GENREQ CLASS(FACILITY) ID(userid) ACC(CONTROL)
  User:   PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(userid) ACC(CONTROL)
  User:   PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(userid) ACC(UPDATE)

  User:   SETROPTS RACLIST(FACILITY) REFRESH
  ```

  where *userid* is ID of the person who will be executing the RACDCERT command to store digital certificates.

  Refer to the Security Server (RACF) Command Language Reference (SC28-1919) for a complete description of the facilities and authorizations needed to create and modify digital certificates and key rings.

Step 3. Authorize the Firewall to ICSF/MVS (Optional)

OS/390 Firewall Technologies can take advantage of the encryption and decryption functions available in the new generation of System/390 processors. The firewall uses several of these functions in two ways:

- Encrypting and decrypting TCP/IP packet data in an IP tunnel.
- Encrypting signature data included as part of the ISAKMP message flows. This encryption is only performed when RSA signature mode authentication is requested and the associated certificate was defined via the RACDCERT command which specified the ICSF keyword.

This support is provided by the combination of the Integrated Cryptographic Feature (ICRF) on the processor and the Integrated Cryptographic Service Facility/MVS (ICSF/MVS) software product.

To use this support, ICFS/MVS must be started and running. It is preferable to do this prior to staring TCP/IP, however, it can also be done when TCP/IP is active.

Note: The remaining configuration is only applicable to the use of hardware crypto when encrypting and decrypting TCP/IP packet data in an IP tunnel.

If you plan to use this hardware crypto support and issue TCP/IP commands such as OPING from a user ID on the system where the firewall is running, this user ID must be permitted to access the ICSF/MVS cryptographic services (CSFSERV). This is because these are RACF controlled. Perform the following steps to set up profiles in the CSFSERV resource class and permit users to access these profiles:

1. Define the appropriate profiles in the CSFSERV class:

   Command:

   User: **RDEFINE CSFSERV service-name UACC(NONE)**
   other-optional-operands

   The service names that the OS/390 Firewall Technologies uses are CSFCKI, CSFDEC1, CSFENC1, CSFRNG, CSFCKM, and CSFOWH1. Note that if triple DES hardware crypto support is not available on your S/390 processor, the CSFCKM service is not used.

2. Permit user access to these profiles:

   Command:

   User: **PERMIT service-name CLASS(CSFSERV) ID(yourid) ACCESS(READ)**

   where *yourid* is the stack name.

3. Activate the CSFSERV class and refresh the in-storage RACF profiles. This is done by the RACF administrator.

   Commands:

   User: **SETROPTS CLASSACT(CSFSERV)**
   User: **SETROPTS RACLIST(CSFSERV)REFRESH**

4. The MAXLEN installation option for hardware crypto determines the maximum length that can be used to encrypt and decrypt data using ICSF/MVS. Set the MAXLEN ICSF/MVS installation option to greater than 65527 as this is the maximum TCP/IP packet size. See the section on displaying installation options in the *OS/390 SecureWay Integrated Cryptographic Service Facility Administrator's Guide* for more information.

For more information, see the section on controlling who can use cryptographic keys and services in the *OS/390 SecureWay Integrated Cryptographic Service Facility Administrator's Guide*, SC23-3975.

Step 4. Configure TCP/IP on the Firewall Host

To configure and start TCP/IP, you need to make the changes described below to the configuration of each Firewall TCP/IP stack. For more information about any of the files or settings mentioned, refer to the OS/390 TCP/IP UNIX services documentation.

1. In the TCP/IP profile (*hlq*.**PROFILE.TCPIP**):
   a. Add DEVICE and LINK statements for your adapters. For example:

   ```
   DEVICE LCS1   LCS        BA0
   LINK TR1    IBMTR      0 LCS1
   LINK ETH1   ETHERNET   1 LCS1
   ```

   In this example, LCS1 is a 3172 model 1 with a Token Ring and Ethernet adapter.

   b. Comment out any standard AUTOLOG statement for the standard TCP/IP servers (FTPSERVE, TELNET, etc.),

   c. Add AUTOLOG statements for FWKERN as follows:

   ```
   AUTOLOG
   FWKERN              ; OS/390 Firewall Kernel
   ENDAUTOLOG
   ```

   Note: When implementing multiple firewall stacks, the FWKERN autolog appears only in one profile.

   d. Comment out any PORT statements for the standard TCP/IP servers,

   e. Add port reserves as follows:

   ```
   PORT
     20 TCP OMVS  NOAUTOLOG    ; Firewall FTP server
     21 TCP OMVS               ; Firewall FTP server
     53 TCP OMVS               ; OS/390 Firewall Domain Name Daemon
     53 UDP OMVS               ; OS/390 Firewall Domain Name Daemon
    500 UDP OMVS               ; OS/390 Firewall ISAKMP Server
   1080 TCP OMVS               ; Firewall SOCKS server
   1014 TCP OMVS               ; Config server
   ```

   f. Add the following keywords to the TCP/IP profile to identify it as a firewall:

   ```
   IPCONFIG FIREWALL DATAGRAMFWD
   ```

   DATAGRAMFWD enables the transfer of data between networks. The DATAGRAMFWD parameter is confirmed by the message:

   ```
   IP forwarding NOFWD multipath support is enabled
   ```

   The profile changes will take effect when you IPL the system later during configuration. The **IPCONFIG FIREWALL DATAGRAMFWD** profile line will only be acted upon by TCP/IP at start-up of the stack and will be ignored during any dynamic profile processing.

   Note: If you use any other method to start TCP/IP servers, such as via an UNIX exit, you must ensure that any such methods are disabled.

2. Create the **/etc/services** file if it does not exist, then:
   a. Add a definition for the ISAKMP server as follows:

   ```
   isakmp   500/udp
   ```

   b. Recycle (stop, then start) TCP/IP.

Step 5. Copying Shell Scripts

OS/390 Firewall Technologies contain the following executable shell scripts.

fwlogmgmt
getmsg

Running shell scripts from locales that are not generated from code page IBM-1047 requires multiple copies of each shell script, one for each different locale's code page. You can use the **iconv** command to convert a shell script from one code page to another. For example to convert the **fwlogmgmt** script to the Da_DK.IBM-277 locale, enter the following command:

```
iconv -f IBM-1047 -t Da_DK.IBM-277 /usr/lpp/fw/bin/fwlogmgmt > /tmp/fwlogmgmt
```

For more information about the **iconv** command, see the *OS/390 UNIX System Services Command Reference*, SC28-1892.

For more information about customizing your locale, see the section about Customizing the Shell in the *OS/390 UNIX System Services User's Guide*, SC28-1891, and the section about Customizing for Your National Code Page in *OS/390 UNIX System Services Planning*, SC28-1890.

Step 6. Activate Sample Configuration Files

Note: It is important to preserve the owner, group, and mode settings when copying sample files. This can be done using the **-p** option of the **cp** command from a superuser (UID=0). For example:

```
cp -p /usr/lpp/fw/etc/security/fwrules.cfg /etc/security/fwrules.cfg
```

The following sample configuration files are shipped with OS/390 Firewall Technologies in **/usr/lpp/fw/etc**:

fwftp.data
: FTP proxy configuration file which enables client response messages in American English and Japanese.

fwftp.deniedusers
: FTP proxy configuration file which lists users that are denied access to the FTP proxy.

To use these samples, copy them into the **/etc** directory.

In addition, the following files are shipped with OS/390 Firewall Technologies in **/usr/lpp/fw/etc/security**. They contain firewall default definitions:

- **fwaudio.cfg**
- **fwdaemon.cfg**
- **fwobjects.cfg**
- **fwrules.cfg**
- **fwservices.cfg**
- **fwsocks.cfg**
- **fwahtran.cfg**
- **fwesptran.cfg**
- **fwkeypol.cfg**
- **fwkeyprop.cfg**
- **fwkeyring.cfg**
- **fwkeytran.cfg**

If you are not migrating from a previous release of OS/390 Firewall Technologies, you must copy these files into the **/etc/security** directory during installation, if they are not already there.

If you are migrating from a previous release, run the command “fwmigrate” on page 275 to preserve your current configuration, and copy the following files into the **/etc/security** directory, if they are not already there:

- fwahtran.cfg
- fwesptran.cfg
- fwkeypol.cfg
- fwkeyprop.cfg
- fwkeyring.cfg
- fwkeytran.cfg

Whether or not you are migrating from a previous release of OS/390 Firewall Technologies, you must copy the following files into the **/etc/security** directory during installation:

- **fwguicmds.En_US**
- **fwguicmds.Ja_JP** (if Japanese version is installed)

Step 7. Define Firewall Stacks through FWSTACK Command

Use the command “fwstack” on page 311 to define the firewall stacks for each firewall stack that was configured in “Step 4. Configure TCP/IP on the Firewall Host” on page 28. For more information on defining TCP/IP stacks, refer to the publication *OS/390 IBM Communications Server: IP Configuration Reference* SC31-8726.

Step 8. Define the Secure Interface to the Firewall

Use the command “fwadapter” on page 179 to list adapters attached to the machine, and to define one or more as your secure interface(s). Use the addresses you filled in your network planning worksheet, “Network Configuration Planning Worksheet” on page 13.

Step 9. Configure Firewall Servers

All the OS/390 Firewall Technologies servers (also referred to as “daemons”) run in their own address spaces. Servers are controlled by the Control Task running in the ***firewall kernel*** referred to as the FWKERN, address space.

How Servers are Controlled

The FWKERN address space must be started before any of the servers can be started. All requests to start, stop, or query the firewall servers (either collectively or individually) are made through the FWKERN control task through the START, STOP, or MODIFY commands, which you issue from the OS/390 operator console. See “fwkern” on page 247 for details on this command.

Use the command, “fwdaemon” on page 201 to list and change server configuration attributes, query server status and start and stop servers.

Step 10. Enable Firewall Services and Features

When you first install the OS/390 Firewall Technologies and start the firewall kernel, all firewall services are available for use, but only the **fwstackd** server is started.

The default filter action is to deny all traffic through the firewall and to only allow local traffic using the secure adapter.

Additional configuration is required to allow users access to the FTP, CFGSRV, and ISAKMP servers on the firewall. Also, any filter, tunnel, or NAT definitions that you want to be active need to be configured using the appropriate commands.

Step 11. Activate System Configuration Changes

System configuration changes that were made in previous steps must be activated on your system. This can be accomplished in one of the following two ways:

1. IPL your system
2. Activate the changes dynamically:
   - APF-authorize the firewall data sets
   - Mount the firewall HFS data set at **/usr/lpp/fw**
   - Activate the BPXPRM*xx* changes via the **SETOMVS** or **SET OMVS** commands
   - Stop and restart TCP/IP

The firewall stack modifications will be found by the stack start-up code in the SEZALINK data set, where the rest of the TCP/IP load modules reside. If the stack modifications load successfully, you will see the message:

```
EZZ0349I FIREWALL SUPPORT IS ENABLED
```

If firewall support cannot be initialized correctly, one or more of the following messages display on the OS/390 operator console:

```
EZZ0349I FIREWALL SUPPORT IS DISABLED

EZZ0392I CANNOT START FIREWALL AFTER TCPIP IS ACTIVE

EZZ0393I FIREWALL MODULE EZAFWALL LOAD FAILED

EZZ0394I FIREWALL SETUP FAILED RETURN <rc> REASON <reason>
```

See the *TCP/IP for MVS: Messages and Codes* manual, SC31-8307.

Step 12. Start the Firewall Kernel

Notes

1. This step is not required if there is an AUTOLOG statement for FWKERN in *hlq*.**PROFILE.TCPIP**:
2. To activate firewall functions, the firewall kernel must be started after a firewall TCP/IP stack is started, or it can be added to the AUTOLOG list in the TCP/IP profile. A firewall TCP/IP stack is defined by the **fwstack** command.

Start the firewall from the OS/390 operator console using the command “fwkern” on page 247:

```
start fwkern
```

This command will start the firewall kernel and the selected servers.

At this point, the firewall kernel program should start, and you will see the following message on the OS/390 operator console:

```
ICAM1003i FWKERN initialization complete.
```

This indicates that the firewall is ready for configuration.

If the firewall kernel cannot start, the following message displays on the OS/390 operator console:

```
IEAF404I  FWKERN - ENDED - TIME=hh:mm:ss
```

Step 13. Define Firewall Logging

The firewall logging activity is an integral part of your secure network, and you should log as much activity as you can. See Chapter 6, “Managing Firewall Logging Activity” on page 45 for more details.

Chapter 5. Setting Up the Configuration Server and the Configuration Client (GUI)

This section describes how to set up the configuration server and the configuration client, which is the graphical user interface (GUI) for the OS/390 Firewall Technologies.

Configuring SSL

The firewall configuration server uses the Secure Sockets Layer (SSL) protocol of OS/390 for communication between the graphical user interface (GUI) clients and the server.

The administrator must run the **gskkyman** SSL command to create a new key database, if one was not already created. The administrator needs use the 'Create a self-signed certificate' option to create and store a Version 3 self-signed certificate and then the 'Store encrypted database password' option. See the *OS/390 System Secure Sockets Layer Programming Guide and Reference*, SC24-5877 for information on how to use the **gskkyman** command. Record the name of the encrypted password file (.sth) and key database file (.kdb) that are created by **gskkyman** for future reference.

Configuring the Configuration Server (CFGSRV)

The configuration server must be configured to use the SSL option. Because the clients can only connect to the server using SSL, failure to configure the server with the SSL option will result in a connection failure between the client and the server.

1. The SSL library **hlq.SGSKLOAD** must be APF authorized, and available to Firewall through either a STEPLIB DD statement or through LNKLSTxx. To use STEPLIB, update the ICAPCFGS PROCLIB member with a //STEPLIB DD statement that references **hlq.SGSKLOAD**.
2. To configure the server with the SSL option, add the **-f** option with the fully-qualified name of the key database file (**.kdb**) to the **daemonopts** parameter for CFGSRV.

   Optionally, the port can also be specified on the **daemonopts** by using the -p option. Port 1014 is the default.

   ```
   fwdaemon cmd=change daemon=CFGSRV daemonopts="-f /somedirectory/key.kdb -p 1014"
   ```

3. Configure the server for starting by FWKERN:

   ```
   fwdaemon cmd=change started=yes daemon=CFGSRV
   ```

   FWKERN will now start the configuration server at startup. If FWKERN is already running, start the server using **fwdaemon** from OMVS or use the Modify command from the OS/390 operator's console:

   ```
   fwdaemon cmd=start daemon=CFGSRV
   ```

   or

   ```
   F FWKERN,START CFGSRV
   ```

Note: You must have filter rules which allow TCP traffic to flow to the configuration server.

Note: To start the configuration server in the Japanese code page (Ja_JP.IBM-939), do the following:

1. A Japanese-based SSL key database and self-signed certificate must be created by the **gskkyman** command. This is accomplished by setting the LANG environment variable to Ja_JP.IBM-939 prior to invoking the **gskkyman** command.

© Copyright IBM Corp. 1997, 2000

2. The LANG environment variable associated with the configuration server must be set to Ja_JP.IBM-939. This is accomplished by issuing the following command prior to starting the configuration server:

```
fwdaemon cmd=change daemon=cfgsrv
         runopts='ENVAR("LANG=Ja_JP.IBM-939")'
```

Setting Up the Configuration Client on AIX

To set up the configuration GUI on an AIX machine, you will need:

- Java.rte 1.1.6 or 1.1.8 (Java Run-Time Environment)
- AIX 4.2 or higher (as long as Java.rte 1.1.6 or 1.1.8 is available and supported with that level of AIX)
- Netscape with Frames support

Note: Java.rte is available on the AIX bonus pack. The level of Java.rte is a key part of the requirements.

Moving the Client Code for AIX

At installation of the OS/390 Firewall Technologies, the AIX configuration GUI code is located in the **/usr/lpp/fw/bin/fwtech.obj**. Transmit this file to the AIX workstation using FTP or a similar facility. If FTP is used to transmit the file, the BINary option must be set before the GET or PUT is issued. Also, if the firewall is already active on OS/390, you must have filter rules to allow the FTP connection between the firewall and the client.

1. When the `fwtech.obj` file resides on the client, run SMIT and select the `latest_install` option as follows:

   ```
   smit install_latest
   ```

   When installed, the configuration GUI will reside in the **/usr/bin** directory as **fwcfgclient**.

   If not already set, the $LANG environment variable should be set to the proper locale. The locale will be one of the following:

   - en_US - US English
   - ja_JP - Japanese EUC
   - Ja_JP - Japanese PC

2. The connection timeout value can also be changed by updating the Timeout field in **/usr/bin/fwcfgclient** file. The default is 45 seconds.
3. The startup script for the AIX configuration client is **/usr/bin/fwcfgclient**. To start the configuration GUI, enter **fwcfgclient**.

Setting Up the Configuration Client on Windows

To set up the configuration GUI on a Windows machine, you will need:

- Windows NT 4.0 or Windows 95
- Browser with Java and frames support
- Zip tool that handles long file names such as the WinZip32 tool in WinZip
- Information about WinZip can be found at http://www.winzip.com

At installation of the OS/390 Firewall Technologies, the Windows configuration GUI code is located in the **/usr/lpp/fw/bin/fwtech.zip**. Transmit this file to the Windows machine using FTP or a similar facility. If FTP is used to transmit the file, the BINary option must be set before the GET or PUT is

issued. Also, if the firewall is already active on OS/390, you must have filter rules to allow the FTP connection between the firewall and the client.

- When the `fwtech.zip` file resides on the client, unzip it using the WINZIP tool. The installation directory is the administrator's choice.

Note: You can use the WINZIP tool to install the file as well.

1. The locales supported are:
   - en_US - US English
   - Ja_JP - Japanese PC
2. Change the current directory (cd) to the installation directory.

   Note: The installation directory is the location of the unzipped files.
3. Enter or select:

   `setup (setup.exe)`
4. Proceed according to the instructions in the windows.

Log On to the Configuration Client (GUI)

To log on to the configuration GUI, you must log in from an AIX, Windows 95 or Windows NT machine where the client is loaded and configured. To connect from the selected machine to the client, you must have filter rules active on the firewall to allow the connection from the clients to the configuration server port, normally 1014.

- The user must be a firewall administrator.
- The firewall administrator must have an authentication scheme defined as superuser or **fwgrp** user.
- The user must have the authority to perform specific configuration functions. The user ID used to log on to OS/390 must be given RACF (or other ESM) authorization to ICA.CFGSRV facility, as noted in “Step 2. Authorize the Firewall to the External Security Manager” on page 23. Using a superuser ID or an ID that is a member of the FWGRP group is not sufficient.

Use the following procedure to logon to the configuration GUI:

1. Enter the Host Name or IP address of the OS/390 configuration server.
2. Enter an authorized user ID for the User Name.
3. Enter the Port Number that the configuration server is listening on. The default is 1014.
4. Click OK.
5. Once the client connects to the server, you are prompted for a password. Enter the authorized user ID's logon password and click Submit or press the Enter key.

   Note: If the logon fails, correct the error given in the message displayed. Click Close to return to the main logon panel, then try again.
6. When the authorization completes, you will see the main configuration panel.

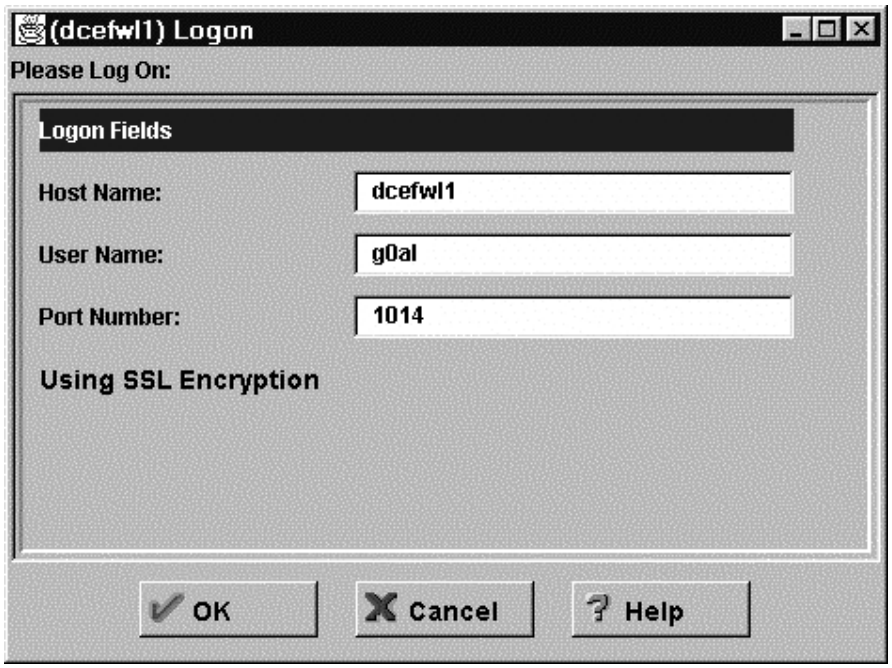


Figure 10. Configuration Client Logon Panel

The Navigation Tree

The configuration client has a collapsible tree-style navigation aid along the left side, as shown in Figure 11 on page 39.

If a node or function has items under it, a file folder icon appears at the left of the node. To see the sub-functions, you can expand the view by double-clicking on the icon. Double-clicking on the icon again collapses the view of this node back to the original view.

Any function that you click is considered selected and is highlighted. You can expand and collapse the nodes without any change to the window view on the right. When the expanded tree exceeds the vertical space available, a scroll bar appears at the right of the navigation tree. A horizontal scroll bar appears if any of the function names do not fit into the navigation tree.

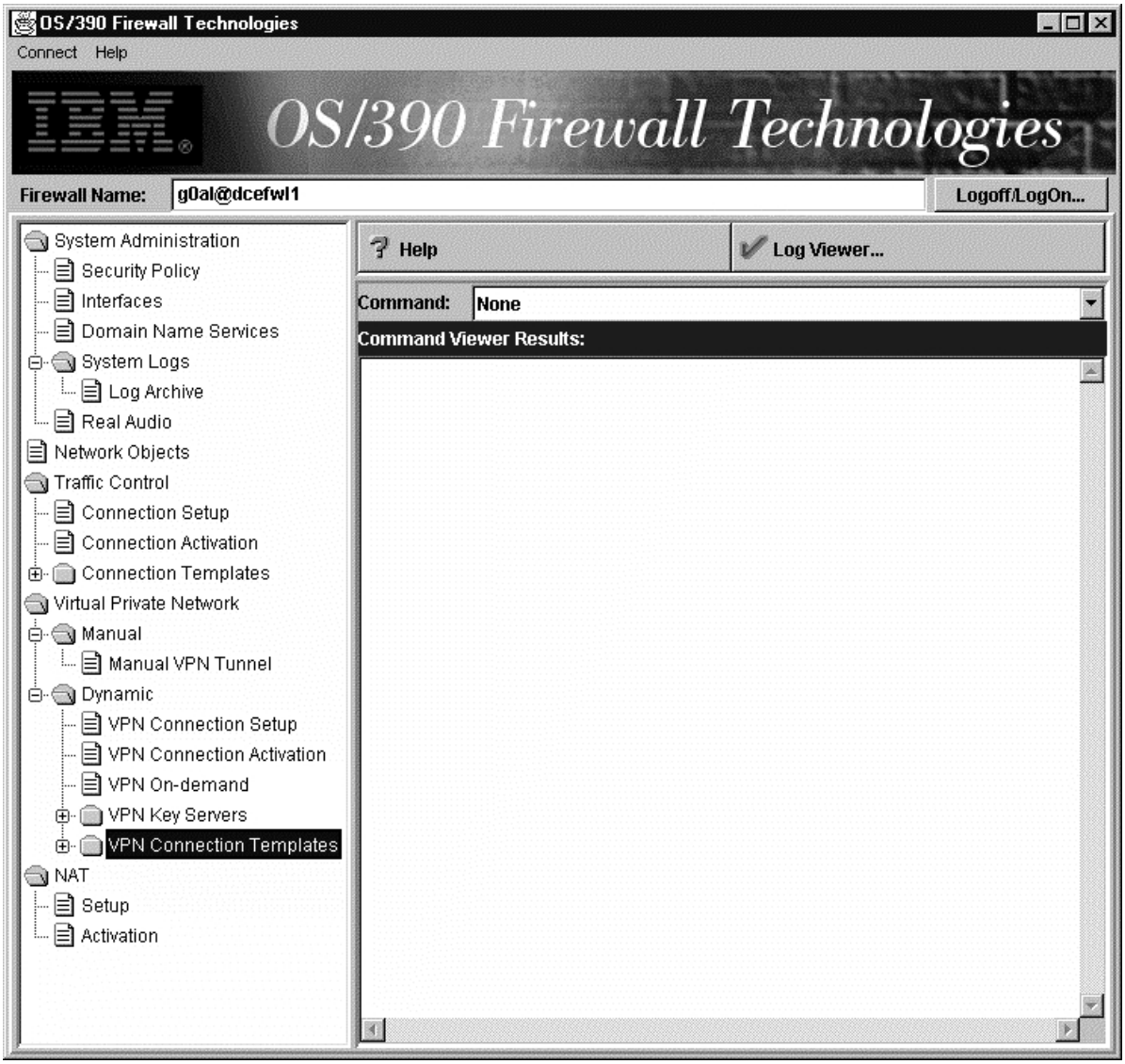


Figure 11. Configuration Client Navigation Tree

General Features on the Main Panel

On the main panel, you see the command area and the following selections, as shown in Figure 11:

Logoff/LogOn

A Logoff/LogOn button is located above the Command viewer on the configuration client main panel. It is a reconnect button. You can restart the logon sequence to connect to a different Firewall or to log on as a different administrator.

To log off, click Logoff, click Cancel on the logon panel, and the application.

Help A Help button is located above the Command viewer on the configuration client main panel. Click Help to see what to do to get your firewall up and running.

Log Viewer

A Log Viewer button is located next to the Help, above the Command viewer. The Log Viewer allows you to view or browse the firewall logs.

Command

Select None, Query Daemon Status, Query Daemon Configuration, Query Firewall Stack(s) Status or Query Firewall Trace and the status appears in the Command Viewer Results area.

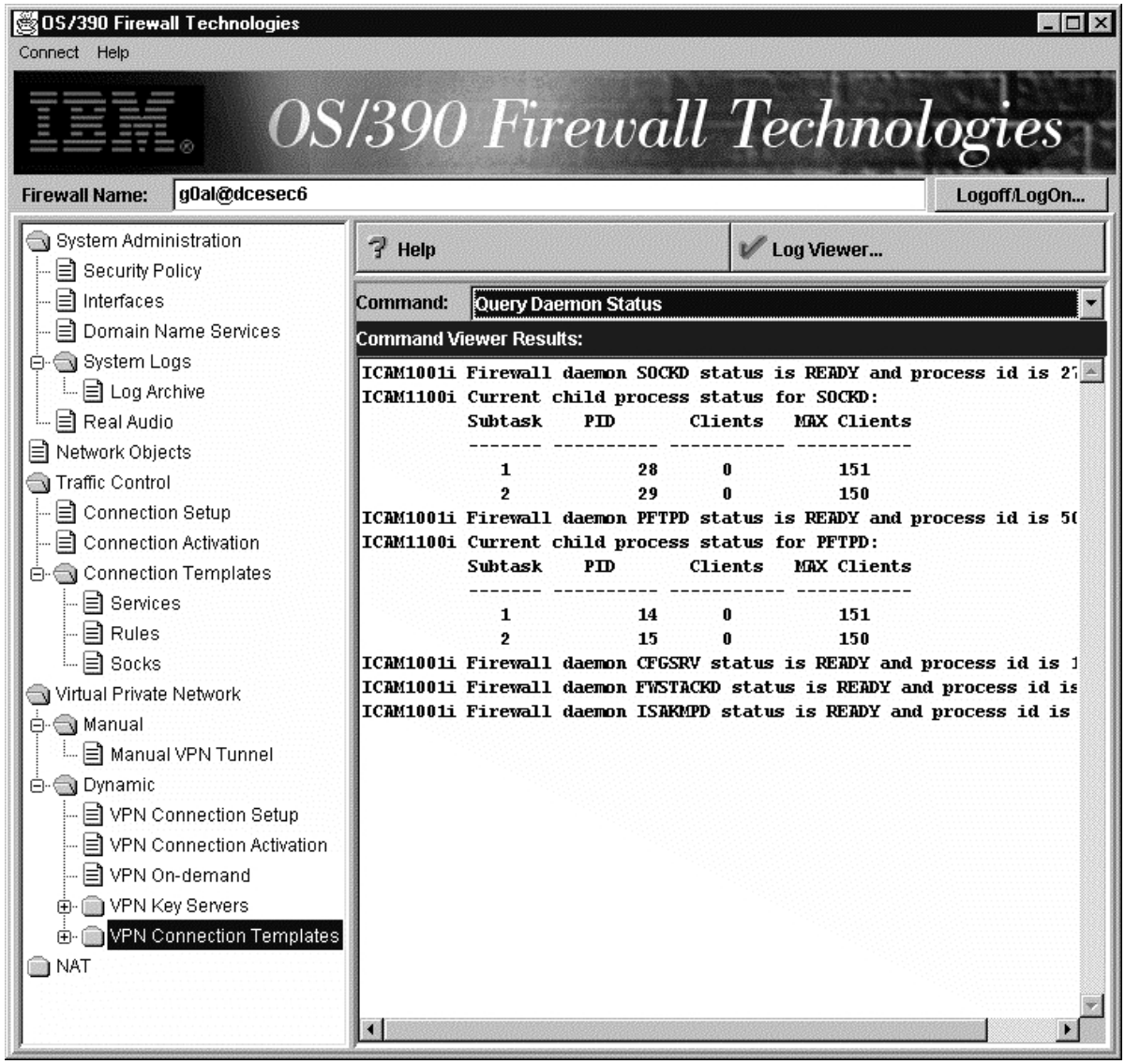


Figure 12. Command Viewer Results

The Log Viewer

Clicking Log Viewer brings up a log viewer window, as shown in Figure 13 on page 41. The log viewer allows you to view firewall log records. You can specify a log file and a record count (default is 25). You can select the target log file from the field's pull-down menu or you can type in the name of a file you want to view.

To request a specific start line, click Start at Line, after entering the line number in the field next to it. To request the last number of lines, click Bottom, which takes you to the bottom of the file. Next advances you to the next set of lines in the file. Previous takes you back to the previous set of lines in the file. Top takes you to the top of the file. By checking Yes, you can optionally expand firewall logs to readable text.

See Chapter 6, “Managing Firewall Logging Activity” on page 45 for more information about log files, facilities, and monitoring.

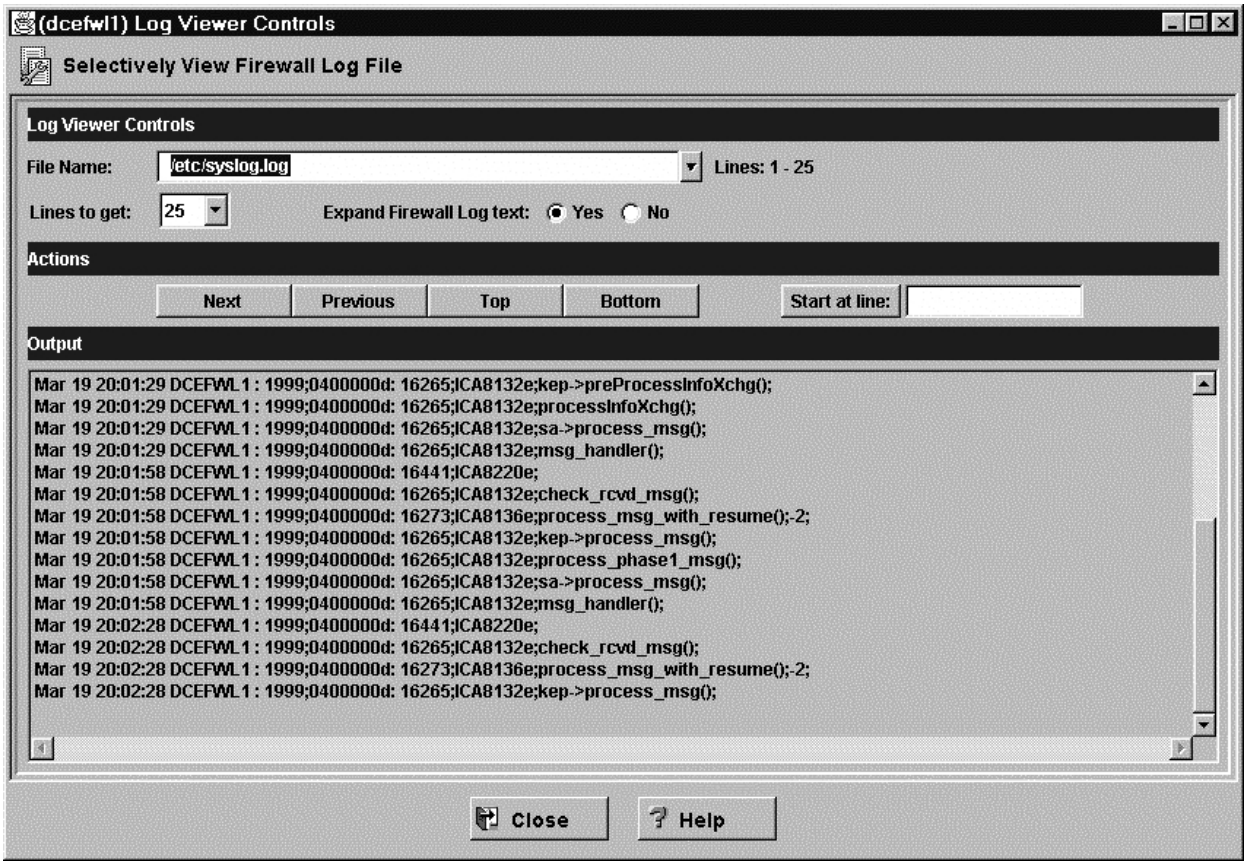


Figure 13. Log Viewer

| **Note:** The Expand Firewall Log Text option should only be used with local4 data. Other data can have
| unprintable characters.

Other Features of the Configuration Client Panels

A Search field is located near the top left corner of some of the panels. You can enter a search string and click Find. Find performs an alphabetic search on the first column of the field name.

Other buttons that you will see on many of the configuration client dialog boxes are:

Apply Click Apply to populate the field on the previous panel with your current selection or to save changes you have made on a panel. The Apply button will not remove the window from your display.

Bottom Click Bottom to view the bottom of a panel.

Cancel Click Cancel to close the window without saving any changes.

Close Click Close to remove the window from your display.

Copy The Copy button saves time when adding new items to the list. After selecting an item on the list, click Copy to create another item that is identical to the selected item. Clicking Copy opens a new item that copies field values from the selected item on the list. These fields can be modified as needed for the new item.

Delete Click Delete to delete a selected item from the list.

Move Down
Select an item in the list and click Move Down to move the item's relative position lower in the list. Each click moves the item down one position.

Move Up Select an item in the list and click Move Up to move the item's relative position higher in a list. Each click moves the item up one position.

OK Click OK to save changes and close the window.

Open After selecting an item on the list, click Open to view or modify that item. To add a new item, click NEW item on the list and click Open.

Refresh Click Refresh to re-access the data from the firewall and display the data on the panel again.

Remove Click Remove to delete a selected item from a list. This will only remove the item from the list. It will not effect other places where the item is defined.

Select Click Select to access a list of candidate items that are valid for this function.

Top Click Top to go to the top of a panel.

Help Click Help or F1 for Help on using this panel.

Common Fields

Common fields seen on many of the configuration client dialog boxes are:

Output Progress information displays in this area as your command processes.

Name Specify a name for this item. The name must be unique for this function in the firewall, and CANNOT contain a vertical bar (|).

Description
This field is optional and can be used if you want to provide a comment or additional information about this item.

Part 2. Firewall Services

© Copyright IBM Corp. 1997, 2000

Chapter 6. Managing Firewall Logging Activity

Your firewall is the focal point between your secure environment and the outside world. Beyond configuring the firewall to eliminate security exposures, the most important step you can take to secure your environment is to make sure that you log as much activity as you can. It is estimated that up to eighty-five percent of network intrusions go undetected, and this lack of detection may well be related to the lack of thorough logging. Log records can be the only tool available to discover how an attack was mounted and what damage was done.

Although it is true that record keeping can consume significant amounts of disk space, this cost does not usually compare with the importance of the business that you are protecting and the cost of lost data or interrupted services.

| As users try to access hosts through the various firewall services, the Communication Server syslog captures this activity and gives you the option to display, send and/or record it, depending on the origin and type of event. For example, you can choose to send all error messages to an administrator, while logging all normal firewall activity in a file.

| The Communication Server logging server, (**syslogd**) can log firewall events (in the form of system messages) and send the results to log files in HFS, other machines, users, or the *OS/390 MVS System Management Facilities (SMF)*, GC28-1783.

You can specify that **syslogd** log events based on three factors:

- **facility** (or origin) of the event
- **priority** (or severity) of the event
- **action** to be taken with the event

Logging Performed by the Firewall

| OS/390 Firewall Technologies only uses a subset of these for its own logging activity:

Figure 14. Logging Performed by the Firewall

Facility	General Use	Priorities and Events Used For	
	local0	used by the OS/390 Firewall Technologies configuration and administration commands to record command information.	**info**	command results
	daemon	used by the UNIX services name server.	**info**	command results
	local4	is used by socks, FTP, filters, tunnels and NAT to record a wide range of events and information such as currently used socks rules, successful connections, connection terminations, proxy logins, etc.	**info**	informational messages
			notice	special conditions
			err	error conditions

| For information about defining syslog entries, and syslog daemon configuration statements, see *OS/390 IBM Communications Server: IP Configuration Reference*, SC31-8726.

The Firewall NAT (Network Address Translation) facility maintains another log of the last 1024 network mapping log entries. This log is kept in storage and can be used to review how reserved addresses are being used. For more information, see Chapter 15, “Network Address Translation” on page 103.

Figure 15 on page 46 shows sample log records in condensed format.

Note: Log records are written in a condensed format to conserve space on logging devices and to allow language-specific viewing of log data. The **fwlogtxt** command can be used to create full-text messages.

© Copyright IBM Corp. 1997, 2000

```
Dec 10 15:20:55 dcefwla : 1998;06000003: 2147;ICA1073i;CS390IP;R:p;i:;10.130.110.1;
s:;1.2.3.4;d:;10.130.110.1;p:;tcp;sp:;1083;dp:;1080;r:;l;a:;s;f:;n;T:;0;l:;44;
Dec 10 15:20:55 dcefwla : 1998;0200000d: 6023;ICA3011i;user1:user1;1.2.3.4;5.6.7.8;21;

Dec 10 15:21:00 dcefwla : 1998;0200000d: 6031;ICA3015i;user1;user1;1.2.3.4;5.6.7.8;21;31;1.2.3.4;148;5.6.7.8;23;

Dec 10 15:21:09 dcefwla : 1998;0200000d: 6025;ICA3012w;user1;user1;1.2.3.4;9.8.7.6;23;

Dec 10 15:22:35 dcefwla : 1998;0400000a: 2147;ICA1073i;CS390IP;R:p; i:;11.130.10.11;s:;
11.130.10.8;d:;11.130.10.11;p:;esp;-:;0;-:;0;r:;l;a:;n;f:;n;T:;0;l:;304;
Dec 10 15:22:35 dcefwla : 1998;0400000a: 2147;ICA1073i;CS390IP;R:p; i:;11.130.10.11;s:;
11.130.10.8;d:;11.130.10.11;p:;icmp;t:;8;c:;0;r:;l;a:;n;f:;y;T:;3;l:;284;
Dec 10 15:22:37 dcefwla : 1998;06000003: 2147;ICA1073i;CS390IP;R:d; o:;10.130.110.1;s:;
10.130.80.10;d:;10.130.110.10;p:;icmp;t:;8;c:;0;r:;l;a:;s;f:;y;T:;0;l:;284;
```

Figure 15. Example of Log Records (Condensed Form)

```
1998 Dec 10 15:20:55 06000003 dcefwla : ICA1073i: CS390IP R:p i:10.130.110.1 s:1.2.3.4
d:10.130.110.1 p:tcp;sp:1083 dp:1080 r:l a:s f:n T:0 l:44

1998 Dec 10 15:20:55 0200000d dcefwla : ICA3011i: connected -- Connect from user1(user1)@1.2.3.4 to 5.6.7.8 (21)

1998 Dec 10 15:21:00 0200000d dcefwla : ICA3015i: terminated --
Connect from user1(user1)@1.2.3.4 to 5.6.7.8 (21).(31 bytes from 1.2.3.4, 148 bytes from 5.6.7.8)

1998 Dec 10 15:21:09 0200000d dcefwla : ICA3012w: refused -- Connect from user1(user1)@1.2.3.4 to 9.8.7.6 (23)

1998 Dec 10 15:22:35 0400000a dcefwla : ICA1073i: CS390IP R

i:11.130.10.11 s:11.130.10.8 d:11.130.10.11 p:esp -:0 -:0 r:l a:n f:n T:0 l:304
1998 Dec 10 15:22:35 0400000a dcefwla : ICA1073i: CS390IP R:p i:11.130.10.11 s:11.130.10.8
d:11.130.10.11 p:icmp t:8 c:0 r:l a:n f:y T:3 l:284
1998 Dec 10 15:22:35 06000003 dcefwla : ICA1073i: CS390IP R:d o:10.130.110.1 s:10.130.110.1
d:10.130.110.10 p:icmp t:8 c:0 r:l a:s f:y T:0 l:284
```

Figure 16. Example of Log Records (Formatted Form)

Logging to HFS: If you record log records in the HFS file system, two important factors to consider are **record retention** and **record organization**. If your logging is to be effective the appropriate records must be there and must be organized in a usable fashion. **You must, however, ensure that the file system does not fill up**.

Important Note to Administrator

Be sure to monitor the logging file system for out-of-storage conditions, and delete records when appropriate. If the file system fills up, logging will stop abruptly, without warning.

One way to control the impact of ever-increasing logs is to create a separate file system.

| The **fwarchive** command provides configuration definitions for record retention. It allows you to define the number of days a log record remains in the log file before it is archived, the archive file name and the number of days before it is purged from the archive. The **fwlogmgmt** command maintains the archive files using the definitions defined through the **fwarchive** command.

The **fwlogmgmt** command moves records from the HFS log file into an archive file, reducing the HFS log file size and expanding the archive file. The records are placed in the archive file in compressed unreadable format. For more information about the command and the archive files, see the command “fwlogmgmt” on page 272.

Using Logging

To make logging information more readable and meaningful, it is usually desirable to divide the logs into *categories*. Each logging record is in the form of a system message which has an identifier associated with it (see Chapter 19, “Messages” on page 329). The identifier indicates both the facility and priority of the message. This information, combined with the ability to send the messages to a file, host or specific user allow you to tailor the output in the way most useful to you. For example, each combination of origin and severity could be written to a separate log file to facilitate searching for record types.

Extracting Log Messages

OS/390 Firewall Technologies provides a reporting utility that you can use to extract log messages from the system log file. The **fwlogtxt** command runs against the system log file to generate a file of readable, full-text messages. (see “fwlogtxt” on page 274.)

Figure 16 on page 46 shows full-text messages that were generated by the **fwlogtxt** utility, using the messages in Figure 15 on page 46 as input. The following describes the messages in those examples.

Message ICA3011i indicates that a successful FTP connection was established through the Socks server between IP addresses 1.2.3.4 and 5.6.7.8 for user user1.

Message ICA3015w records the termination of the FTP session.

Message ICA3012w indicates that user user1 attempted a telnet connection through the Socks server from 1.2.3.4 to 9.8.7.6 and the connection was denied. Note that in the source log file, message ICA3012w appeared only once. This was followed by a message specifying that the previous message (ICA3012w) was repeated 2 times. **fwlogtxt** replaces this message with the specified number of instances of the original message.

The following field explanations refers to the ICA1073i log records. These are log record examples and not to be considered as actual packet flows because many records were edited out. The first instance of message ICA1073i was created when an the initial FTP packet came from a client through the firewall's secure adapter. The second instance was created when a permitted, encrypted packet came in through the firewall's nonsecure adapter. The third instance of ICA1073i was created when the encapsulated packet represented by the previous ICA1073i record was decrypted by the firewall. The final instance of ICA1073i was created when an initial ping packet was denied permission to be sent out through the firewall's secure adapter. The fields to the right of the ICA1073i record number use data from the first instance of ICA1073i in this figure:

R:p The rule's type. p for permit rules, d for deny rules.

i: The packet's direction. i for incoming packets, o for outgoing packets.

10.130.110.1
The address of the adapter through which the packet arrived, or will be sent (would be sent for denied packets).

s:1.2.3.4 The packet's source address.

d:10.130.110.1
The packet's destination address.

p:tcp The packet's protocol.

sp:1083 The packet's source port. (Or the ICMP type, eg. t:8.)

dp:1080 The packet's destination port. (Or the ICMP code, eg. c:0.)

r:l The packet's destination with respect to the firewall machine. r for routed, l for local.

a:s The state of the adapter through which the packet arrived. s for secure, n for nonsecure.

f:n The packet's fragmentation state. y (yes) for fragmented, n (no) for non-fragmented

T:0 The tunnel number associated with the packet and the rule which it matched. This field is zero for packets not using a tunnel.

l:44 The packet's length, in bytes.

Note: Use **fwlogtxt** only with local4 records. Other types of records may have unprintable characters that cannot be expanded.

Chapter 7. Tracing Firewall Daemon Activity

The firewall administrator can use the **fwtrace** command to set up tracing for the following firewall components:

- Administration utilities (like fwtrace itself)
- Configuration commands
- FTP daemon
- system functions (character conversions, system utility functions, etc.)
- Socks daemon
- Firewall kernel services
- Configuration Server
- ISAKMP daemon

See “fwtrace” on page 314 for more information about the **fwtrace** command.

Trace Data Files

When tracing has been activated, an output file named **fwtrace.data** is created in directory path `/var/fw/fwdata/` to hold the output. The file may also have a user-defined name and directory assigned by the firewall administrator.

The format of the data in each of these files is:

```
1997/08/26 07:25:52 (0647EF30:00000001): COMPNAME function_name(): Calling function x, input = 1
1997/08/26 07:25:53 (0647EF30:00000001): COMPNAME function_name(): Return from function x, rc = 0
A         B       (   C   :  D    ):   E
```

where:

A Date stamp for trace record

B Time stamp for trace record

C Process ID of currently running process

D Thread ID of currently running thread

E Component name (up to 8 characters) followed by trace data (function name and string data)

© Copyright IBM Corp. 1997, 2000

Chapter 8. Controlling Traffic Through the Firewall with Filters

OS/390 Firewall Technologies acts as an IP router, passing data between the secure and nonsecure interfaces. **IP filters** provide controls to decide which data packets are passed and which are blocked. Filter rules can be defined to block or pass packets based on the following:

- Source IP address
- Destination IP address
- Direction of data flow
- IP protocol
- Port number
- Time (month, day or time period)
- Type of interface (secure or nonsecure)

IP filtering provides the basic protection mechanism for the firewall. This mechanism allows you to determine what traffic passes across the firewall based on IP session end details, to protect the secure network from outsiders using unsophisticated techniques (such as scanning for secure servers) or even more sophisticated techniques (such as IP address spoofing). Think of the filtering facility as the base on which the other tools are constructed. It provides the infrastructure in which they operate and denies access to all but the determined intruder.

Filtering in OS/390 Firewall Technologies is actually composed of several interacting components. The sections that follow explain these components and how to define them.

© Copyright IBM Corp. 1997, 2000

Filtering Components

Using filters involves the components illustrated in Figure 17.

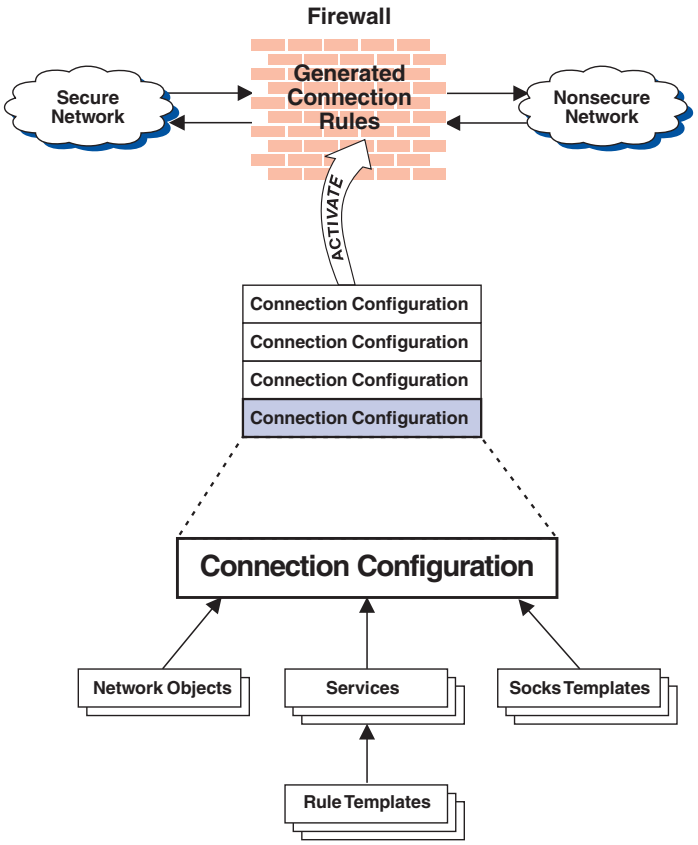


Figure 17. Building Connections

Network Objects

Represent the various hosts and entities which interact with the OS/390 Firewall. Network objects can be grouped. OS/390 Firewall Technologies comes pre-loaded with a default object. This object refers to the entire set of IP V4 addresses and is known as "The World". You must keep this defined object. Define network objects using the command “fwnwobj” on page 288, or use the configuration client “Defining Network Objects” on page 112.

Network Object Groups

Represent a collection of network objects. They are used as a convenience in setting up connections and can eliminate repetitive work. For example, you could group several hosts together into a network object group to represent a department. This network object group can then be used as either the source or destination for a connection. Define network object groups using the command “fwnwgrp” on page 284, or use the configuration client “Network Object Groups” on page 113.

Rule Templates

Provide instructions to the firewall to permit or deny IP packets based on their various attributes. OS/390 Firewall Technologies comes pre-loaded with a default set of rules. You cannot delete or modify these pre-loaded default rules. If they do not meet your needs, you can add to them by creating new rules. Define filter rules using the command “fwfrule” on page 238, or use the configuration client “Creating Filter Rules” on page 115.

Socks Templates
Provide instructions to the firewall socks server to permit or deny connections to the socks server based on their various attributes. While socks rules are not part of IP filtering, they do use other IP filtering components such as Network Objects, and are used in creating connections. Define Socks rules using the command “fwsrule” on page 305, or use the configuration client “Adding a New Socks Rule” on page 122.

Services Groups rules templates together to instruct the firewall to permit or deny some meaningful type of communication. For example, one of the FTP services is comprised of eight rules. OS/390 Firewall Technologies comes pre-loaded with a default set of services. You cannot delete or modify these pre-loaded default services. However, if these predefined services do not meet your needs, you can add services by creating services. Define services using the command “fwservice” on page 298, or the configuration client “Creating a Service” on page 120.

Connections
Associate network objects with services and socks templates to define the types of communications allowed between endpoints. Define connections using the command “fwconns” on page 195, or the configuration client “Building Connections Using Predefined Services” on page 125.

Activating the Setup:

Note: You must copy the default installation rule files from their installation location to the **/etc/security** directory. See “Step 6. Activate Sample Configuration Files” on page 30 for details.

Predefined Services

The OS/390 Firewall comes pre-loaded with a default set of services. Services are a collection of rules or a set of instructions to permit or deny a particular type of traffic through the firewall, for example, a telnet session. You can add to services by using the rule templates to create new rules.

Using the **cmd=list** parameter on the objects, services, socks rules, and filter rules shows all items. Those with IDs up to 500 are predefined.

Network Objects

Network objects are representations of objects that exist in your network such as hosts, networks, routers, or virtual private networks.

Note: Tip:

It is a good idea to encompass contiguous address ranges into a single network object whenever possible. This improves the performance of the connection rule processing. The following example illustrates this:

```
ACCOUNTING DEPARTMENT
  Kevin's machine  191.1.10.1
  Susan's machine  191.1.10.3
  Helen's machine  191.1.10.5
  Peter's machine  191.1.10.7
  Bob's machine    191.1.10.9
```

To create a network object for this accounting department, you would enter the IP address information for this group as: 191.1.10.0 with a Subnet Mask of: 255.255.255.0. This network object, accounting department, can be used as either the source or destination for a connection.

Defining Network Objects

To define a network object, use the command “fwnwobj” on page 288 or the configuration client “Defining Network Objects” on page 112.

Objects can be identified by name, type, and description. There are several types of network objects, but **host** and **firewall** are the most common. The default network object shipped with the firewall is "The World". This is a global object that encompasses all possible IP addresses. After you have filled in the network configuration worksheets, which can be found in “Network Configuration Planning Worksheet” on page 13, you are ready to build objects.

In general, the steps to specify a network object are:

1. Specify the object name.
2. Specify an optional description.
3. Specify the type of the network object.

   The types of network objects are:

   host a particular node on your network. It will usually have a mask of 255.255.255.255.

   network a collective range of network addresses characterized by an address and subnet mask.

   firewall a single machine with a firewall installed on it.

   router a host that routes traffic between two or more networks.

   interface a network adapter on a machine. It does not have to be an adapter on the firewall.

   VPN (Virtual Private Network) the network on the other side of a tunnel.
4. Enter a dotted-decimal IP address for this object, and enter a subnet mask that specifies the bits in the address to compare to the address of the IP packet.
5. Or, enter the starting and ending addresses of a range of IP addresses.

You can create single or group objects. All network objects are defined by an IP address and an address mask (subnet mask) so that it is possible for one object to represent a range of network addresses.

Network Object Groups

A group represents a collection of network objects. Groups are used as a convenience in setting up connections and can eliminate repetitive work. An example would be to group some addresses, individually represented by network objects, into a network object group to represent a department. This department can be used as either the source or destination address for a connection.

To define a group of network objects, use the command “fwnwgrp” on page 284 or the configuration client “Network Object Groups” on page 113. In general, to define a network object group:

1. Specify the group name.
2. Specify an optional description.
3. Specify the objects to be included in the group.

Filter Rules

Filter rules serve two basic functions:

- They are instructions to permit or deny certain types of IP packets to and from various endpoints on your network.
- Anchor filter rules serve as a placeholder for dynamic filter rules in the ordered list of filter rules. You can then include these rules in **services**, which, in turn, are used in **connections**.

Creating the Rule Base

Use the command "fwfrule" on page 238 or the configuration client "Creating Filter Rules" on page 115 to create and change filter rules.

This section gives an overview of how to use the **fwfrule** commands, and explains some of the values you must specify for them. Complete information about the syntax and parameters, as well as examples, can be found in the command information.

Adding a Rule:

Use the **cmd=add** keyword to add a rule.

1. Specify the rule name.
2. Specify the rule description.
3. Specify the rule type (PERMIT, DENY or ANCHOR)
4. Specify an internet protocol
5. The logical operation and port number operands are used together. The source *opcode* and destination *opcode* operands are logical operations that state a relationship between the port number (destination or origin) for the packet and the source *port_num* and destination *port_num* operands. For example, if the packet destination port is port 20, and the destination *opcode* and destination *port_num* operands are "ge 15," the packet matches because port 20 is greater than or equal to port 15.

   If you use a source *opcode* or destination *opcode* with the value **any**, the filter does not look at the port number; **any** means that any port will match. The port number cannot be changed in this case. The *opcode* and port number are specified differently for ICMP, OSPF and IPIP protocols.

   The values for source *opcode*, destination *opcode*, and *icmp opcode* are:

   - Any
   - EQ (equal to)
   - NEQ (Not equal to)
   - LT (Less than)
   - GT (Greater than)
   - LE (Less than or equal to)
   - GE (Greater than or equal to)

   Here are some of the more important ports to protect:

Port	Use
20	FTP data
21	FTP control
23	Telnet
25	Mail
53	Domain Name Server

70 Gopher

80 HTTP

111 RPC

1080 socks

In the rule values, note these constraints:

- The values for the port numbers must be in the range 0 through 65535.
- A port number assigned by TCP/IP will never be less than 1 or greater than 65535.

6. Specify the type of interface (adapter) you want:

   both For packets coming or going on either the secure or the nonsecure interface

   secure For packets coming or going on the secure interface

   nonsecure
   For packets coming or going on the nonsecure interface

7. Specify the routing:

   both Applies to all traffic.

   local Implies that the packet is local to the firewall host. This means that:

   - Incoming local packets are packets that are received by the interface and are destined for this firewall host; they will not be routed to another host. Their destination is local.
   - Outgoing packets are transmitted from the interface, but originate on the firewall host. Their origin is local.

   route Implies that the packet is routed by the firewall host. This means that:

   - Incoming routed packets are packets that are received by the interface and are destined for some other host; they will not remain on the firewall. Their destination is remote.
   - Outgoing packets are transmitted from the interface, and originated on some other host. Their origin is remote.

8. Specify the desired direction:

   both For packets going out from or into the selected interface

   inbound For packets coming into the selected interface from the network

   outbound
   For packets going out from the selected interface to the network

9. If you specify Yes for the Log Control field, every packet that matches that rule gets recorded in the **local4** log. **local4** is also used by Socks, FTP, tunnels and Firewall services. (See Chapter 6, “Managing Firewall Logging Activity” on page 45 for more information on logging.)

   For **permit|deny|anchor** rules: If you specify Yes, every packet that matches that rule gets recorded in the **local4** log. If this parameter is not specified, the default is No for permit rules, Yes for deny rules, and No for anchor rules.

10. Specify the desired fragment control. For IP packet information to match a rule fragmentation control specification, the control is interpreted as follows:

    Yes The rule will match fragment headers, fragments and non-fragments. For fragments, the port information will be ignored and assumed to match.

Only Only fragments and fragment headers can match. For fragment headers, port information must match. For fragments, port information will be ignored.

No Only non-fragments can match. Fragment headers and fragments are excluded by this parameter.

Headers Only non-fragments and fragment headers can match. Fragments are excluded by this parameter.

If this parameter is not specified, the default for both "permit" rules and "deny" rules is Yes.

Note: The **fwfrule** command displays options for fragmentation, however, fragment options are ignored and not needed. All packet fragments will be reassembled before any filtering is performed.

11. You can specify a tunnel ID on a filter rule.

For a packet header to match a defined IP rule, the packet information must match all the parameters specified in the coded rule.

Changing a Rule: To change a rule, use the **fwfrule** command with the **cmd=change** keyword or the configuration client “Changing IP Rule Configuration Entry” on page 119.

1. Specify an existing rule.
2. Modify the appropriate fields as described in “Adding a Rule” on page 55.

Deleting a Rule: To delete a rule, use the **fwfrule** command with the **cmd=delete** keyword or the configuration client “Deleting Rule Configuration Entry” on page 119.

1. Specify an existing rule.
2. Specify **force=** yes or no.

Services

A **service** is a set of filter rules that work together to perform a task. For example, if six rules work together to allow all of the requests and acknowledgements that make up an FTP transfer, these six rules could be members of a service called "Permit FTP Transfer". Usually, services permit or deny a particular type of traffic through the firewall, for example, a telnet session.

OS/390 Firewall Technologies is installed with a predefined set of default services. All defined services can be created and displayed using the command line or the configuration client (Chapter 17, “Using the Configuration Client Graphical User Interface (GUI)” on page 111).

Default services are those with a service identifier less than or equal to 500. User defined services are those with a service identifier greater than 500.

Creating a Service

To define a new service, you must define the rules that make up the service. The rules may already exist on the firewall or you may add them using the **fwfrule** command. OS/390 Firewall Technologies comes preloaded with a default set of rules, but you can add to these by first using the **fwfrule** command to create new rules, and then using the **fwservice** command to define services that use the new rules.

Configuring Socks Rules

When you first start the OS/390 Firewall Technologies without configuring the firewall services, the socks server (**sockd**) is enabled, but its initial state is to deny all access. For socks clients to use the socks server, you must define some socks rules for the server to use. Define socks rules and connections using the **fwsrule** and **fwconns** command or the configuration client.

For details on how to use these commands and tasks and examples of socks rules, see:

- "fwsrule" on page 305
- "fwconns" on page 195
- "Adding a New Socks Rule" on page 122
- "Building Connections Using Predefined Services" on page 125

Socks with Filters

In addition to using the socks server, we suggest that you use filter rules as described in Chapter 8, "Controlling Traffic Through the Firewall with Filters" on page 51. A sample filter rule to enhance the use of socks follows:

```
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 1080 nonsecure both both
```

This rule denies access:

- From all hosts
- To all hosts
- Using the TCP protocol
- From any source port
- To the socks port (port 1080)
- On the nonsecure interface
- Both local and routed
- Both inbound and outbound

In other words, users cannot use the socks server from outside the secure network. This rule can be selected by choosing `service 34` on the command "fwsecpolicy" on page 296, or see "Defining a Security Policy" on page 111.

Connections

You need to build a connection to permit or deny specific types of communications between two named network objects or network object groups that serve as endpoints.

After you have defined your network objects, filter rules, services, and socks rules, you create **connections**. Select one network object or group to be the source and another network object or group to be the destination for the traffic flow through the firewall.

Connections permit or deny specified types of communications between two network objects that serve as endpoints, thereby allowing you to control the type of network traffic that can occur between them.

To build a connection, use the command "fwconns" on page 195 or "Building Connections Using Predefined Services" on page 125 on the configuration client. In general, the steps to define a connection are:

1. Specify the name of the connection.
2. Specify a description of the connection.
3. Specify a source network object.

4. Specify a destination network object.
5. Specify a list of services and socks rules that choose the type of traffic you wish to control between the source and destination objects. This list is composed of services and socks rules you have already defined with the **fwservice** and **fwsrule** commands.

Ordering Connections

The more rules you have, the greater the impact there will be on performance.

When a packet is received at a network interface, whether going into or out of the firewall host, filter rules are applied starting at the top of the connection rule file. The packet is compared to each rule, one by one, and when the information from the packet exactly matches the information in the rule, the action in the rule (permit or deny) is enacted. If the entire file is searched without a match, the request is automatically denied. The order of the connections within the file is therefore highly significant, and can be changed.

In general, it is recommended to place more specific rules closer to the top, and less specific rules closer to the bottom. For example, you might have a **Department ABC** with an address of (1.1.10.X) and a machine that is used as a server inside of **Department ABC**, with an address of (1.1.10.7). If you want to exclude machine (1.1.10.7) because it is a server that should not be used for telnet traffic, you must place the connection `Deny telnet for Dept A server` before the `Permit telnet for Dept A` connection. If you reverse the order of the connections, the deny connection will never be encountered.

Use the **fwconns cmd=move** command to re-order connections within the connection rule file. (See "fwconns" on page 195.)

Managing the Rule Base

Although you use the **fwfrule** command to create, delete and change individual rules, functions that are performed on the entire rule base, such as logging, validating and activating all rules are accomplished using the **fwfilter** command, as follows:

fwfilter Command Keyword	Function	Actions
cmd=update	refreshes and activates the filter rule base	Builds the generated connection rules from the connection configuration, and activates that rule set.
cmd=shutdown	Deactivate the rule base	Deactivates all rules you have defined. If you deactivate rules, the firewall is now protected by the default rules.
cmd=list	lists last generated filter rules	Lists the most recent generated connection rules set. If you did a deactivate (shutdown), these rules are not being used.
cmd=listactive	lists current static and dynamic filter rules	Lists the currently activated connection rule set.
cmd=verify	verifies rules	If you verify rule generation the firewall rebuilds the rule base. If the rules base is built without error, you can activate it.
cmd=startlog	enables rule logging	Logs selected traffic to the Firewall (**local4**) log.
cmd=stoplog	disables rule logging	Stops logging.

See "fwfilter" on page 235 and "Creating Filter Rules" on page 115 for more information.

Filter Rule Activation States

Filtering can be in one of these states:

1. **Not Active**

   You have not yet activated rules, or you have deactivated them. This is the state of filter rules when you first install the firewall and IPL your system. However, default filters *are* in place to protect your network from intrusion when you first install the firewall.

   Firewall Access

   - Local access is permitted from the secure and nonsecure interfaces.
   - All other access is denied. This means that all routing is denied between the secure and nonsecure interfaces.

   These rules are also in effect after the **fwfilter cmd=shutdown** command is issued.

2. **Active, but Has Errors**

   The rules base is active, however it contains incorrect rules, the rules file is empty, or the rules file does not exist.

   Firewall Access

   - All local access permitted only from secure interface to the firewall.
   - All other access is denied. This means that all routing is denied between the secure and nonsecure interfaces.

3. **Active and Valid**

   Use the **fwfilter** command to activate the configuration you defined.

   Note that the configuration file can be valid and still contain no rules. In this case, an implied “deny all access” rule is in effect.

   Firewall Access

   - Access determined by the configuration file.

     Each packet that is received by, or is about to be sent by, any network interface is examined and its contents compared against each rule in the generated connection rules. When a match is found, the action (permit or deny access) on that rule is carried out.
   - If no rules match the packet, there is an implied “deny all” rule that denies access.

   Note: You can have a valid configuration file without rules.

   In that case, the implied “deny all” rule is used, and all access is denied.

Chapter 9. Configuring and Using the Socks Server

This section explains how the socks server works, and how to configure it on OS/390 Firewall Technologies.

Overview

The socks server (or daemon) intercepts TCP/IP requests that would cross between your network and the Internet. However, a socks server provides a remote application program interface so that the functions executed by client programs in secure domains are piped through secure servers at the firewall.

The socks server intercepts the request, checks for authorization to come into or go out of the secure network, and either denies the request or pipes it through the firewall. The actual TCP/IP application is performed at the client workstation, not in the firewall.

How the Socks Server is Used

With **sockd** installed on a firewall host, users in the secure network can safely access remote hosts on the Internet. To cross the firewall, these users must use client programs that are designed especially to work with **sockd** server. These `socksified` clients replace their normal TCP/IP counterparts. (`socksified` TCP/IP stacks are also available on some platforms, which eliminates the need for `socksified` client programs.)

`socksified` applications work like their normal counterparts, but because they don't require direct access to the Internet, they do not breach the security of the internal network.

A socks server receives a request from a client within the secure network to a remote host as follows:

1. A user starts a `socksified` client application to access a remote host outside the internal network.
2. The application sends that request to the socks server (**sockd**) in the firewall host.
3. The socks server permits or denies the request based on current configuration rules.
4. **sockd** optionally verifies the user's identity with the **identd** server running on the user's host system. **Identd** returns the user ID of the user currently communicating with **sockd**, and this must match the user ID provided on the user's original request. If it matches, **sockd** connects the user to the requested remote host. If there is no match, **sockd** will deny the request.
5. After establishing the connection, the socks server acts as a secure pipeline to the remote host. The user is unaware of its existence.

To the user, this appears to be a one-step program. The user in the network uses a `socksified` TCP application (for example FTP) to access the intended host; the socks server merely redirects the data after validating both the user and the request.

The real TCP/IP application (such as FTP) runs in the client workstation, not in the firewall. This saves processing in the firewall.

The available socks client programs can be configured to:

- Use socks only when necessary, to access a host outside the secure network.
- Use direct TCP/IP to access hosts within the secure network.

© Copyright IBM Corp. 1997, 2000

In summary, you need only the socksified version of the client program at the workstations in your network. You can remove, or save and rename, the standard versions and rename the socksified versions to replace them.

Where to Get Socks Clients: Although IBM does not provide any standard socksified client for OS/390, most Web browsers are socksified and you can get socksified stacks for other platforms. For additional information refer to:

http://www.socks.nec.com

How the Socks Server Controls Access

When the socks server (**sockd**) receives a request, it checks against the lines in the socks configuration file (/etc/sockd.conf), one line at a time, starting at the **TOP** of the file. When it finds a line with conditions that are matched by the request, it stops. The request is either granted or denied based on the action field of that line. When a match is found, the remaining lines of the file are not searched. Therefore, it makes sense to position lines specifying more restrictive conditions ahead of those specifying less restrictive conditions in the file.

socks rules are placed in the socks configuration file in the same order that they are defined in a connection. In other words, if **fwconns cmd=list** returns **sockslist** = 501,502, then socks rule 501 will be the first line in the socks configuration file, and socks rule 502 will be the second. socks rules can be reordered within a connection by using the **fwconns cmd=movesocks** command.

If no matching line is found in the entire file, the request is denied.

Socks and Secure and Nonsecure Domains

sockd has no concept of inside or outside your network. It does know the requesting host from the destination host, and that is the basis of its access control. Therefore you can use it to control and allow access from the outside world as well.

Be careful if you choose to allow any access into your network from outside. If you do not need that kind of access, it is recommended that you specifically deny such connections in the configuration file.

Starting the Socks Server

OS/390 Firewall Technologies uses a **sockd** entry in the server configuration file to configure the Socks server. These options are specified on the *daemonopts* parameter of the **fwdaemon** command. The socks server supports several options:

[–ver] [–I|–i] [–L] [–N]

-ver
This option displays the **sockd** version number and whether it supports clients that use Rrcmd(), and then quits.

A client that uses Rrcmd() commands uses TCP/IP functions such as REXEC and RSH that perform functions on remote hosts.

-I If the connection to the client's **identd** fails, or if the result does not match the user ID reported by the client program, deny access. A client host without a properly installed **identd** server will not be served. User verification is done before, and in addition to, the normal access control.

-i This is similar to `-I` but more lenient. Access is denied only if the client's **identd** reports a user ID that is different from that which the client program claims.

-L Echoes all **sockd** log messages to the OS/390 operator console where **sockd** is running.

-N By default, **sockd** log records are written using IP addresses. If this option is specified, the server performs a reverse name look-up on IP addresses and, if successful, records domain names in the log file.

Note: Using this option can degrade performance.

Example: To start the socks server with the **-i** option, change the **sockd** configuration entry using the following command:

```
fwdaemon cmd=change daemon=sockd daemonopts=-i
```

The *daemonopts* parameter on the **fwdaemon** command specifies the configuration options to the socks server.

Determining Socks Server Connections Settings

The socks server configuration allows an installation to configure the maximum number of socks connections that can exist concurrently. This value, *maxconns*, is specified on the command “fwdaemon” on page 201.

An installation can now weigh the expected traffic through the socks server with the resources consumed by the socks server to more intelligently configure the server. One address space is created for each 300 connections that are allowed through the socks server. An attempt is made to balance the work among the socks server instances (address spaces).

Examples

1. To configure the socks server to allow 900 concurrent socks connections, the following command must be issued:

   ```
   fwdaemon cmd=change daemon=sockd maxconns=900
   ```

   Once the socks server is started, three address spaces are created to handle the socks server traffic (in addition to the *parent* socks server address space). Each address space handles a *maximum* of 300 connections.

2. Enter the following command to configure the socks server to allow 400 concurrent socks connections:

   ```
   fwdaemon cmd=change daemon=sockd maxconns=400
   ```

   Once the socks server is started, two address spaces are created to handle the socks server traffic (in addition to the *parent* socks server address space). Each address space handles a *maximum* of 200 connections.

Chapter 10. Configuring and Using the FTP Proxy Server

This section contains general information about how to configure and use the FTP proxy server (**pftpd**) from workstations both inside and outside your secure network.

Overview

File Transfer Protocol (FTP) is a TCP/IP service that transfers files from one network host to another. The FTP proxy server (`pftpd`) in the firewall checks authorization for an FTP user to go out of the secure network or to come in from the outside world. If the authorization is in order, `pftpd` makes the contact with the remote host (in this case, an FTP server outside the secure network).

This is really a 2-step process:

1. The workstation user starts the FTP client to connect to the proxy server at the firewall.
2. The proxy server relays an authorized FTP request to the intended remote host.

For example, a user inside the secure network would FTP to the OS/390 Firewall Technologies proxy FTP server, **pftpd**. Once **pftpd** checks the user's credentials, and the user's permission to cross the firewall with FTP, **pftpd**, in turn, accesses the remote host specified by the user. It appears that data and commands then flow through the proxy FTP server as if it were not there.

Configuring and Starting FTP

The FTP server uses only ports 20 and 21, which cannot be changed. The server listens to all active TCP/IP stacks when started. It does not connect or reconnect to the stacks that are started or restarted later. Use the following considerations when the proxy server is implemented in a multiple TCP/IP stack environment:

- Port 20 and 21 must be reserved on all TCP/IP stacks
- If normal FTP TCP/IP services are also needed, allocating different port addresses is required
- Routing and DEFAULTNET information should be reviewed across all stacks
- Filtering rules must be set up correctly

The FTP server is automatically started with default options when you start the firewall. These options are specified on the *daemonopts* parameter of the **fwdaemon** command. You can add the following options to the **pftp** entry in the configuration file:

[–ns] [–t *nnnn***] [–l** *nnnn***]**

-ns
Permits FTP clients to connect from nonsecure networks.

> **Important Note**
>
> This option poses a severe security risk, particularly when used with anonymous FTP.

-t *nnnn*
Logs out inactive sessions after the number of seconds specified by *nnnn*. *nnnn* can be 0-9999. A value of 0 specifies no time-out. The default limit is 900 seconds (15 minutes).

© Copyright IBM Corp. 1997, 2000

-l *nnnn*

Specifies the number of seconds, specified by *nnnn*, for an FTP session to complete login authentication. *nnnn* can be 0 through 9999. A value of 0 specifies no timeout. The default time is 60 seconds.

Example: To specify a 20 minute timeout value and a 2 minute login timeout value for the FTP proxy server, issue the following command:

```
fwdaemon cmd=change daemon=pftpd daemonopts="-t 1200 -l 120"
```

Determining Proxy FTP Server Connections Setting

The proxy FTP server configuration allows an installation to configure the maximum number of proxy FTP connections that can exist concurrently. This value, *maxconns*, is specified in the command “fwdaemon” on page 201.

An installation can now weigh the expected traffic through the proxy FTP server with the resources consumed by the proxy FTP server to more intelligently configure the server. One address space is created for each 300 connections that are allowed through the proxy FTP server. An attempt is made to balance the work among the proxy FTP server instances (address spaces).

Examples

1. To configure the proxy FTP server to allow 900 concurrent proxy FTP connections, the following command must be issued:

   ```
   fwdaemon cmd=change daemon=pftpd maxconns=900
   ```

   Once the proxy FTP server is started, three address spaces are created to handle the proxy FTP server traffic (in addition to the *parent* proxy FTP server address space). Each address space handles a *maximum* of 300 connections.

2. Enter the following command to configure the proxy FTP server to allow 400 concurrent proxy FTP connections:

   ```
   fwdaemon cmd=change daemon=pftpd maxconns=400
   ```

   Once the proxy FTP server is started, two address spaces are created to handle the proxy FTP server traffic (in addition to the *parent* proxy FTP server address space). Each address space handles a *maximum* of 200 connections.

Defining Users for Proxy FTP

To access FTP, users need:

- A user ID that is authorized to the External Security Manager (ESM) by the firewall administrator
- A UNIX services home directory on the firewall
- A password

As a rule, the administrator can **prevent** a user ID from being able to access FTP by adding it to the **/etc/fwftp.deniedusers** file. The exception to this is the `ftp` user ID itself, which is used for anonymous FTP. If this user ID exists on the OS/390 system, it will be able to access FTP regardless of whether it has been added to the **/etc/fwftp.deniedusers** file or not.

Changing User Passwords

For security reasons, new FTP users might either be enrolled with a password that is initially expired, and/or be required to change their password on a regular basis. The OS/390 Firewall Technologies FTP server allows them to do this from an FTP client in two ways:

1. On the initial FTP login, specify a password string of the following format:

   ```
   old_password/new_password/new_password
   ```

2. If already connected to FTP, issue the **USER** *userID* FTP command. At the prompt, enter a password-change string as above.

Password Validation Errors: If you receive an error when trying to change your FTP password, it is most likely because you specified:

- an incorrect old password,
- only one instance of the new password,
- two instances of the new password that do not match, or
- an incorrect password delimiter character in the password-change string.

If this occurs, retry the login by reissuing the **USER** *userID* command.

Anonymous FTP: Enabling the special user FTP ID “anonymous” requires that the user ID “ftp” must be defined on the OS/390 system.

Important Note

No password authentication is performed for the “anonymous” user ID. Therefore, for security reasons, its use is not recommended.

FTP Process

- Use FTP to access the firewall host. (We will use `ftp_gw.domain.net.com` as our host name.)

  User:

  ```
  ftp ftp_gw.domain.net.com
  ```

  The proxy server will ask for your user name:

  System:

  ```
  Connected to ftp_gw.domain.net.com.
  220 FTP_GW FTP GATEWAY (Version 1.2 12/06/1994 22:49:31) ready.
  Name(ftp_gw:fw_user):
  ```

- Enter your user name as authorized to use the firewall:

  User:

  ```
  Name (ftp_gw:fw_user):me_user
  ```

The server authenticates your identity by asking for your password (which is not being displayed) before letting you proceed. This password is then authenticated by the External Security Manager (ESM) installed on OS/390.

System:

```
331 Password required for me_user(oldpw/newpw/newpw to change password).
```

```
Password:
```

Enter your password. This is the same password with which you were enrolled in the firewall.

User:

```
Password: **********
```

After you are authenticated, the proxy server displays an FTP command prompt.

Note:

You only have the amount of time specified in the login timeout option (**-l**) to complete the userID/password authentication. If authentication does not complete within the time limit specified, the connection to the firewall FTP server is closed.

System:

```
230 To specify destination, type 'quote site remote.host.com'
```

Use the `quote` and `site` FTP commands to connect to the foreign host:

User:

```
ftp> quote site forhost.network.outside.com
```

(Specifying the word `quote` is optional)

The remote host will now ask for a user name and password for you to connect. Again, remember that this is most likely a different user name and password from those you used to FTP to the firewall.

Notes:

1. OS/390 Firewall Technologies FTP will accept the `site qlang` and `site lang=` commands, which select and set the language in which the client and host interact. Both American English and Japanese will be enabled if you have copied the sample configuration file into **/etc/fwftp.data**.
2. The commands supported vary according to the FTP client you are using. See your client documentation for specific information.

Chapter 11. Configuring and Using the ISAKMP Server

This section contains general information about how to configure and use the ISAKMP server (ISAKMPD).

Overview

The ISAKMP server (or daemon) negotiates dynamic security associations (SAs) with other servers or hosts which support the ISAKMP/Oakley standards developed by the IETF. The ISAKMP server can negotiate an ISAKMP (Phase 1) SA which will be used to protect the negotiation of IPSec (Phase 2) SAs. These IPSec SAs can then be used to protect user data, allowing secure communication through nonsecure networks.

Configuring and Starting the ISAKMP Server

The ISAKMP server uses UDP port 500 to communicate with other servers or hosts. Filter rules must be established to allow this communication to occur. Predefined services 52 and 53 (ISAKMPD UDP Non-Secure and ISAKMPD UDP Secure) can be used to enable this communication. The server listens only to active firewall stacks.

To instruct the firewall kernel (FWKERN) to automatically start the ISAKMP daemon, issue the following command:

User: **fwdaemon cmd=change daemon=isakmpd started=yes**

In order for the ISAKMP server to operate correctly, the fwstackd daemon must also be started. This server should already be configured to be started, but if it is not, issue the following command:

User: **fwdaemon cmd=change daemon=fwstackd started=yes**

The ISAKMP server supports several options, which can be specified on the daemonopts parameter of the fwdaemon command. You can add the following options to the ISAKMPD entry in the configuration file:

`[-keyretry nnnn][-keywait nnnn][-dataretry nnnn][-datawait nnnn][-L]`

-keyretry nnnn Specifies the number of times that an unanswered key negotiation (Phase 1) message will be retransmitted before the negotiation is aborted. *nnnn* can be 0-9999. The default is 10 retransmissions.

-keywait nnnn Specifies the number of seconds between retransmissions of key negotiation (Phase 1) messages. *nnnn* can be 0-9999. The default is 30 seconds.

-dataretry nnnn Specifies the number of times that an unanswered data negotiation (Phase 2) message will be retransmitted before the negotiation is aborted. *nnnn* can be 0-9999. The default is 10 retransmissions.

-datawait nnnn Specifies the number of seconds between retransmissions of data negotiation (Phase 2) messages. *nnnn* can be 0-9999. The default is 15 seconds.

-L Echoes all ISAKMPD log messages to the job output file, normally named IKEDOUT.

© Copyright IBM Corp. 1997, 2000

Example

To specify that key negotiation messages should be retransmitted 5 times, at 20 second intervals, issue the following command:

User: **fwdaemon cmd=change daemon=isakmpd daemonopts="-keyretry 5 -keywait 20"**

Chapter 12. Configuring and Using the Domain Name Service (DNS)

This section explains Domain Name Service (DNS) on the OS/390 Firewall Technologies, gives an overview of the recommended configuration, and tells you how to implement it. You do not need to configure a DNS for the firewall to be functional, but it is recommended that you do so in order to enable the recommended configuration.

Access to the domain name records for the secure network is of great interest to intruders, because it gives them a list of hosts to attack. A subverted domain name service server can also provide an access route for an intruder. From the external network, the name server on the firewall only knows itself and never gives out information on the internal IP network. From the internal network, this name server knows the Internet network and is very useful for accessing any machine on the Internet by its name.

Configuration is performed via configuration commands. You have to specify the following:

- Secure domain name
- Secure domain name server(s)
- Nonsecure domain name server(s)
- UNIX services resolver file (optional)

This generates all the necessary configuration files for the domain name server. It is still necessary to configure any name servers in the secure network to use the firewall service.

The operation of DNS on the firewall relies on three features:

1. The forwarders function, so that name servers inside the secure network can receive information about hosts outside its domain from the firewall name server, but the reverse cannot happen.
2. The caching capability which allows the firewall name server to get name information from the nonsecure network without pre-definition.
3. The fact that name resolution requests can be directed to any name server, whether or not the host from which the request is coming is a name server itself. This allows the firewall to be able to resolve names inside the secure network without giving those names away to hosts in the nonsecure network.

Figure 18 on page 72 shows how name resolution requests flow for different combinations of requester and node.

© Copyright IBM Corp. 1997, 2000

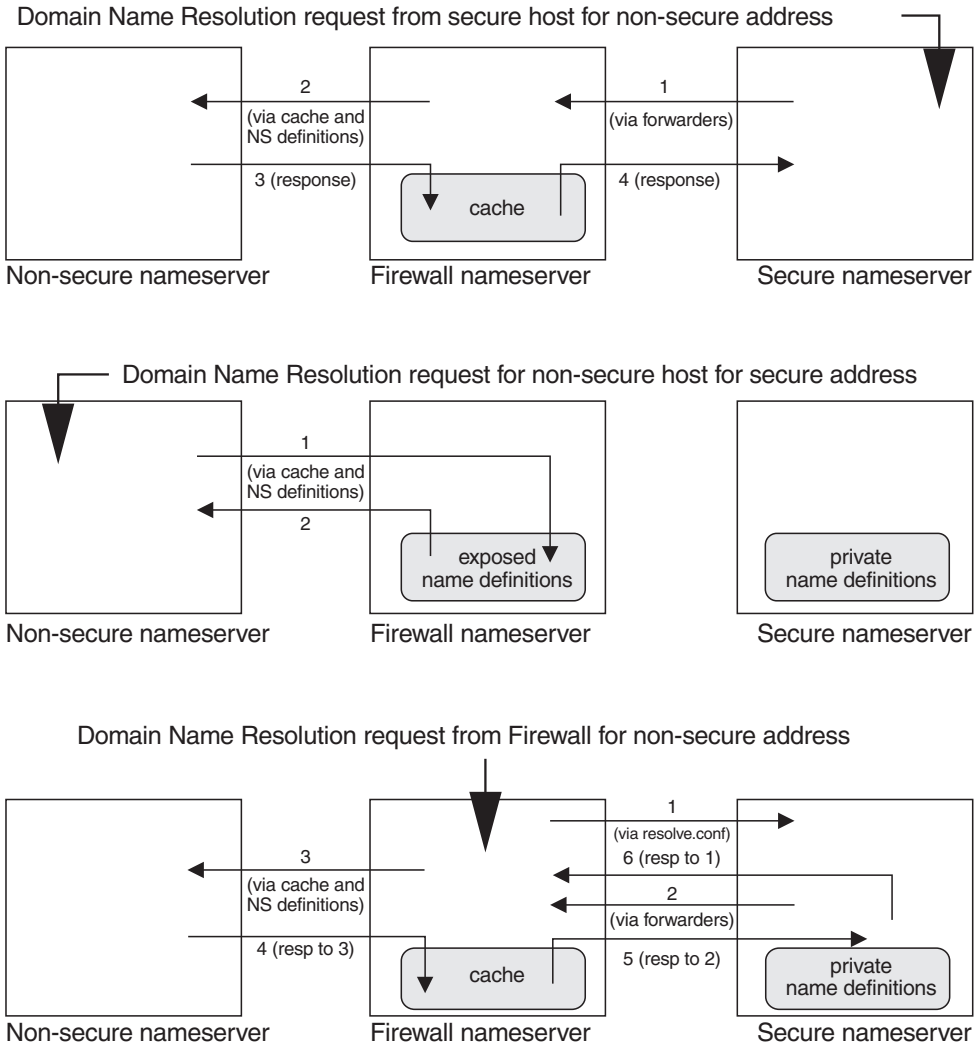


Figure 18. Name Resolution Flows

Overview

The Domain Name Service is the distributed database system that maps host machine names to IP addresses, and vice versa. For each host, the name server can contain IP addresses, nicknames, mailing information, and available well-known services (for example, FTP or Telnet). In the DNS, each host name is composed of a set of naming qualifiers called domains. The qualifiers are separated by periods, and denote a hierarchical relationship, where each qualifier represents an increasingly higher domain level.

The DNS consists of three major parts:

resource records (RR) and the domain name space
: are specifications for the tree-structured domain name space and the data associated with the names

resolver routines
: resolve host names into IP addresses based on data from name servers in response to client requests

name servers
: are the server programs that hold the information about the name space's tree structure and the data associated with it.

DNS on the Firewall

After you install a firewall to protect your secure network, you will want to isolate the domain name services that are accessible inside the secure network from those accessible to those outside the secure network, so that your internal network structure is not visible outside. Accordingly, the purpose of DNS on the firewall is to provide full domain name service to hosts inside the secure network while providing minimal information to hosts outside the secure network.

How It Works

Figure 19 shows how DNS works on OS/390 Firewall Technologies.

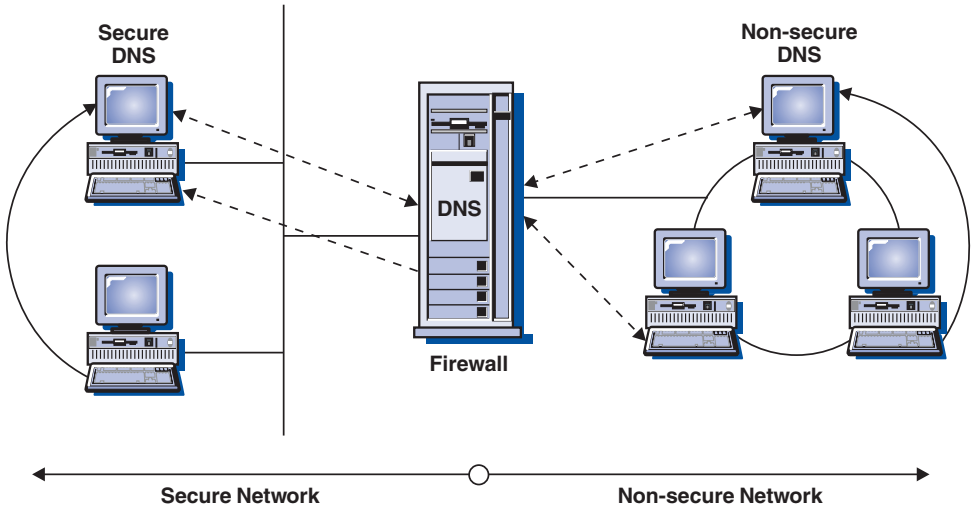


Figure 19. DNS on the Firewall

DNS requests from the inside network are sent to the name server inside the secure network. If the query cannot be answered by the inside name server, the query is forwarded to the firewall. If the query cannot be answered by the firewall name server, the query is forwarded to a name server on the outside network. Once the query is resolved, the answer is returned to the originating name server which sends the answer to the originator of the query.

In this recommended configuration, the firewall machine has one adapter (token ring or ethernet) connecting the firewall to the secure network, and one connecting it to the nonsecure network. Each adapter has a different IP address; therefore the firewall machine will appear to be part of the nonsecure network to the users in that network, but will appear to be a part of the secure network to the users behind the firewall.

For example, if the firewall machine has the host name **firewall**, it will be known to users in secure domain **secure.company.com** and users on the firewall as:

```
host name  : firewall.secure.company.com
IP address : sss.sss.sss.sss

where sss.sss.sss.sss is the IP address of the secure adapter
```

The firewall machine will be known to users in the nonsecure domain "nonsecure.company.com" as:

```
host name  : firewall.nonsecure.company.com
IP address : nnn.nnn.nnn.nnn

where nnn.nnn.nnn.nnn is the IP address of the nonsecure adapter
```

The Recommended Configuration

Three domain name servers are required to accomplish this recommended configuration:

- one on the firewall host

  This server responds to requests from outside your network, and resolves outside host names in response to requests from inside the secure network.

  The domain name server on the firewall responds to external requests only with the IP address of the firewall host, effectively hiding the structure of the internal network. The OS/390 Firewall Technologies product provides this domain name server.

- one inside the secure network (**secure**) name server

  This server resolves secure addresses in response to requests from inside the secure network.

  The domain name server inside the internal network knows all the IP addresses of all the hosts in the internal network. It consults the firewall domain name server to satisfy any requests it cannot. You install and maintain this domain name server.

- one outside the secure network (the **nonsecure** name server)

  This server resolves addresses of hosts outside the secure network that the name server on the firewall cannot resolve. The address of this server is usually provided by your Internet Service Provider.

Setting Up DNS on the Firewall

The following sections detail the changes to make to your DNS configuration. **fwdns** allows you to specify the following:

stack (stack=)
: is the stack name of the TCP/IP stack This parameter provides a stack name to identify the location of the DNS configuration files for each stack.

  This parameter is required only if you have defined more than one stack using the **fwstack** command; otherwise, this parameter is optional.

secure domain name (secdomain=)
: is the domain name of your internal, secure network.

secure domain name server(s) (secaddr=)
: contains the IP addresses of domain name server(s) that are inside, and provide name resolution only for your internal network.

non-secure domain name server (remaddr=)
: contains the IP addresses of the domain name servers outside your secure network. These are the domain name servers to which your internal domain name servers go for authoritative name resolution outside your own network.

resolver configuration file (resolver=)
: is where you specify (among other things) if the UNIX services resolver should use a name server or not. **fwdns** allows you to specify the UNIX services resolver file to be used instead of the default UNIX services resolver file **/etc/security/fwstacks/**<*STACK*>**/fwresolv.conf**.

force specifies that the change is made without a confirmation prompt.

fwdns creates or modifies the following files:

/etc/security/fwstacks/<*STACK*>**/fwnamed.boot**
main configuration file read by the **named** daemon

/etc/security/fwstacks/<*STACK*>**fwnamed.ca**
hints file (contains the names and IP addresses of the authoritative root domain name servers)

/etc/security/fwstacks/<*STACK*>**/fwnamed.loc**
reverse resolution data file for local host

/etc/security/fwstacks/<*STACK*>**/fwresolv.conf**
default UNIX services resolver configuration file (You may optionally specify your own **resolv.conf** file.)

Note: For an explanation of resolver files, see *OS/390 IBM Communications Server: IP Configuration Guide*, SC31-8725.

As a result, the firewall becomes a caching-only name server that caches all unresolved requests and forwards them to the non-secure name server.

You may wish to further change this configuration to suit the needs and specific features of your network. For instance, you could configure the name server on the firewall to provide name/address mappings for resources in the secure network that you want to reveal; for example, a web server that needs to be accessed by hosts in the nonsecure network.

Also, if you do not have a secure name server, an alternative configuration is presented in Appendix D, “Alternative DNS Configuration” on page 451.

Changes to DNS Settings

Make the following changes to the files described:

Resolver Configuration File (Optional): You may optionally specify your own resolver file with **fwdns resolver=**.

If you want to use a PDS data set (rather than an HFS file) for the resolver file, you must change it based on the contents of the default file; that is, comment out the existing domain and name server statements and add the statements found in **/etc/security/fwstacks/**<*STACK*>**/fwresolv.conf**. The TCPIPJOBNAME statement in **/etc/security/fwstacks/**<*STACK*>**/fwresolv.conf** must match that in the UNIX services resolver file used by **named**; which in turn gets the value from the **fwdns.cfg** configuration file (optional file in **/etc/security/fwstacks/**<*STACK*>). If the statements do not match, you should verify that this UNIX services resolver file was used with the firewall stack.

Secure Name Server: If you have a secure name server, you must have the following information in the data file for the secure domain:

1. Add the IP address of the secure adapter of the firewall to the data or resource file on the secure name server. This way, the secure adapter can be referenced by name; in this example, 'firewall'.

   ```
   firewall      99999999  IN  A   sss.sss.sss.sss
   ```

2. Add the following to the boot file (**/etc/named.boot**), to enable the secure name server to forward unresolved requests to the firewall:

   ```
   forwarders     sss.sss.sss.sss
   cache                          /etc/named.cache
   ```

Also, **/etc/named.cache** should be:

```
. 99999999 IN NS firewall.secure.company.com.
firewall.secure.company.com. 99999999 IN A sss.sss.sss.sss
```

Note: You must activate the changes to the secure name server before starting the UNIX services name server.

3. Use the **nslookup** or **onslookup** command to verify that the secure name server can do reverse resolution of its own IP address. If you cannot successfully do reverse resolution, you will need to make the necessary changes to the DNS configuration on the secure name server.

Starting the UNIX Services Name Server

Before starting the UNIX services name server (**named**):

1. Modify the sample start procedure `TCP.SEZAINST(NAMED)` that you moved to SYS1.PROCLIB, as follows:
2. Change the boot file path from **/etc/named.boot** to **/etc/security/fwstacks/**<*STACK*>**/fwnamed.boot**
3. Set the environment variable **RESOLVER_CONFIG** to point to the same file as you specified for the *ResolverFile* parameter of the **fwdns** command

   For example:

   ```
   //NAMED PROC B='/etc/security/fwstacks/<stack>/fwnamed.boot',P='53',
   //    R='ENVAR("RESOLVER_CONFIG=/etc/security/fwstacks/<stack>/fwresolv.conf")'
   //************************************************
   //NAMED  EXEC PGM=EZANSNMD,REGION=0K,TIME=NOLIMIT,
   //  PARM='POSIX(ON) ALL31(ON) &R/ -b &B -p &P'
   ```

4. Issue **start named** from the OS/390 operator console.
5. Check the **syslog** messages after starting **named** for the first time to ensure that it is set up correctly.

Chapter 13. Internet Protocol (IP) Security

IP Security enables secure communications over the Internet and within company networks by securing data traffic at the IP layer. This allows individual users or organizations to secure traffic for all applications, without having to make any modifications to the applications. Therefore the transmission of any data, such as e-mail or application-specific company data, can be made secure.

The mechanism for securing data between two nodes is accomplished by creating a virtual tunnel between two hosts. This is also referred to as creating a Virtual Private Network (VPN). The secure tunnel encapsulates all IP traffic between the two hosts in a manner specified by the user. It provides data integrity, privacy, and authentication depending on how the tunnel is defined.

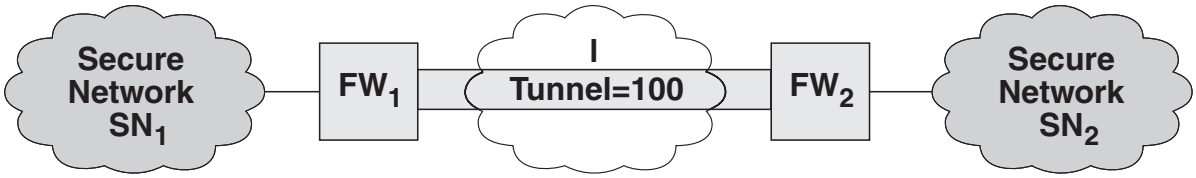


Figure 20. Tunnel, All IP Traffic between Two Secure Networks. FW_1 and FW_2 represent non-secure interface IP address and mask. SN_1 and SN_2. represent any host in the secure network. The shaded Tunnel area of the picture represents the VPN.

Benefits of a Virtual Private Network (VPN)

A virtual private network is an extension of an enterprise's private intranet across a public network such at the Internet. It creates a secure private connection through what is essentially a private tunnel. VPNs securely convey information across the Internet connecting remote users, branch offices, and business partners/suppliers into an extended corporate network. Internet service providers (ISPs) offer cost-effective access to the Internet through direct lines or local telephone numbers. Therefore, companies can eliminate their current networking options that use more expensive leased lines, long-distance calls, and toll-free telephone numbers.

IPSec Architecture

The IP Security (IPSec) Working Group of the Internet Engineering Task Force (IETF) has defined an open architecture and an open framework, known as "IPSec". IPSec is called a framework because it provides a stable, long-lasting base for providing network layer security.

IPSec protects your data traffic in three ways, using robust cryptographic techniques:

1. **Authentication** - the process by which the identity of a host or end point is verified
2. **Integrity Checking** - the process of ensuring that no modifications were made to the data while in-transit across the network
3. **Encryption** - the process of ensuring that no modifications were made to the data while in transit across the network.

Authentication algorithms provide proof of the sender's identity and data integrity by using a cryptographic hash function to process a packet of data (with the immutable IP header fields included) with a secret key to produce a unique digest. On the receiver's side, the data is de-capsulated using the same function and key. If either the data has been altered or the sender's key was not valid, the datagram is discarded.

© Copyright IBM Corp. 1997, 2000

Encryption uses a cryptographic algorithm to modify and randomize the data using a certain algorithm and key to produce cyphertext. Encryption makes the data unreadable while in transit. Once received, the data is recovered using the same algorithm and key (with symmetric encryption algorithms). Encryption must occur with authentication to verify the data integrity of the encrypted data.

IPSec is a security architecture in the IP layer of TCP/IP that provides security services to ensure packet authentication, integrity and confidentiality. IPSec is essentially an encapsulation architecture, namely, one that defines the syntax and semantics of placing one packet inside another. The principal IPSec protocols are:

- IP Authentication Header (AH) which provides data origin authentication and data integrity.
- IP Encapsulating Security Payload (ESP) which provides data confidentiality, authentication and integrity.

The protocol formats for AH and ESP are independent of the cryptographic algorithms used to perform the authentication and encryption. Once an IP security association is established between two systems, they can perform message encryption and message authentication. A security association is simply a set of items of information which, when shared between two communicating systems, enables the two systems to communicate in a specified way.

The encryption algorithms available include Triple Data Encryption Standard (DES), DES, as well as Commercial Data Masking Facility (CDMF). Keyed Message Digest-5 and two versions of the Hashed-Based Message Authentication Code (HMAC) algorithm are used to perform authentication.

The ESP or AH protocols can be applied alone or in combination or even nested within another instance of itself. With these combinations, authentication and/or encryption can be provided between a pair of communicating hosts, between a pair of firewalls, or between a host and a firewall.

The OS/390 Firewall Technologies supports the original IPSec standards described in Request for Comments (RFCs) 1825 through 1829 as well as the latest standards described in RFCs 2401 through 2406 and 2410. The original standards continue to be supported to ensure backward compatibility.

Security Associations

The building block on which secure communications is built is a concept known as a security association. Security associations (SAs) relate a specific set of security parameters to a type of traffic. With IP Security-protected data, a separate SA exists for each direction and for each header type, AH or ESP. The information contained in the SA includes the IP addresses of the communicating parties, a unique identifier known as the Security Parameters Index (SPI), the algorithms selected for authentication and/or encryption, the authentication and encryption keys, and the key lifetimes. The goal of key management is to negotiate and compute the SAs that will be used when protecting IP traffic.

Tunnels and Key Management

To set up a secure communication between two hosts, SAs must be negotiated and managed during the use of the tunnel. Key management is a complex issue for IP Security and new standards are being developed to ensure interoperability. Two types of tunnels are supported by the OS/390 Firewall Technologies. Each uses different key management techniques. They are:

Manual tunnels (static, IETF-standard)
ISAKMP tunnels (dynamic, IETF-standard)

Manual Tunnels

Manual tunnels provide backward compatibility to all header types and will interoperate with non-IBM machines or those that do not have ISAKMP support. The disadvantage of manual tunnels is that the key values are static. In other words, the encryption and authentication keys are the same for the life of the tunnel and must be manually updated. The manual tunnel support has the widest choice of header and encryption options.

Manual tunnels may be used between a host running the OS/390 Firewall Technologies support and any other machine running IP Security and having a common set of cryptographic and authentication algorithms. Almost all vendors offer Keyed_MD5 with DES, or HMAC_MD5 with DES. This is a base subset that works with almost all implementations of IP Security.

The ability to inter-operate with another OS/390 system or any other compatible IBM platform is simplified by using export/import capability of the **fwtunnl** command or Virtual Private Network selection on the configuration client. This capability allows a tunnel definition to be exported from one system and imported into another. Detailed information about the format of the export files is in Appendix F, “FWTUNNL Export File Formats” on page 457.

The **fwtunnl** command and configuration client provide the ability to define and display the complete list of parameters which comprise a tunnel definition. This capability should be used when inter-operating with a system that does not support the OS/390 export file formats. Manual configuration can be performed in one of two ways:

- Define and list the tunnel on OS/390. At this point someone must examine the tunnel definition, understand the format of the displayed data and manually input this data into the non-IBM system using whatever mechanism is provided.
- Define the tunnel on the non-OS/390 system and determine the values of all parameters which define the tunnel. At this point the **fwtunnl** command or configuration client must be used to input all of the same tunnel definition parameters on the OS/390 system.

Only after performing one of these steps does the same security association exist on both systems.

Use the **fwtunnl** command or Virtual Private Network selection on the configuration client to add, delete, change, import, export, activate, deactivate, and shutdown a manual tunnel. See “fwtunnl” on page 316 for syntax and command information. See “Configuring Tunnels” on page 130 for configuration client information. The **fwtunnl** command also displays the currently active manual tunnels.

Dynamic Tunnels (ISAKMP)

Dynamic tunnels are based on the ISAKMP standards developed by the IETF. Dynamic tunnels negotiate and refresh security parameters and exchange keys securely. Two types of authentication are supported - pre-shared key and digital signature.

The negotiation uses a two-phase approach. The first phase authenticates the communicating parties and specifies the method for security communications. In phase two, IP security associations (SAs) are negotiated and keys are exchanged.

The following ISAKMP support is available:

- Authentication with pre-shared keys and digital signatures
- Use of main mode (identity protect mode) and aggressive mode
- Support for Diffie Hellman groups 1 and 2
- ESP support for triple DES, DES, Null, and authentication with HMAC_MD5 and HMAC_SHA
- AH support for HMAC_MD5 and HMAC_SHA

The ISAKMP support is based on the following IETF standards:

- RFC 2407 - The Internet IP Domain of Interpretation for ISAKMP
- RFC 2408 - The Internet Security Association and Key Management Protocol (ISAKMP)
- RFC 2409 - The Internet Key Exchange

Choosing a Tunnel Type

The decision to use manual tunnels or dynamic tunnels depends on the tunnel support of the remote end and the type of key management desired. Dynamic tunnels are preferable (when available) because they offer secure key negotiation and key refreshment in an industry-standard way. They also take advantage of the new IETF ESP and AH header types and provide replay protection.

If the remote end does not support dynamic tunnels, or uses one of the algorithms supported only by manual tunnels, manual tunnels should be used. Manual tunnels ensure interoperability with a large number of hosts. Because the keys are static and difficult to change and may be cumbersome to update, they are not as secure.

Refer to Chapter 14, “Dynamic Virtual Private Network (VPN)” on page 85 for detailed information on the configuration and use of dynamic tunnels.

Defining Tunnel Modes

One of the parameters that must be specified when defining a tunnel is its operational mode. A tunnel can operate in either transport mode or tunnel mode.

Transport mode only protects the transport-layer packet (such as a TCP or a UDP packet) inside an IP packet. In this mode the IP protocol header is first separated from the transport-layer packet, then the transport-layer packet is encapsulated in an IPSec packet, then the IP protocol header is attached to the IPSec packet to form a new IP packet, and finally the length, protocol and header `checksum` fields in the IP protocol header are modified accordingly. The source and destination addresses in the IP protocol header remain unchanged. This mode is used when the endpoints of the security association are the two communicating systems.

Tunnel mode protects the entire IP packet. The entire IP packet is encapsulated in an IPSec packet and a new IP protocol header is constructed and attached at the beginning of the IPSec packet to form a new IP packet. The source and destination addresses may or may not be the same as those in the encapsulated IP packets. This mode is typically used for a security association between two firewalls, or between a firewall and a remote system, for example, whenever either of the two communicating systems is not an endpoint of the tunnel. The source and destination addresses in the new IP Header are the addresses of the tunnel's endpoints.

Tunnel Policy

The data (original IP packets) passing through a tunnel can be either:

- Encrypted (ESP, authentication is optional)
- Authenticated (AH)
- Encrypted and then authenticated (ESP and AH)

The **tunnel policy** determines the level of protection and should be based on your security requirements. A different policy can be used for different IP protocols, for example, telnet may be different from FTP.

The concept of a tunnel carrying encrypted and/or authenticated data is integrated with the IP filtering rules. This provides the level of granularity for defining which IP data packets will be encrypted and/or authenticated at the IP address and port level. Any particular tunnel policy must have the same specification at both ends or the transmitted packets will be discarded.

VPN Customer Scenarios

Let's look at the three most likely business scenarios well suited to the implementation of a VPN solution.

1. Branch office connection network
2. Business partner/supplier network
3. Remote access network

This section provides a general description of those scenarios. For a detailed example on how to configure OS/390 Firewall Technologies in the Branch Office scenario, refer to Appendix C, “Dynamic Virtual Private Networking (VPN) Tunnel (ISAKMP) Examples” on page 437.

Branch Office Connection Network

The branch office scenario securely connects two trusted intranets within your organization. In this case, your security focus is on both protecting your company's intranet against external intruders and securing your company's data while it flows over the public Internet. This differs from the business partner/supplier network where the focus is on enabling your business partners/suppliers access to data in your corporate intranet.

For example, suppose corporate headquarters wants to minimize the costs incurred from communicating to and among its own branches. Today, the company may use frame relay and/or leased lines, but wants to explore other options for transmitting their internal confidential data that will be less expensive, more secure, and globally accessible. By exploiting the Internet, branch office connection VPNs can easily be established to meet the company's needs.

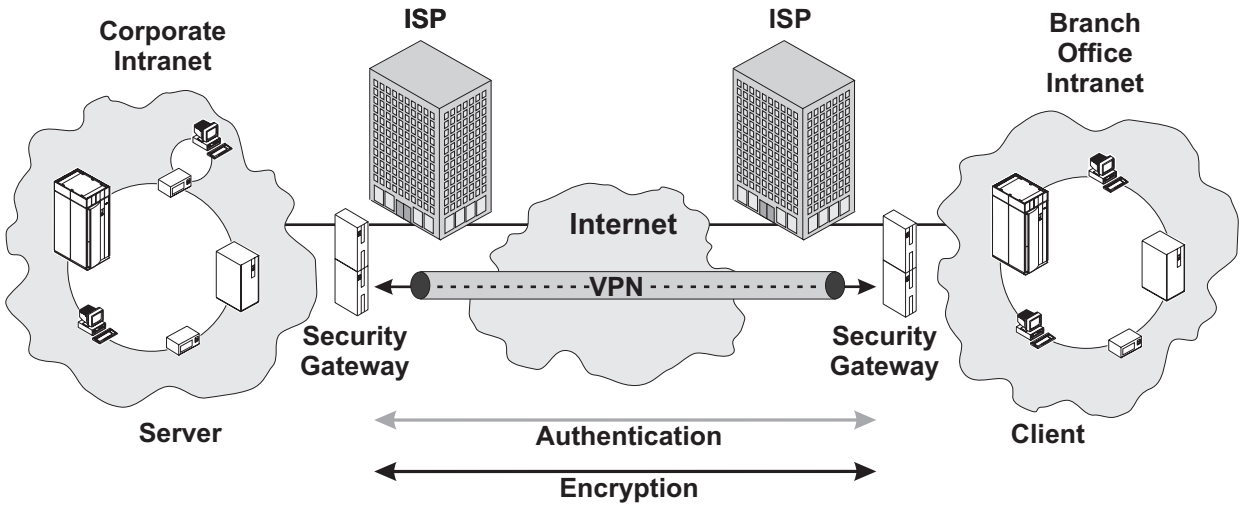


Figure 21. Branch Office Connection Network

This figure shows one way to implement this VPN connection between the corporate headquarters and one of its branch offices is for the company to purchase internet access from an ISP, such as IBM Global Services. IBM firewalls, or routers with integrated firewall functionality, or in some cases an IBM server with IPSec capability, will be placed at the boundary of each of the internets to protect the corporate traffic from internet hackers. With this scenario, the clients and servers need not support IPSec technology,

since the IPSec enabled firewalls (or routers) provide the necessary data packet authentication and encryption. With this approach, any confidential information is hidden from untrusted internet users, with the firewall denying access to potential attackers.

With the establishments of the branch office connection VPNs, the company's corporate headquarters is able to communicate securely and cost effectively to its branches, whether located locally or far away. Through VPN technology, each branch can also extend the reach of its existing intranet to incorporate the other branch intranets, building an extended, enterprise-wide corporate network.

And, as in the business partner/supplier network scenario, this company can easily expand this newly created environment to include its business partners, suppliers, and remote users, through the use of open IPSec technology.

Business Partner/Supplier Network

Industry-leading companies will be those that can communicate inexpensively and securely to their business partners, subsidiaries, and vendors. Many companies chose to implement frame relay and/or purchase leased lines to achieve this interaction. But this is often expensive, and geographically may be limited. VPN technology offers an alternative for companies to build a private and cost-effective extended corporate network with worldwide coverage exploiting the Internet or other public network.

Suppose you are a major parts supplier to a manufacturer. Because it is critical that you have the specific parts and quantities at the exact time required by the manufacturing firm, you always need to be aware of the manufacturer's inventory status and production schedules. Perhaps you are handling this interaction manually today, and have found it to be time consuming, expensive and maybe even inaccurate. You'd like to find an easier, faster, and more effective way of communicating. However, given the confidentiality and time-sensitive nature of this information, the manufacturer does not want to publish this data on their corporate Web page or distribute this information monthly using an external report.

To solve these problems, the parts supplier and manufacturer can implement a VPN, as shown in the figure below. A VPN can be built between a client workstation in the parts supplier's intranet, directly to the server residing in the manufacturer's intranet. The clients can authenticate themselves either to the firewall protecting the manufacturer's intranet, directly to the manufacturer's server (validating that they are who they say they are), or to both, depending on your security policy. Then a tunnel can be established, encrypting all data packets from the client, through the Internet, to the required server.

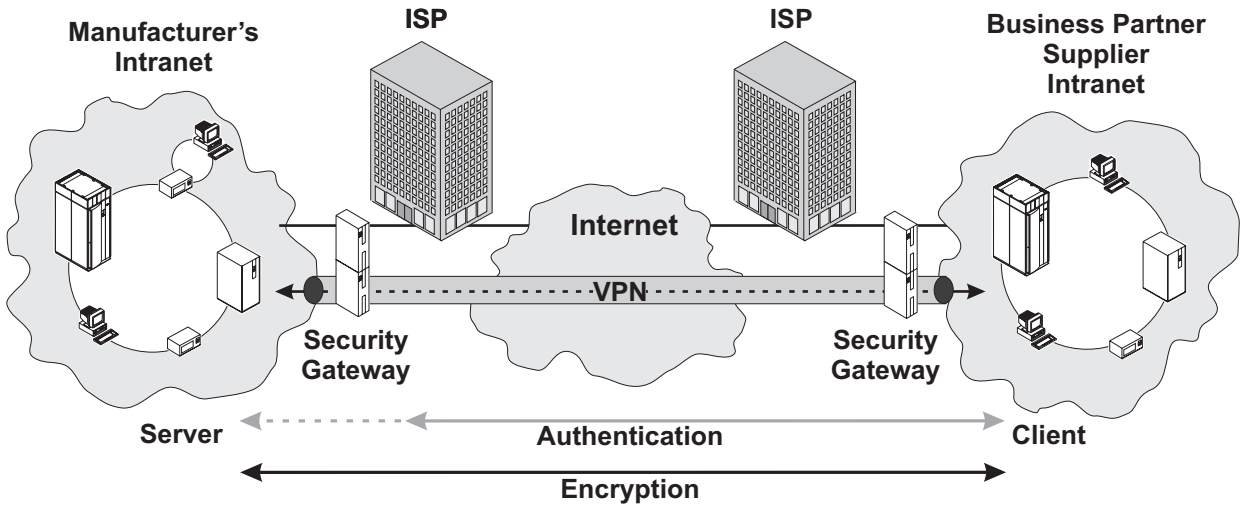


Figure 22. Business Partner/Supplier Network

With the establishment of this VPN, the parts supplier can have global, access to the manufacturer's inventory plans and production schedule at all times during the day or night, minimizing manual errors and eliminating the need for additional resources for this communication. In addition, the manufacturer can be assured that the data is securely and readily available to only the intended parts suppliers.

One way to implement this scenario is for the companies to purchase internet access from an internet service provider (ISP). Then, given the lack of security of the internet, deploy either an IBM firewall or IPSec-enabled router, or an IBM server with IPSec capability to protect the intranets from intruders. If end-to-end protection is needed, then both the client and server machines need to be IPSec-enabled as well.

Through the implementation of this VPN technology, the manufacturer would easily be able to extend the reach of their existing corporate intranet to include one or more parts suppliers (essentially building an extended corporate network) while enjoying the cost-effective benefits of using the Internet as their backbone. And, with the flexibility of open IPSec technology, the ability for this manufacturer to incorporate more external suppliers is limitless.

Remote Access Network

A remote user, whether at home or on the road, wants to be able to communicate securely and cost-effectively back to his/her corporate intranet. Although many still use expensive long-distance and toll-free telephone numbers, this cost can be greatly minimized by exploiting the Internet. For example, you are at home or on the road, but need a confidential file on a server within your intranet. By obtaining Internet access in the form of a dial-in connection to an ISP such as IBM Global Services, you can communicate with the server in your intranet and access the required file.

One way to implement this scenario is to use an eNetwork VPN IPSec-enabled remote client and firewall, as shown in Figure 23. The client accesses the Internet via dial-up to an ISP, and then establishes an authenticated and encrypted tunnel between itself and the firewall at the intranet boundary.

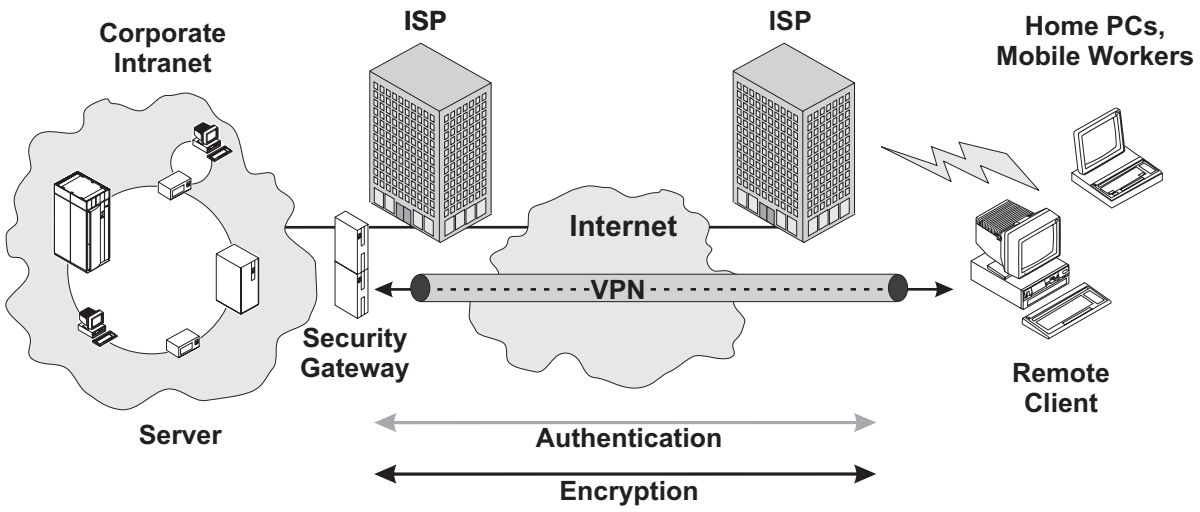


Figure 23. Remote Access Network

By applying IPSec authentication between the remote client and the firewall, you can protect your intranet from unwanted and possibly malicious IP packets. And by encrypting traffic that flows between the remote host and the firewall, you can prevent outsiders from eavesdropping on your information.

Chapter 14. Dynamic Virtual Private Network (VPN)

The critical elements of IP Security are Security Associations (SA) and the information that they provide with regard to identifying the partners of a secure communications channel, the cryptographic algorithms and keys to be used. The ISAKMP protocol provides a framework for exchanging messages to automate the negotiation of Security Associations. The Internet Key Exchange (IKE) framework, also referred to as Internet Security Association Key Management Protocol/Oakley is used in conjunction with the ISAKMP protocol to automate the generation and refresh of cryptographic keys. The ability to perform these functions with minimal manual configuration of machines will be a crucial element as a VPN grows in size.

The secure exchange of keys is the most critical factor in establishing a secure communications environment. No matter how strong your authentication and encryption are, they are worthless if your key is compromised. The concept and basic operation of ISAKMP/Oakley is addressed in this chapter.

ISAKMP Overview

The ISAKMP procedures deal with initializing the keys, so they must be capable of running over networks where no security can be assumed to exist. Hence, the ISAKMP protocols use the most complex and processor-intensive operations in the IPSec protocol suite.

In addition, the ISAKMP methods have been designed with the explicit goals of providing protection against several well-known exposures:

- Denial-of-Service: The messages are constructed with unique pseudo-random numbers that can quickly identify and reject incorrect messages without the need to execute processor-intensive cryptographic operations.
- Man-in-the-Middle: Protection is provided against the common attacks such as deletion of messages, modification of messages, reflecting messages back to the sender, replying of old messages, and redirection of messages to unintended recipients.
- Lack of Perfect Forward Secrecy (PFS): Compromise of past keys provides no useful clues for breaking any other key, whether it occurred before or after the compromised key. That is, each refreshed key will be derived without any dependence on predecessor keys.

Phases of ISAKMP

The robustness of any cryptography-based solution depends much more strongly on keeping the keys secret than it does on the actual details of the chosen cryptographic algorithms. Hence, the IETF IPSec Working Group has prescribed a set of extremely robust ISAKMP exchange protocols. The ISAKMP protocol is implemented at the application layer of TCP/IP and communicates using UDP port 500. The protocol uses a 2-phase approach:

Phase 1: This set of negotiations establishes a master secret from which all cryptographic keys will subsequently be derived for protecting the user's data traffic. In the most general case, public key cryptography is used to establish an ISAKMP security association between systems, and to establish the keys that are used to protect the ISAKMP messages that flow in the subsequent Phase 2 negotiations. Phase 1 is concerned only with establishing the protection suite for the ISAKMP messages themselves, but it does not establish any security associations or keys for protecting user data.

In Phase 1, the cryptographic operations are the most processor-intensive but need to be done infrequently, and a single Phase 1 exchange can be used to support multiple Phase 2 exchanges. As a rule,

© Copyright IBM Corp. 1997, 2000

Phase 1 negotiations are executed once a day or maybe once a week, while Phase 2 negotiations can be executed as often as once every few minutes.

Phase 2: Phase 2 exchanges are less complex, since they are used only after the security protection suite negotiated in Phase 1 is activated. A set of communicating systems negotiate the security associations and keys that will protect user data exchanges. Phase 2 ISAKMP messages are protected by the ISAKMP security association generated in Phase 1. Phase 2 negotiations generally occur more frequently than Phase 1. The ISAKMP protocol offers a solution even when the remote host's IP address is not known in advance. ISAKMP allows a remote host to identify itself by a permanent identifier, such as a name or an e-mail address. The ISAKMP Phase 1 exchanges then authenticate the remote host's permanent identity using public key cryptography:

- Certificates create a binding between the permanent identifier and a public key. Therefore, ISAKMP's certificate-based Phase 1 message exchanges can authenticate the remote host's permanent identify.
- Since the ISAKMP messages themselves are carried within IP datagrams, the ISAKMP partner (for example, a firewall or destination host) can associate the remote host's dynamic IP address with its authenticated permanent identity.

See Appendix H, “Initializing Security Associations with ISAKMP and IKE” on page 463 for a detailed discussion of how ISAKMP/Oakley exchanges authenticate the remote host to its peer and set up the security associations dictated by its corporate VPN policy.

Authentication in ISAKMP

The ISAKMP protocol requires that ISAKMP peers authenticate themselves as part of phase 1 processing. The OS/390 Firewall Technologies' ISAKMPD server supports authentication via either pre-shared keys or RSA Signatures.

With pre-shared key authentication both ISAKMP peers agree to an arbitrary value. This value is included in the calculation that dynamically creates the secret encryption key that will be used to encrypt messages between the two peers. The ability to successfully send a message encrypted with this secret key proves that the pre-shared value is known.

In the RSA Signature-based authentication each ISAKMP server signs a piece of information with their private key. If the peer can successfully verify the signature with the public key contained in the ISAKMP server's certificate, and can also verify the server's certificate, then that ISAKMP server is the entity identified by the certificate. A certificate is a binding between an entity and the public key portion of an asymmetrical cryptographic key pair. A third party, called a Certificate Authority (CA), certifies the identity of the entity involved and that the entity does possess the associated private key.

In the RSA Signature-based authentication the local ISAKMP server needs to obtain the certificate that is bound to the public key/private key pair that the remote ISAKMP server will use to generate its signature. The ISAKMP protocol allows the local ISAKMP server to request this certificate from the remote ISAKMP server. The remote ISAKMP server may have many certificates to choose from. To make sure the remote ISAKMP server picks an acceptable certificate for signing, the local ISAKMP server can send some information to guide the remote ISAKMP server. In the OS/390 ISAKMP implementation part of this information is an indication of which certificate authority should have issued (i.e. signed) the certificate the remote ISAKMP server picks. In essence, the local ISAKMP server says “Use a certificate signed by this CA to authenticate yourself and then send me that certificate.”

The ISAKMP protocol requires that ISAKMP peers identify themselves as part of phase 1 processing by sending a message containing their identity. In RSA signature-based authentication an ISAKMP server's

identity is also contained in the subject name field and/or a subject alternate name extension of it's certificate. The identity contained in the certificate must match the identity sent as part of phase 1 processing.

Steps to Defining Dynamic VPNs

All the OS/390 Firewall Technologies enhancements for Version 2 Release 8 were made in the area of VPN support. Prior to V2R8, VPNs had to be manually established. This meant that the VPN administrators had to manually agree to the attributes of a VPN as well as manually exchange and maintain the encryption keys used by the VPN. The combination of these attributes and encryption keys are referred to as Security Associations (SAs).

Manual procedures for establishing and maintaining VPNs are practical when dealing with a small number of VPNs. But as the number of VPNs increase, manual procedures become time-consuming, costly, and error-prone. In particular, a process for periodically exchanging encryption keys must be established on a VPN basis. The exchange of encryption keys should be done in a secure manner and should be done frequently enough to prevent the compromise of the current set of encryption keys. Maintaining a process for exchanging encryption keys for two or three VPNs may be a manageable task; however, do so for hundreds of VPNs quickly becomes difficult to manage.

The OS/390 Firewall Technologies now includes a new server, ISAKMPD. ISAKMPD implements a set of protocols to dynamically negotiate VPN attributes, dynamically manage VPN encryption keys, and dynamically establish VPNs. The protocols implemented by ISAKMPD are defined by the IPSec working group of the IETF and include:

- RFC 2407 (the Internet Domain of Interpretation for ISAKMP)
- RFC 2408 (the Internet Security Association and Key Management Protocol - ISAKMP)
- RFC 2409 (the Internet Key Exchange - IKE)

On OS/390 a VPN that is created and maintained using these protocols is referred to as a dynamic VPN. The ISAKMP and IKE protocols are widely accepted, as such, many other venders have also implemented these protocols. The ISAKMPD server shipped with the OS/390 Firewall Technologies should interoperate with any implementation that is compliant with RFC 2407, RFC 2408, and RFC 2409.

A dynamic VPN is created between two devices (e.g. ISAKMP servers) that support the ISAKMP/IKE protocols. The device that starts the process of creating a dynamic VPN is known as the initiator. The device with which the initiator is creating a dynamic VPN is known as the responder.

The following high level steps are required to define dynamic VPNs:

- Define policies
  - Key management
  - Data management
  - Dynamic tunnel
- Define key server information
- Define authentication data
- Define anchor filter rules
- Define local activation information
  - Dynamic connections
  - On-demand connections

Each high level step is comprised of a series of lower level steps which will be covered in detail in this chapter. Some of these steps will be performed less frequently than others. In general, defining policy

will be performed less frequently than defining key server information, authentication data, and anchor rules. The frequency at which dynamic connections are defined is dependent on a number of factors. These factors will become more evident when dynamic connections are discussed later in this chapter.

Policy

Policy is used to define the acceptable combination of security services and security attributes for protecting IP traffic. Currently there are three IPSec security services supported by the OS/390 Firewall Technologies.

- The AH and ESP security services are used to protect data sent in the VPN
- The IKE security service is used to protect key management and security association attribute negotiations.

A security service is implemented using a security protocol. For example, the AH security service implements the AH protocol.

There are three types of policy:

- Key management
- Data management
- Dynamic tunnel

Key Management Policy: The key management policy defines an acceptable manner with which to protect ISAKMP messages sent during phase 2 negotiations. In addition, this policy determines how the master secret key will be derived and how ISAKMP peers will authenticate each other. Key management policy is defined using a series of templates. The top most template is a key policy template. The key policy template utilizes key proposal templates which utilize key transform templates.

Figure 24 shows the hierarchy between the various key management templates.

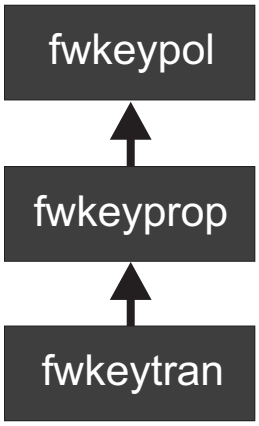


Figure 24. Hierarchy of Key Policy Templates

Key Policy Templates: The key policy, key proposal and key transforms templates are defined using the **fwkeypol**, **fwkeyprop**, and **fwkeytran** commands (or the corresponding configuration client panels).

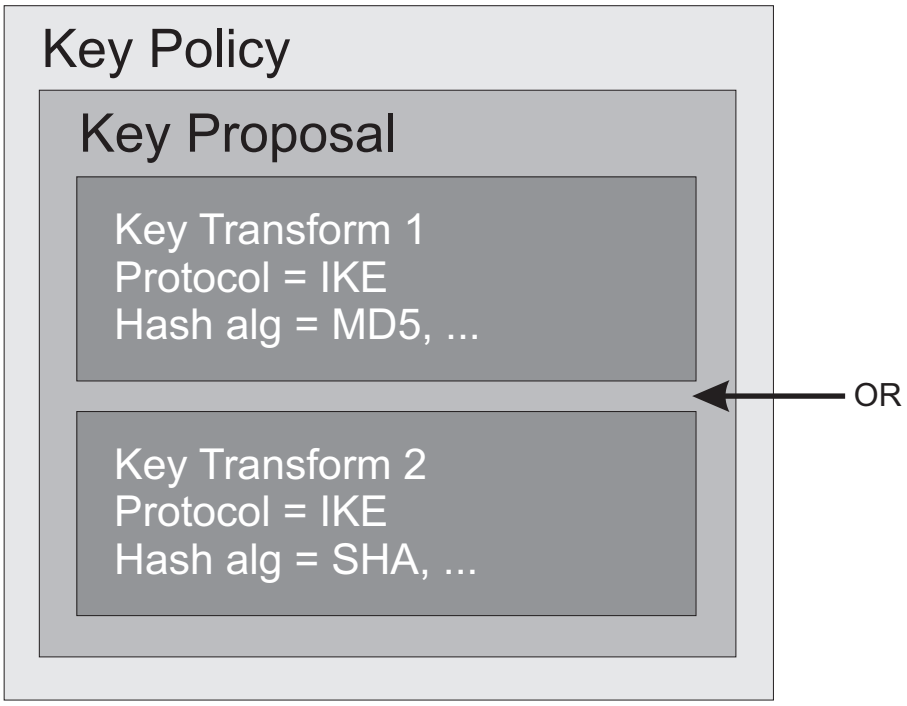


Figure 25. Relationship Between Various Key Management Templates

A key policy template can contain only one key proposal template. There is only one valid IKE protocol. A key proposal template can contain multiple key transform templates each supporting the IKE protocol. Figure 25 shows the relationship between the various key management templates. Multiple key transform templates are treated as if they are combined with a logical OR. For example, use key transform 1 or key transform 2.

The role of key policy in ISAKMP negotiations is as follows:

- Key policy is sent by the initiating ISAKMP server to it's peer. Only one proposal may be sent.
- If the proposal contains multiple key transforms the peer picks one.
- The peer sends back the choice made.

Data Management Policy: Data management policy defines an acceptable manner with which the data sent in a VPN is to be secured. Data management policy is defined using a series of templates. The top-most template is a data policy template. The data policy template utilizes data proposal templates which utilize data transform templates. Figure 26 on page 90 shows the hierarchy between the various data management templates.

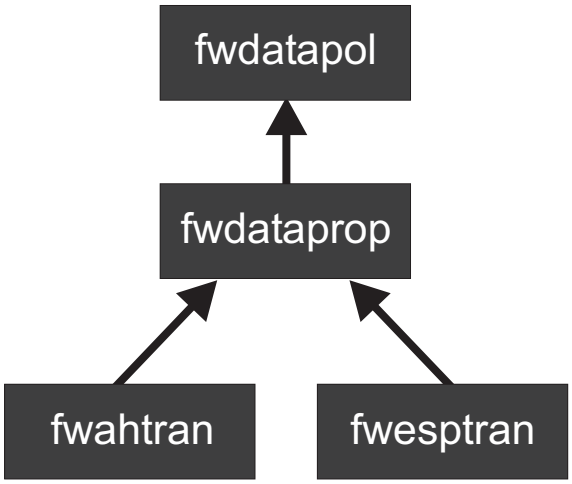


Figure 26. Hierarchy of Data Management Templates

The goal of a data management policy is to define a scheme for managing (i.e. protecting) data in a dynamic VPN. This scheme is not specific to one VPN. Many dynamic VPNs can use the same policy. Data management policy is specified as part of dynamic tunnel policy.

Data Policy Templates: The data policy, data proposal, and data transform templates (AH and ESP) are defined using the **fwdatapol**, **fwdataprop**, **fwahtran**, and **fwesptran** commands (or the corresponding configuration client panels).

Figure 27 shows the relationship between the various data management templates.

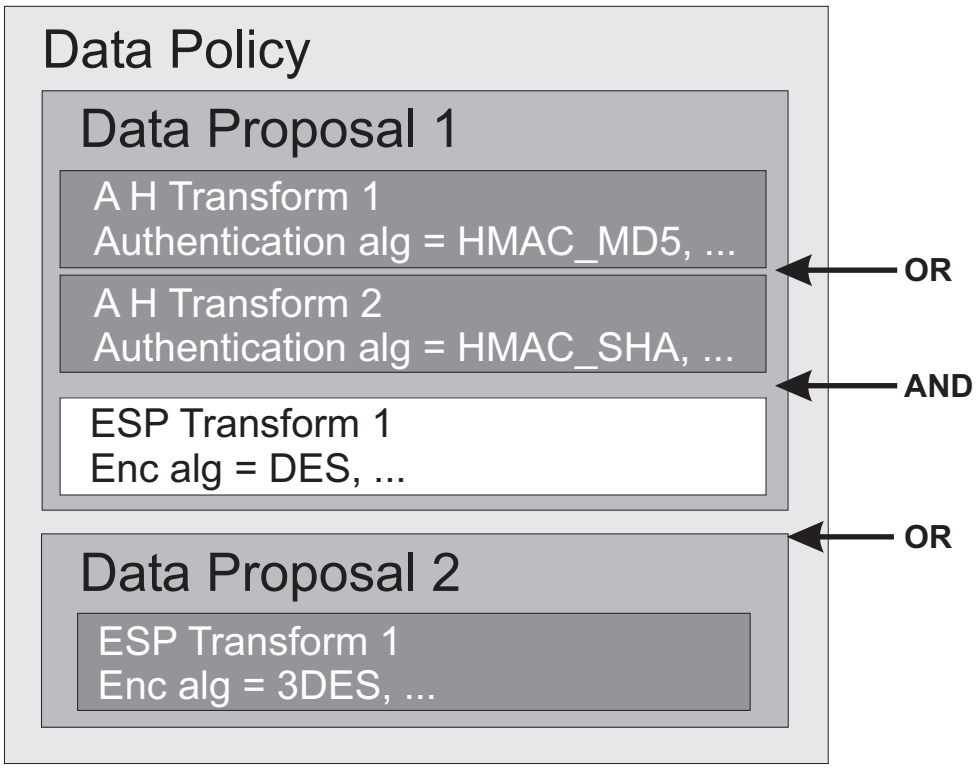


Figure 27. Relationship Between Data Management Templates

A data policy template can contain multiple data proposal templates. When multiple data proposal templates are present they are treated as if they are combined with a logical OR. For example, use data proposal template 1 or data proposal template 2.

A data proposal template can contain multiple AH transform templates and multiple ESP transform templates. Multiple templates of the same transform type are treated as if they are combined with a logical OR. For example, use AH transform 1 or AH transform 2. When both AH and ESP transform templates are specified, they are treated as if they are combined with a logical AND. For example, use an AH transform and an ESP transform.

A data proposal may contain only AH transforms, only ESP transforms, or both AH and ESP transforms. When a data proposal contains both AH and ESP transforms, the ESP protocol is applied before the AH protocol.

The role of data policy in ISAKMP negotiations is as follows:

- Data policy is sent by the initiating ISAKMP server to its peer.
- If multiple proposals are sent, the peer picks one proposal.
- The peer must choose one transform for each security service in the proposal.
- If the proposal contained multiple transforms for the same security service (e.g. multiple AH transforms) the peer picks one.
- The peer sends back the choice made.

Dynamic Tunnel Policy: The dynamic tunnel policy defines acceptable attributes for a dynamic tunnel. A dynamic tunnel is the portion of a dynamic connection that is protected by IPSec. A dynamic connection is the communication circuit between the source and destination of data that is sent through a VPN. A dynamic VPN is comprised of one dynamic tunnel and two or more networks.

In Figure 28, Host A and Host B are sending data to each other. A dynamic tunnel between Gateway A and Gateway B has been established to create a Virtual Private network (VPN) between Network A and Network B. The communication circuit between Host A and Host B is called a dynamic connection.

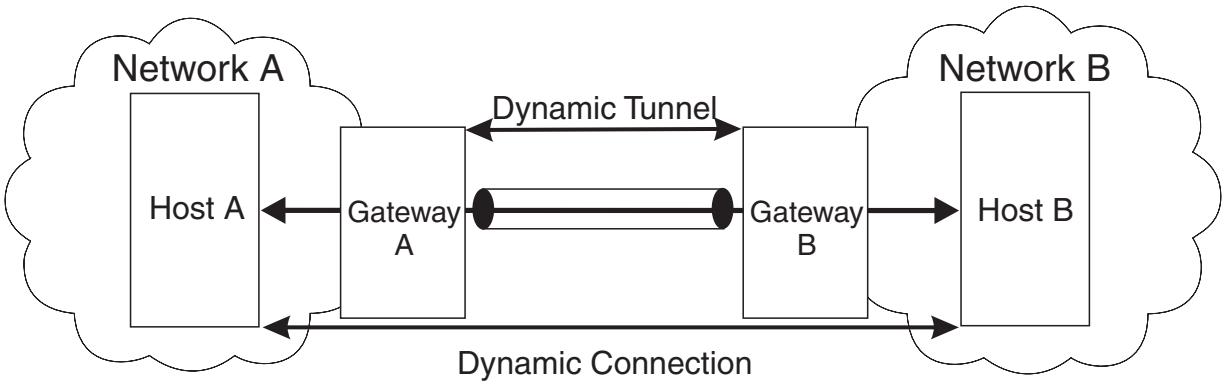


Figure 28. Dynamic Tunnel Example

Dynamic tunnel policies are defined using a dynamic tunnel template. The goal of a dynamic tunnel policy is to define a set of attributes that can be associated with the management of a dynamic tunnel. Dynamic tunnel policies are specified when defining anchor filter rules. Anchor filter rules are a special type of filter rule and are discussed in detail later in this chapter. Dynamic tunnel policy specifies the policy for the management of one set of AH and/or ESP SAs. Dynamic tunnel policy specifies policy for a sequence of

such SAs (i.e. during the life span of a dynamic tunnel it is comprised of multiple sets of AH and/or ESP SAs).

Dynamic tunnel templates are defined using the **fwdyntun** command (or the corresponding configuration client panel). Figure 29 shows the relationship between a dynamic tunnel template and the data management templates.

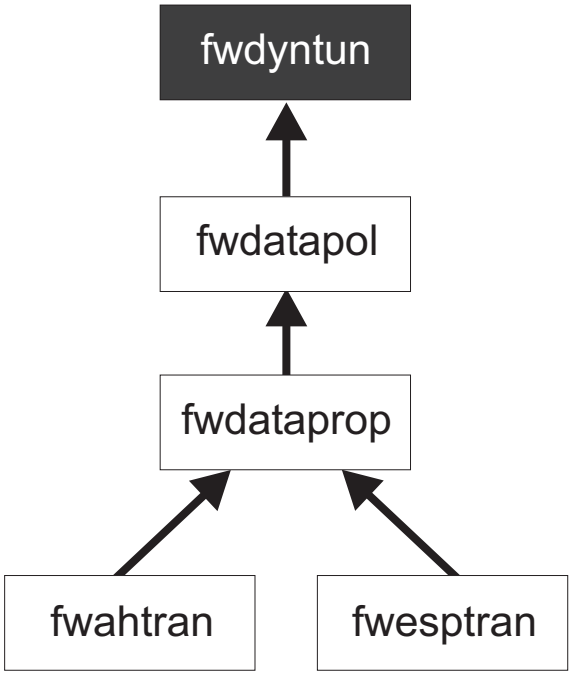


Figure 29. Hierarchy of a Dynamic Tunnel Template and Data Management Templates

Key Server Information

Key Servers: After key management policy has been defined, it needs to be associated with entities that can use it. The entities that can use key management policy are known as key servers. Key servers identify entities that implement the ISAKMP protocol. ISAKMP negotiations are performed between two key servers. Key management policy is associated between a pair of key servers.

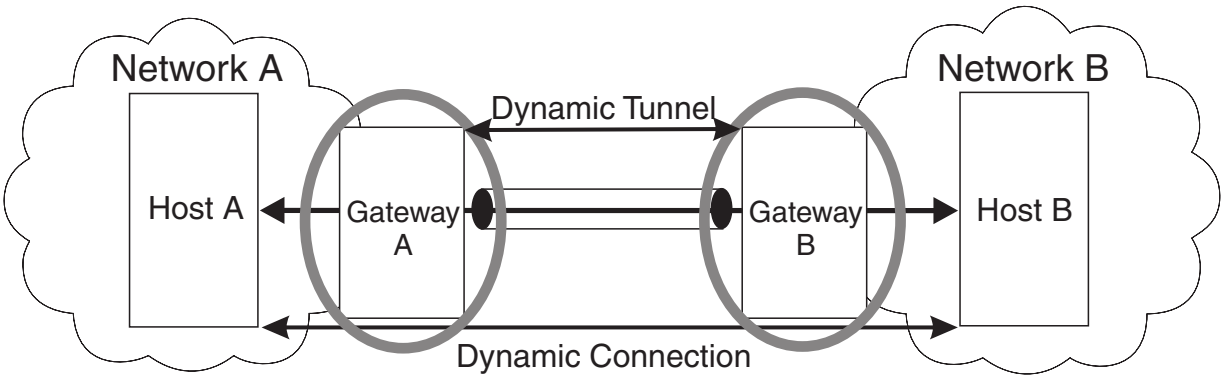


Figure 30. Dynamic Connection Scheme

In Figure 30, Gateway A and Gateway B are the key servers. By definition, they are also the endpoints of the dynamic tunnel.

Key servers can be identified by either a version 4 IP address, a fully-qualified domain name, user@fully-qualified domain name, or an X.500 distinguished name. Key servers can have multiple identities. There are two types of key servers: local key servers and remote key servers. If the key server resides on the system being configured, it is known as a local key server. If it resides on a different system, it is known as a remote key server.

Key servers are defined using the **fwkeysrv** command (or the corresponding configuration client panel).

Key Server Groups: Key server groups are used to associate a key management policy with a local key server and a list of remote key servers. Both a remote key server and the key server group it resides in are specified when the dynamic connection was defined. When a dynamic connection is initiated by the local key server, the key management policy is determined by looking at the key server entry specified when the dynamic connection was defined. When a dynamic connection is initiated by the remote key servers the key management policy is determined by searching for a key server group entry that matches the remote ID and local IP address or host name.

Key Server Groups are defined using the **fwkeysrvgrp** command (or the corresponding configuration client panel). Figure 31 shows the relationship between key server groups, key servers and key management templates.

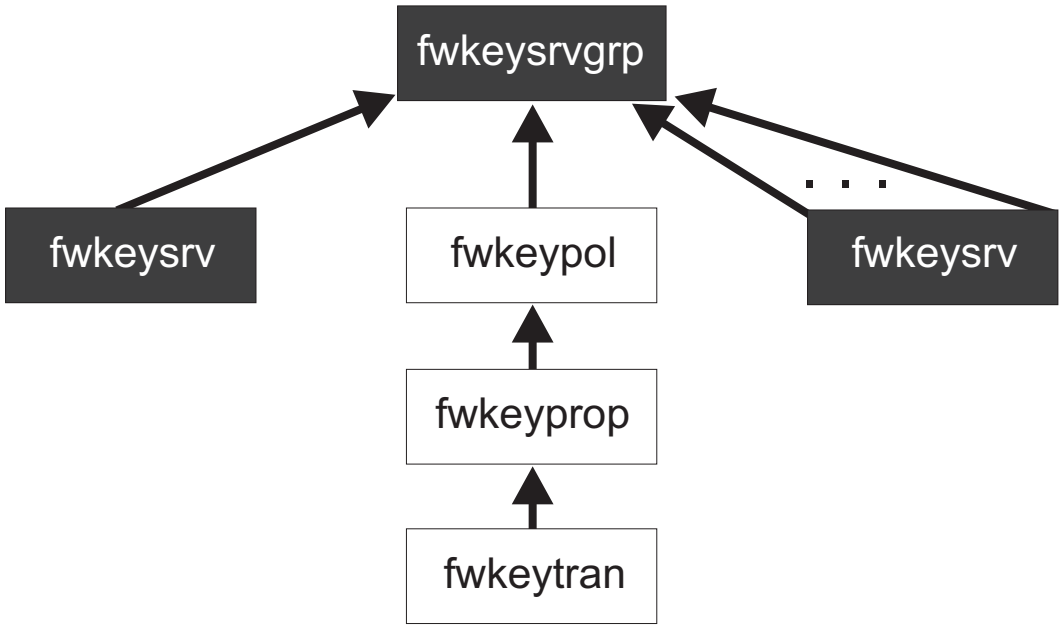


Figure 31. Relationship Between Key Server Groups, Key Servers and Key Management Templates

Authentication Data: A RACF key ring is used to store certificates that will be used by the local key server. This includes the local key server's own certificates, all CA certificates in the CA hierarchies that signed the local key server's certificates, and all CA certificates in the CA hierarchies used to sign a remote key server's certificate. If is expected that the remote key server's certificate will generally be signed by the same CA hierarchy that signed the certificate being used by the local key server. Certificates of remote key servers are not stored on the RACF key ring. The local key server will only require certificates if key policies supporting RSA Signature mode of authentication have been specified.

The RACF **racdcert addring** command is used to create a key ring. The RACF **racdcert connect** command is used to add certificates to a key ring. All certificates must reside on one key ring. The **fwkeyring** command (or the corresponding configuration client panel) is used to identify the owning userID and name of this RACF key ring to ISAKMPD.

Before adding a root CA certificate to the key ring, the administrator must ensure that the certificate is authentic. That is the certificate being added really belongs to the root CA (ie. not an imposture). Determining if the certificate is authentic, can be accomplished through an out of band mechanism such as comparing the hash of the certificate to the published hash value obtained from the trusted source.

racdcert gencert (with the ICSF option) supports storing private keys owned by the ISAKMP server in the Integrated Cryptographic Service Facility (ICSF) Public Key Data Store (PKDS). Operations using the private key stored in ICSF will take advantage of the Integrated Cryptographic Feature (ICRF) hardware available in the new generation of System/390 processors.

Refer to the *OS/390 SecureWay Security Server RACF Command Language Reference*, SC28-1919, for a complete description of the **racdcert** command.

CA Certificates: As previously mentioned, ISAKMPD provides a technique to request that a peer uses a certificate signed by a particular CA. Information identifying the Certificate Authority's certificates (via the RACF key ring label) is defined using the **fwcertauth** command (or the corresponding configuration client panel). This information should be referenced by the **fwauthinfo** command (or the corresponding configuration client panel).

Authentication Information: For each remote key server listed in a key server group, authentication information should be defined. The type of information defined is dependent on the key policy. If pre-shared key mode of authentication is proposed to a remote key server, then a pre-shared key value is required. If the policy supports RSA Signature authentication, then a certificate authority entry should be specified as part of the authentication information.

The information associating a remote key server to the authentication information is done using the **fwauthinfo** command (or the corresponding configuration client panel). Failure to specify Certificate Authority information when RSA Signature mode of authentication is proposed may lead to the peer sending a certificate that is not issued by a supported certificate authority. A key policy can support both modes of authentication.

Figure 32 shows the relationship between key servers, authentication information, key ring and certificate authority definitions. Note that the dotted line between the **fwkeyring** and **fwcertauth** indicates that there is a logical relationship between the two. Namely, that **fwcertauth** entries contain the label of a certificate on the RACF key ring named by the **fwkeyring** entry.

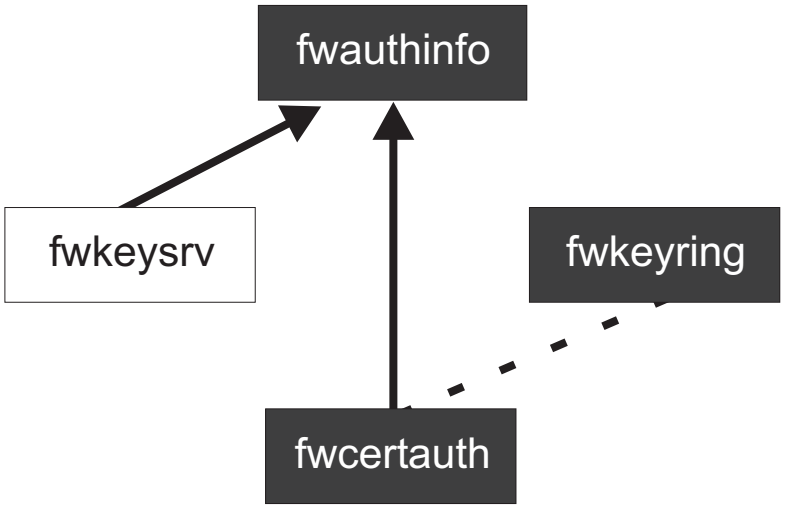


Figure 32. Key Servers, Authentication Information, Key Ring and Certificate Authority Definitions

Anchor Filter Rules

Filter rules are used to define the set of traffic that can flow through a system. When an IP packet is received the filter rules are consulted to determine what action should be taken. On a normal filter rule, one of two actions can be specified - either permit or deny. In the case of manual VPNs, permit rules can also specify the ID of a manual tunnel. If an IP packet matches the selector values of a normal filter rule the action specified (i.e. permit, permit with manual tunnel, or deny) is applied to the packet. Selector values include information such as source address, source port, source protocol, destination address, destination port, and destination protocol. After dynamic tunnel policy has been defined, it needs to be associated with objects that allow dynamic connections to be created. These objects are anchor filter rules. An anchor filter is just a filter rule with a special action. That action is anchor.

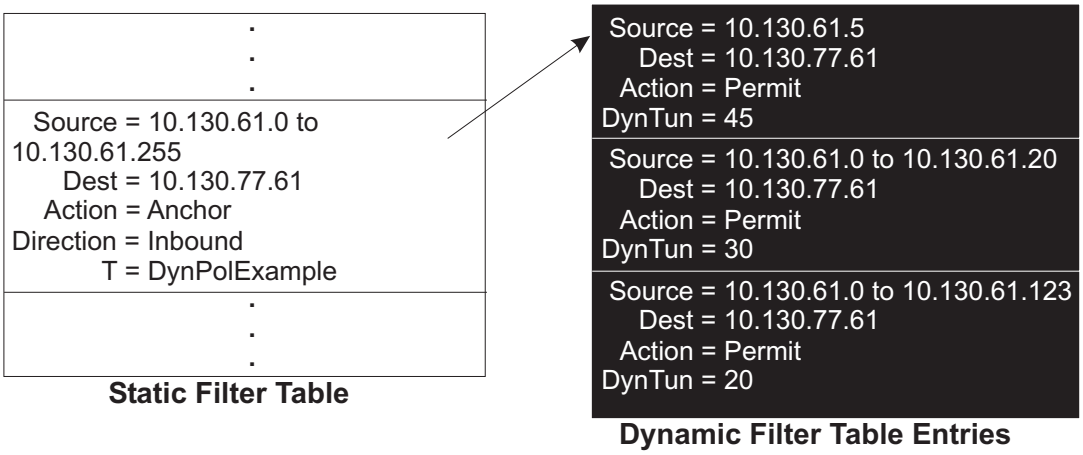
Filter rules are ordered. An anchor filter rule serves as a place holder in the ordered list of filter rules for another type of filter rule called a dynamic filter rule. An anchor filter defines a range of selector values that can be used to create a dynamic filter. It also defines a dynamic tunnel policy that must be adhered to by all dynamic filter rules.

If an IP packet matches the selector value on an anchor filter rule a matching dynamic filter rule will be searched for. If one is not found the packet is denied. If one is found the action taken is permit with IPSec processing applied. The dynamic filter rule will indicate what dynamic tunnel should be used when applying IPSec processing.

Dynamic filters are an extension to an anchor filter. They are associated with a dynamic tunnel. Dynamic tunnels are created as a result of a dynamic connection activation (either remotely initiated or locally initiated). There is a 1-to-1 correspondence between dynamic filter pairs (inbound and outbound), dynamic connections, and dynamic tunnels.

The selector values of a dynamic filter rule must fit within the range of selector values for an anchor filter. This means the dynamic filter must have the same selector values as the anchor rule or they must specify a range that is a proper subset of the anchor filter. Two dynamic filter rules may overlap. In this case, one must be a proper subset of the other. When a dynamic filter overlaps another dynamic filter, the more specific rule is applied first.

Figure 33 on page 96 shows the relationship between anchor filter rules and dynamic filter rules.



Assume that an inbound packet is destined for 10.130.77.61
Packets from 10.130.61.5 will use dynamic tunnel 45
Packets from 10.130.61.0 to 10.130.61.4 will use dynamic tunnel 30
Packets from 10.130.61.6 to 10.130.61.20 will use dynamic tunnel 30
Packets from 10.130.61.21 to 10.130.61.123 will use dynamic tunnel 20
Packets from 10.130.61.124 to 10.130.61.255 will be denied

Figure 33. Relationship Between Anchor Filter Rules and Dynamic Filter Rules

Anchor Filter Creation: Anchor filter rules are just another static filter rule with the exception of a few parameters. They are created the same way a normal filter rule is created. The steps to creating an anchor filter are listed:

- First a generic filter Rule with action of anchor is created. This Rule is generic because not all selector values are specified on it (e.g. source and destination addresses).
- Next, the generic filter rules are grouped into Services. Services can be used to define a logical grouping of rules for a service such as FTP.
- The services are combined with two Network Objects to create Connections. The Network Objects provide the missing selector values.
- Finally, the Connections are activated using the **fwfilter** command (or the corresponding configuration client panel) which causes an expansion of all the generic Rules and Network Objects to create the static filter rules. VPNs require two anchor rules, one for each direction that traffic flows. One **fwfrule** with **type=anchor** results in a pair of anchor filter rules being generated when **fwfilter cmd=update** is issued.

  This pair of rules contain one inbound and one outbound rule. The outbound rule is created with the source and destination IP values specified in the service rule list. Forward flows means that the filter rule source is the packet's source and the filter rule destination is the packet's destination. Backward flows means that the filter rule destination is the packet's source and the filter rule source is the packet's destination.

Filter Rules, Services, Network Objects and Connections are defined using the **fwfrule**, **fwservice**, **fwnwobj** and **fwconns** commands (or the corresponding configuration client panel). Figure 34 on page 97 shows the relationship between Dynamic Tunnel Policy, Filter Rules, Services, Network Objects and Connections.

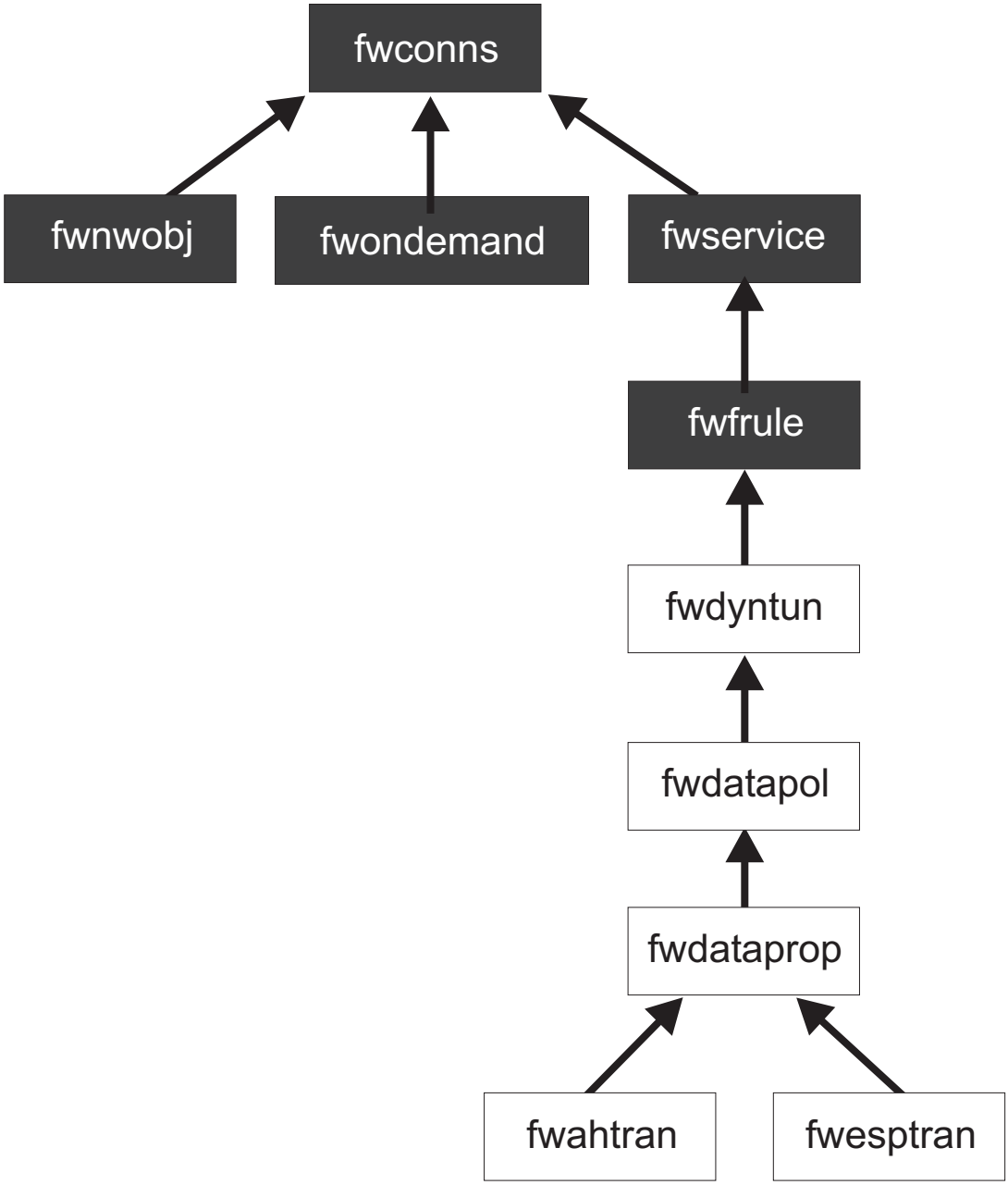


Figure 34. Dynamic Tunnel Policy, Filter Rules, Services Network Objects and Connections

The following example illustrates how a single anchor rule generates two filter rules.

1. DynamicTunnel10 represents a dynamic tunnel policy.
2. NetworkObject1 represents a host network object with the IP address 10.1.1.1 mask 255.255.255.255
3. NetworkObject2 represents a host network object with the IP address 10.2.2.2 mask 255.255.255.255
4. Example_Anchor represents a filter rule whose id is 501 and was defined with the following command.

```
fwfrule cmd=add name=Example_Anchor desc="Example anchor rule"
          type=anchor protocol=tcp srcopcode=gt port=1023
          destopcode=eq destport=21 interface=secure
          routing=both log=deny tunnel=DynamicTunnel10
```

5. Example_Service represents a service rule whose id is 502 and was

```
fwservice cmd=create name=example_service desc="Example Service"
            rulelist=501/f
```

6. Example_connection represents a connection that was defined using the following command.

```
fwconns cmd=create name=example_connection desc="Example Connection"
          source=NetworkObject1 destination=NetworkObject2
          servicelist=502
```

After **fwfilter cmd=update** is issued, the following two rules are generated as a result of the above connection.

```
permit 10.1.1.1 255.255.255.255 10.2.2.2 255.255.255.255 tcp gt 1023
                                eq 21 secure both outbound l=d ...
permit 10.2.2.2 255.255.255.255 10.1.1.1 255.255.255.255 tcp eq 21
                                gt 1023 secure both inbound l=d ...
```

Local Activation Information

Once the appropriate anchor filter has been generated and the appropriate key server group has been defined, dynamic connections can be activated. If the dynamic connection is to be created as a result of responder mode, no further configuration is required. If the dynamic connection is to be activated locally, additional configuration steps are required.

If the dynamic connection is to be auto-started or manually started, then a local dynamic connection must be defined. If the dynamic connection is to be started by on-demand (i.e. by the existence of an outbound IP packet), then an on-demand object must be defined and referenced by one or more connections.

Dynamic Connections: Locally defined dynamic connections are not consulted when dynamic connections are created as a result of responder mode or an on-demand activation. There are two basic pieces of information contained within a dynamic connection. The first piece is the identity of the data endpoints. The second is the identity of the remote dynamic tunnel endpoint (i.e. the location of the ISAKMP peer).

Dynamic Connections are defined using the **fwdynconns** command (or the corresponding configuration client panel). Figure 35 shows the relationship between Dynamic Connections, Network Objects, Key Servers and Key Server Groups.

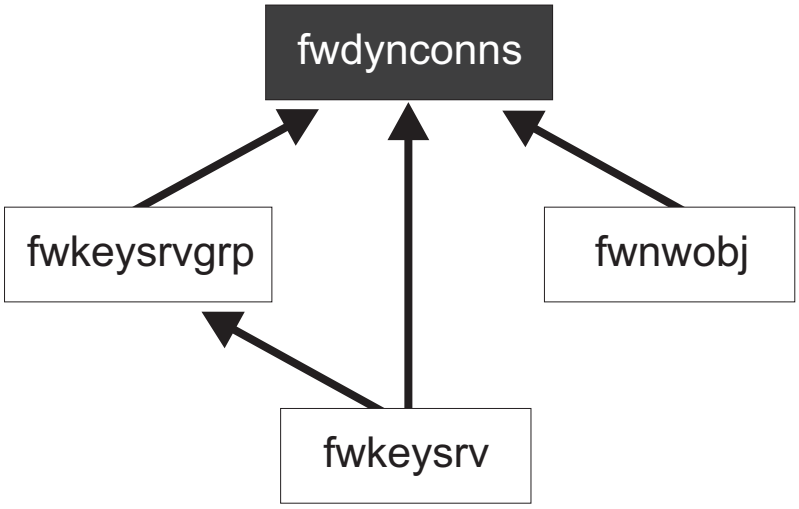


Figure 35. Dynamic Connections, Network Objects, Key Servers and Key Server Groups

Locally defined dynamic connections can be automatically activated. Auto activation can occur at two times. The first time is when the TCP/IP stack associated with the local key server of a dynamic connection comes up. The second time is when the ISAKMP server comes up and the TCP/IP stack associated with a dynamic connection is already up.

When manually initiating a dynamic VPN, the information required to negotiated key policy is contained in the Dynamic Connection definition. The Remote Key Server field contained in the dynamic connection definition identifies the dynamic tunnel partner. The Key Server Group field contained in the dynamic connection definition identifies the key policy to be used. When a dynamic connection is locally activated, the dynamic tunnel endpoints negotiate how to protect key exchanges using the defined key policy (Note that this is only done if a valid negotiated key policy with this dynamic tunnel partner does not already exist).

The source Network Object, destination Network Object, source port, destination port, and protocol values of the dynamic connection definition are used to find a matching anchor filter rule. A matching anchor filter rule is one where the dynamic connection has the same selector values as the anchor rule or the dynamic connection selector values specify a range that is a proper subset of the anchor filter. If a matching anchor filter is found, then it will identify the dynamic tunnel policy. The dynamic tunnel endpoints will then negotiate security association attributes and key information based on the data policy associated with the dynamic tunnel. The result of a successful negotiation is a dynamic tunnel. Once a dynamic tunnel is created, a dynamic filter rule using this dynamic tunnel is associated with the anchor filter.

On-demand Connections: On-demand objects are consulted when all of the following conditions are met:

- An outbound IP packet matches an anchor filter rule
- No matching dynamic filter rules exist
- The connection object that resulted in the generation of the anchor filter contains a reference to an on-demand object.

The on-demand object contains information indicating how the source and destination IP specification of an on-demand dynamic connection should be derived. Either IP specification could be derived in one of two ways. The first way is by using the corresponding IP specification contained in the matching anchor filter rule. The second way is by using the corresponding IP specification contained in the IP packet that matched the anchor filter rules.

The protocol information for an on-demand dynamic connection is derived from the matching anchor rule. The source and destination port information for an on-demand dynamic connection is also derived from the matching anchor rule, except in the case where a relational port (i.e. LT 1080) is used. In this case, the port will be taken from the packet that matched the anchor rule.

The on-demand object also contains information identifying the dynamic tunnel partner and the key server group that the dynamic tunnel partner belongs to.

On-demand objects are defined using the **fwondemand** command (or the corresponding configuration client panel) and are specified on Connection objects.

Figure 34 on page 97 shows the relationship between connection objects and on-demand objects.

Summary

Figure 36 on page 100 shows the relationship between all of the objects discussed throughout this chapter.

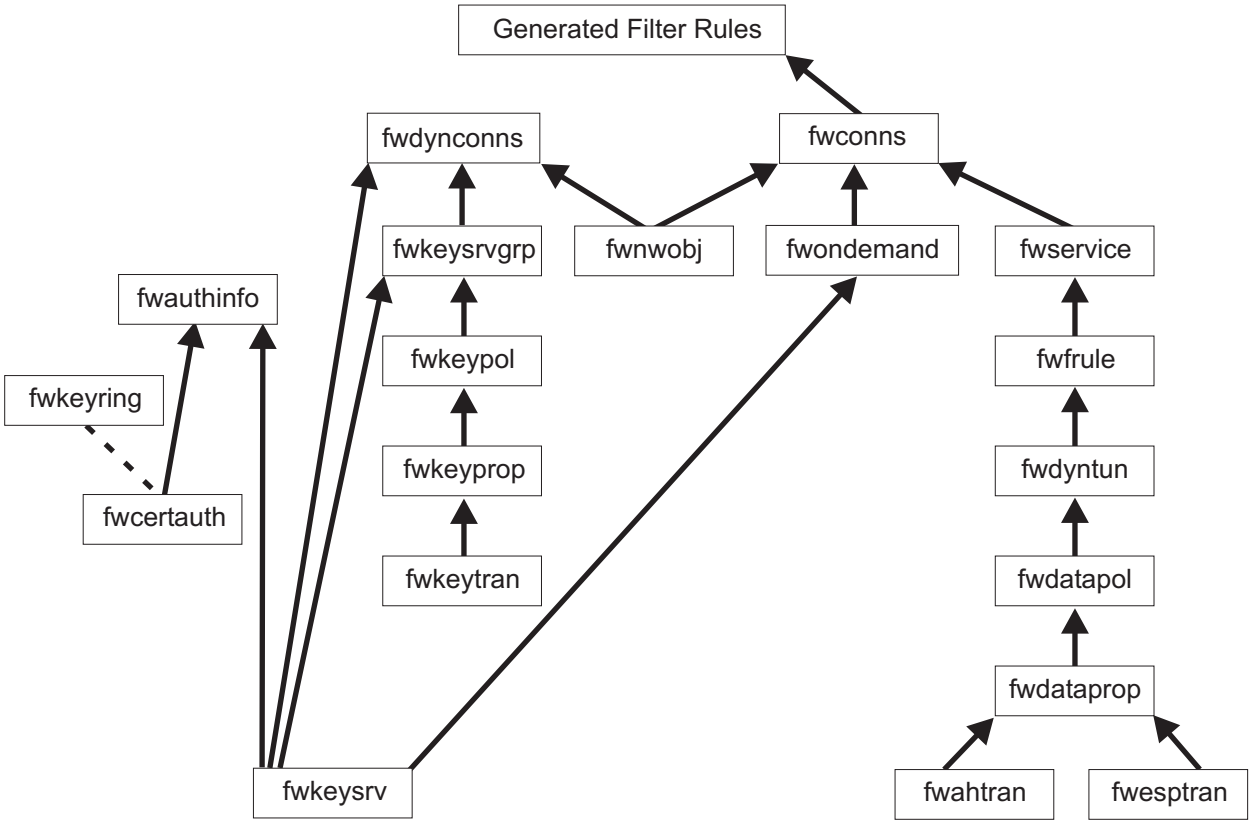


Figure 36. Dynamic VPN Scheme

Miscellaneous

Pre-Decapsulation Filtering Option: A new option, pre-decapsulation filtering, is now available when regenerating filter rules via the **fwfilter** command (or the corresponding configuration client panel). If this option is enabled then pre-decapsulation filtering occurs. This means that before attempting to decapsulate an AH or ESP packet, the filter rules will be checked to see if the AH/ESP packet is allowed to flow between the local host and the sender. The result is that AH and ESP filter rules must be specified for each host that the local host maintains an IPSec tunnel with.

By disabling the pre-decap filter option an attempt will be made to decapsulate any AH/ESP traffic without consulting the filter rules first. The result is that AH and ESP filter rules do not need to be specified for each host that the local host maintains an IPSEC tunnel with. The pre-decapsulation filtering option has no bearing on the filtering that occurs after decapsulation. The filter rules are always consulted after a packet has been successfully decapsulated to see if the packet should be allowed to flow through the local system.

ESP, AH and HDP Port 500 Filter Rules: ESP, AH, and UDP port 500 filter rules must be manually defined. Since ISAKMP communications occur on UDP port 500 filter rules which permit communication with ISAKMP peers must be defined. If pre-decapsulation filtering is enabled, then ESP and AH filter rules must also be defined to permit IPSec traffic to flow between ISAKMP peers. AH and ESP are also required if IPSec traffic is allowed to flow through to a local host.

ISAKMP Main Mode Limitations

Main Mode Scenario 1: Key policy definition is based on the identities of remote key servers. Unfortunately, during a phase 1 main mode exchange the responding ISAKMP server must accept a key proposal prior to learning the identity of the initiating ISAKMP server. The responder must later verify that the proposal agreed to is acceptable with defined policy when the identity becomes known.

In order to help pick a proposal the OS/390 ISAKMP server will consult a default key policy definition. This default policy definition is shipped with the product, but may be modified to be consistent with local key policy. For example, if local key policy always requires encryption to be triple DES, the default key policy file should not contain any DES transforms. The default key policy as shipped accepts any transform.

Main Mode Scenario 2: Pre-shared keys are defined based on the identities of remote key servers. Unfortunately, during a phase 1 main mode exchange the responding ISAKMP server must use the pre-shared key prior to learning the identity of the initiating ISAKMP server.

In the case of OS/390 acting as a responder in main mode using pre-shared key authentication, the IP address of the initiator is used to find the key server entry that identifies the initiator. If no match is found a DNS lookup is performed to determine the host name associated with the initiator's IP address and the host name is used to find the key server that identifies the initiator.

Main Mode Scenario 3: CA certificates are associated with the identities of remote key servers. CA certificates are used to help guide the ISAKMP peer in picking a certificate to use when main mode using RSA signature authentication is being performed. Unfortunately, during phase 1 main mode exchange the responding ISAKMP server must send the CA certificate prior to learning the identity of the initiating ISAKMP server.

In the case of OS/390 acting as a responder in main mode using RSA signature authentication, the IP address of the initiator is used to find the key server entry that identifies the initiator. If no match is found, a DNS lookup is performed to determine the host name associated with the initiator's IP address and the host name is used to find the key server that identifies the initiator. If no match is found, then no CA certificate will be sent to the peer.

Chapter 15. Network Address Translation

With the growth of the Internet, the pool of available IP addresses is being depleted. Network address translation (NAT) provides a solution to this problem by translating secure IP addresses to temporary external registered addresses in order to provide sufficient connections. NAT occurs before TCP/IP routing and filter rule checking for inbound data and after TCP/IP routing and filter rule checking for outbound data. Therefore, the actual secure network addresses should be in your filter rules, and no TCP/IP profile changes are required for the NAT addresses, however, routing information may need updated accordingly.

IBM provides network address translation for any routed TCP or UDP application packets. NAT is not used for packets created by proxy servers, for example FTP, that reside on the firewall because they use the firewall address itself as the source address. In addition, FTP is supported by providing translation of the IP address in the **port** command.

Important Note

NAT is not applied to packets that are neither TCP nor UDP; for example, any routed ICMP packets that filters permit to pass will not have their source address translated.

NAT Configuration File

RESERVE, **TRANSLATE**, and **EXCLUDE** are used to configure dynamic clients. **MAP** is used to define specific one-to-one mappings between IP addresses. The NAT configuration can contain up to 512 entries of the types listed below.

- RESERVE Registered Addresses

  A RESERVE registered address entry defines a set of registered IP addresses that can be used for outbound (that is, going from the secure to the non-secure network) connections. When a secure host sends a packet to a non-secure network, a registered IP address is allocated from the reserved registered address pool. This unique registered IP address is substituted for the source address of the outbound packet. This mapping prevents secure addresses from being known to the world and allows a network behind a firewall to have as many addresses as it needs.

  Note: Although the subnet mask field of a RESERVE registered address entry has the same properties as a normal subnet mask, NAT restricts the valid values that the subnet mask may have. To determine the number of addresses being reserved by the entry, the subnet mask can only have byte values of 255, 254, 252, 248, 240, 224, 192, 128, and 0. In addition, if one of the bytes has a value not equal to 255, then all bytes following it must be zero.

  Examples:

  ```
  RESERVE ADDR       MASK
  RESERVE 1.2.3.0 255.255.255.255
  ```

  defines an address pool containing one address, 1.2.3.0.

  ```
  RESERVE ADDR       MASK
  RESERVE 1.2.3.1 255.255.255.252
  ```

  defines an address pool containing 4 addresses, 1.2.3.0, 1.2.3.1, 1.2.3.2, and 1.2.3.3.

  ```
  RESERVE ADDR       MASK
  RESERVE 1.2.3.0 255.255.0.0
  ```

  defines an address pool containing 65536 addresses, all addresses in the form 1.2.n.n.

© Copyright IBM Corp. 1997, 2000

```
RESERVE ADDR       MASK
RESERVE 1.2.3.0 255.0.255.255
```

is not a valid definition.

Once a RESERVE registered address is allocated, the connection can remain idle for a given time period before that address is returned to the available address pool. The **fwstack** daemon monitors idle connections and the return of addresses to the available pool.

- EXCLUDE Secured IP Addresses

  An EXCLUDE secured IP address entry defines a set of secure network addresses that does not require NAT to perform IP address translation. The IP packet passes through the firewall unmodified. Subnet masks are used for the EXCLUDE mask.

  Note: This requires the IP packet to have a registered IP source address in order for the nonsecure host to reply correctly.

  Example:

  ```
  EXCLUDE 128.1.2.0 255.255.255.0
  ```

- TRANSLATE Secured IP Addresses

  A TRANSLATE secured IP address entry defines a set of secure network addresses that require NAT to perform IP address translation. These translations are taken from the RESERVE registered address pool. If none are available, the IP packet is discarded. Subnet masks are used for the TRANSLATE mask.

  Example:

  ```
  TRANSLATE 126.1.2.0 255.255.255.0
  ```

- MAP Secured IP Address

  A MAP secured IP address entry defines a one-to-one mapping from a secure IP address to a registered IP address. This one-to-one IP address mapping allows external application clients, such as FTP or Telnet clients, to set up TCP or UDP sessions with server machines that reside within the secure network. The registered IP addresses in the map secure IP address entries can overlap the IP address space specified by the RESERVE registered IP address entries.

  Example:

  ```
  MAP ADDR          ADDR
  MAP 126.1.2.6 195.9.5.6
  ```

NAT Configuration File Entry Interactions

The following table describes how the firewall's NAT function affects packets that the filter function permits to pass through the firewall. Interactions between the different types of NAT configuration entries are described.

Figure 37 (Page 1 of 3). NAT Configuration File Entry Interactions

Active Entries	Firewall's Actions
None	When there are no active NAT configuration file entries, NAT is not active and no secure addresses are translated.
RESERVE only	All secure source addresses are translated.
TRANSLATE only	Secure source addresses listed in a TRANSLATE entry are discarded; packets with other secure source addresses are allowed through without translation. (The TRANSLATE entry secure source address packets are discarded because there is no RESERVE entry specifying available external addresses.)

Figure 37 (Page 2 of 3). NAT Configuration File Entry Interactions

Active Entries	Firewall's Actions
EXCLUDE only	Secure source addresses listed in an EXCLUDE entry are allowed through without translation; packets with other secure source addresses are discarded.
MAP only	Secure source addresses listed in a MAP entry are translated to the corresponding addresses and allowed through; packets with other secure source addresses are discarded.
RESERVE and TRANSLATE	Secure source addresses listed in a TRANSLATE entry are translated to an address from the RESERVE pool and allowed through; packets with other secure source addresses are allowed through without translation.
RESERVE and EXCLUDE	Secure source addresses listed in an EXCLUDE entry are allowed through without translation; other secure source addresses are translated to an address from the RESERVE pool and the packets are allowed through.
MAP and RESERVE	Secure source addresses listed in a MAP entry are translated to the corresponding addresses; packets with other secure source addresses are translated to addresses from the RESERVE pool and allowed through.
EXCLUDE and TRANSLATE	Secure source addresses listed in an EXCLUDE entry are allowed through without translation; secure source addresses listed in a TRANSLATE entry are discarded; packets with other secure source addresses are allowed through without translation even though they do not match the addresses listed in an EXCLUDE entry. (The TRANSLATE entry secure source address packets are halted because there is no RESERVE entry specifying available external addresses.)
MAP and EXCLUDE	Secure source addresses listed in a MAP entry are translated to the corresponding addresses; secure source addresses listed in an EXCLUDE entry are allowed through without translation; packets with other secure source addresses are discarded.
MAP and TRANSLATE	Secure source addresses listed in a MAP entry are translated to the corresponding addresses; secure source addresses listed in a TRANSLATE entry are discarded; packets with other secure source addresses are allowed through without translation. (The TRANSLATE entry secure source address packets are discarded because there is no RESERVE entry specifying available external addresses.)
RESERVE, TRANSLATE, and EXCLUDE	Secure source address listed in a TRANSLATE entry are translated to an address from the reserve pool and allowed through; packets with other secure source addresses are allowed through without translation even if they do not match the addresses listed in an EXCLUDE entry.
RESERVE, TRANSLATE, and MAP	Secure source addresses listed in a MAP entry are translated to the corresponding addresses; secure source addresses listed in a TRANSLATE entry are translated to an address from the RESERVE pool; packets with other secure source addresses are allowed through without translation.
EXCLUDE, TRANSLATE, and MAP	Secure source addresses listed in a MAP entry are translated to the corresponding addresses; secure source addresses listed in an EXCLUDE entry are allowed through without translation; secure source addresses listed in a TRANSLATE entry are discarded; packets with other secure source addresses are allowed through without translation even though they do not match the addresses listed in an EXCLUDE entry. (The TRANSLATE entry secure source address packets are discarded because there is no RESERVE entry specifying available external addresses.)
RESERVE, EXCLUDE, and MAP	Secure source addresses listed in a MAP entry are translated to the corresponding addresses; secure source addresses listed in an EXCLUDE entry are allowed through without translation; packets with other secure source addresses are translated to addresses from the RESERVE pool.

Figure 37 (Page 3 of 3). NAT Configuration File Entry Interactions

Active Entries	Firewall's Actions
RESERVE, TRANSLATE, EXCLUDE, and MAP	Secure source addresses listed in a MAP entry are translated to the corresponding addresses; secure source addresses listed in a TRANSLATE entry are translated to addresses from the RESERVE pool; packets with other secure source addresses are allowed through without translation even if they do not match the addresses listed in an EXCLUDE entry.

Note: When the same secure network address is listed in more than one entry the precedence order is:

1. MAP,
2. EXCLUDE,
3. TRANSLATE.

It is important to remember that once a RESERVE address pool is activated, address translation will occur for all eligible packets. A set of translated addresses does not have to be activated. If a TRANSLATE entry is activated, only that range of addresses is translated, any other address is implicitly excluded.

Note: NAT configuration must not conflict between different firewall stacks. NAT requires a stack parameter for multiple stacks. An error message displays if the NAT configuration conflicts between stacks.

NAT Logging

The NAT facility maintains a log of the last 1024 NAT mapping entries. The log is kept in storage and can be started using the **fwnat cmd=startlog** command. To capture all possible log entries, NAT logging:

```
fwnat cmd=startlog stack=stackname
```

must be started before NAT initialization. The log entry looks like the following:

```
Time(%1$s %2$s) Secured: %3$s Registered: %4$s Static(%5$d)
```

where:

%1$s Date of log entry

%2$s Time of log entry

%3$s Secure address

%4$s Registered address

%5$d Static:

1 value indicates the entry is static (MAP) - logged on rules updates
0 value indicates the entry is dynamic (RESERVE/TRANSLATE) - logged when allocated from address pool

The log is not permanent and is cleared when logging is restarted. Also, because NAT logging stops when NAT facility shutdowns, the log entries should be staged to a file, if necessary, to keep a permanent audit record. For more information, see “fwnat” on page 279 and “Configuring Network Address Translation” on page 169.

Chapter 16. Supporting the RealAudio Protocol

RealAudio is a special protocol developed by Progressive Networks, which supports live and on-demand audio from the Internet.

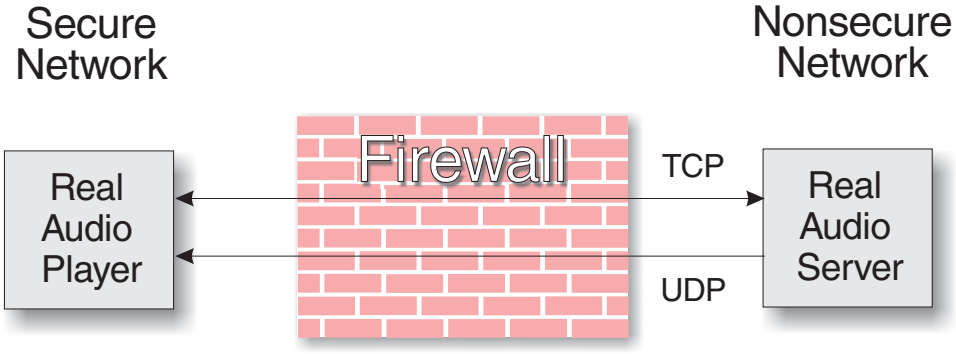


Figure 38. RealAudio connections through the Firewall

When a RealAudio TCP connection is detected, the back channel UDP packet from the RealAudio server to the RealAudio player is permitted to pass through the firewall as long as the TCP connection is active.

How Firewall Supports RealAudio

In the recommended configuration, the protocol requires two connections. The first connection is a TCP connection from the RealAudio player to the RealAudio server. After this connection is established, the RealAudio server optionally establishes a UDP channel back to the player. If the RealAudio server is TCP only, no further action is required by the firewall. In the scenario where UDP is used, the UDP connection is dynamic in the sense that the destination port number is dynamic.

OS/390 Firewall Technologies supports the RealAudio protocol by monitoring and identifying these RealAudio TCP connections. When a connection is identified, a dynamic filter rule for a UDP packet is defined. This filter rule is removed once the RealAudio TCP connection is closed. This is transparent to the RealAudio user. No extra configuration or knowledge is needed. When a RealAudio TCP connection is detected, the back channel UDP packet from the RealAudio server to the RealAudio player will be permitted to pass through the firewall as long as the TCP connection is active.

Configuring RealAudio

Use the command “fwaudio” on page 187 and the following steps or the configuration GUI “Configuring RealAudio” on page 173 to configure RealAudio:

The RealAudio configuration provides two options:

1. Setting the port number for the RealAudio server. The RealAudio default server port number is 7070, however, you can reconfigure it to any valid TCP port number.
2. Limiting the number of concurrent RealAudio sessions enabled by the Firewall. The default is 10. It can be any non-negative integer.

© Copyright IBM Corp. 1997, 2000

RealAudio Web Site

You can find out more about RealAudio at *http://www.real.com*.

Part 3. Configuration Client Graphical User Interface (GUI)

© Copyright IBM Corp. 1997, 2000

Chapter 17. Using the Configuration Client Graphical User Interface (GUI)

Defining a Security Policy

One of the first things to consider when configuring the OS/390 Firewall Technologies is the general security policy for your installation.

The OS/390 Firewall Technologies provides a dialog box to assist you in setting up your security policy, as shown in Figure 39.

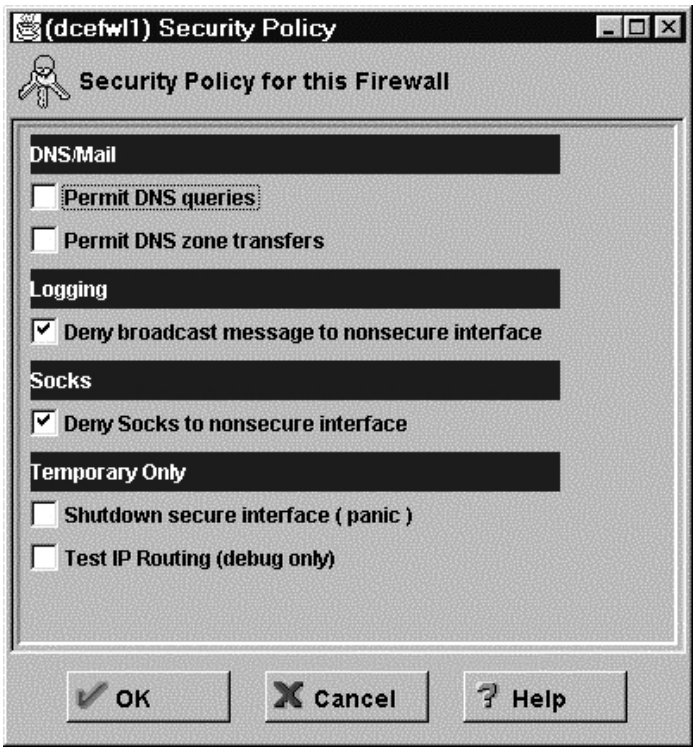


Figure 39. Security Policy

Click Help to learn more about the security policy panel.

The Security Policy provides a quick and easy way for administrators to set blanket policies for the firewall. Most of the check boxes displayed in the security policy window provide a fast path to selecting certain Predefined Services that will apply to all network traffic received by the firewall.

When you select a security policy, the firewall builds the filter rules, which you then need to activate. The firewall enables the services selected and makes them globally available.

Note that any time you select a check box that pertains to a Predefined Service and you click OK, you must activate these changes through the Connection Activation window. See “Predefined Services” on page 53 for a list of predefined services.

You are presented with the following list of check boxes from which you can select attributes that reflect the security policy for your site. The attributes selected apply to all addresses on both sides of the OS/390 Firewall Technologies.

© Copyright IBM Corp. 1997, 2000

- Select Permit DNS Queries to allow Domain Name Service resolution requests and replies.
- Select Permit Zone Transfers to allow Domain Name Service data files to be transferred from name server to name server.
- Select Deny broadcast message to nonsecure interfaces to prevent broadcast messages from being received at the nonsecure port. If your firewall's nonsecure interface is connected to the Internet, this service can help reduce the amount of logging on the OS/390 Firewall Technologies.
- Select Deny Socks to nonsecure interface to disallow socks traffic to enter the firewall from the nonsecure network.
- Select Shutdown secure interface (panic) to disallow all traffic to and from the firewall over the secure interfaces. This is used for emergency purposes only.
- Select Test IP Routing (debug only) to allow all traffic to and from firewall over any interface. Note that if you change the value of this check box, you must save it by clicking OK, and activate it through the Connection Activation window. **Use of this Service can cause security exposures for your firewall. Use it with extreme caution.**

Network Objects

Network objects represent components in your network such as hosts, networks, routers, or virtual private networks. Network objects designate source and destination addresses for services when you create your connections.

Defining Network Objects

Network objects define source and destination addresses for services when you create your connections.

To define a single network object, select Network Objects from the configuration client navigation tree. The Network Objects dialog box appears. Double click NEW. The Add a Network Object dialog box appears, as shown in Figure 40 on page 113.

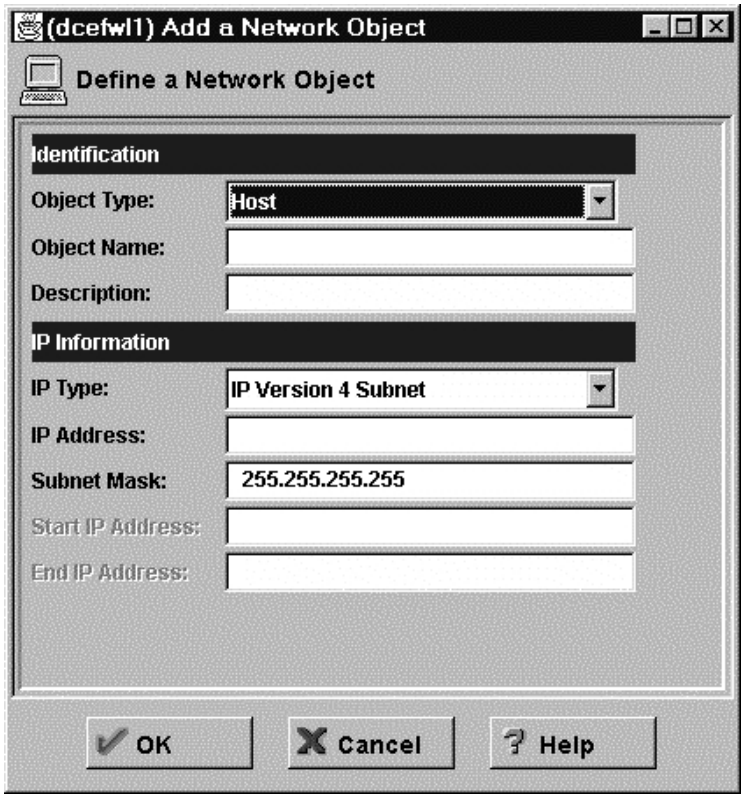


Figure 40. Add a Network Object

1. Enter the object type. Click the Object Type arrow to see the object types you can create. For performance reasons, it is better to create network type objects instead of host type objects. The object types you can create are:
   - Host - a particular node on your network with a mask of 255.255.255.255.
   - Network - a collective range of network addresses that is characterized by an address range and a specific subnet mask.
   - Firewall - a single machine with a firewall installed on it with a mask of 255.255.255.255.
   - Router - a host that routes traffic between two or more networks with a mask of 255.255.255.255.
   - Interface - a network adapter on a machine with a mask of 255.255.255.255. It does not have to be an adapter on the Firewall.
   - VPN - a Virtual Private network on the other side of a tunnel.
2. Fill in the object name.
3. Fill in the description. This field is optional.
4. Enter a dotted-decimal IP address for this object.
5. Enter a subnet mask that specifies the bits in the address to compare to the address of the IP packet.
6. Click OK

Network Object Groups

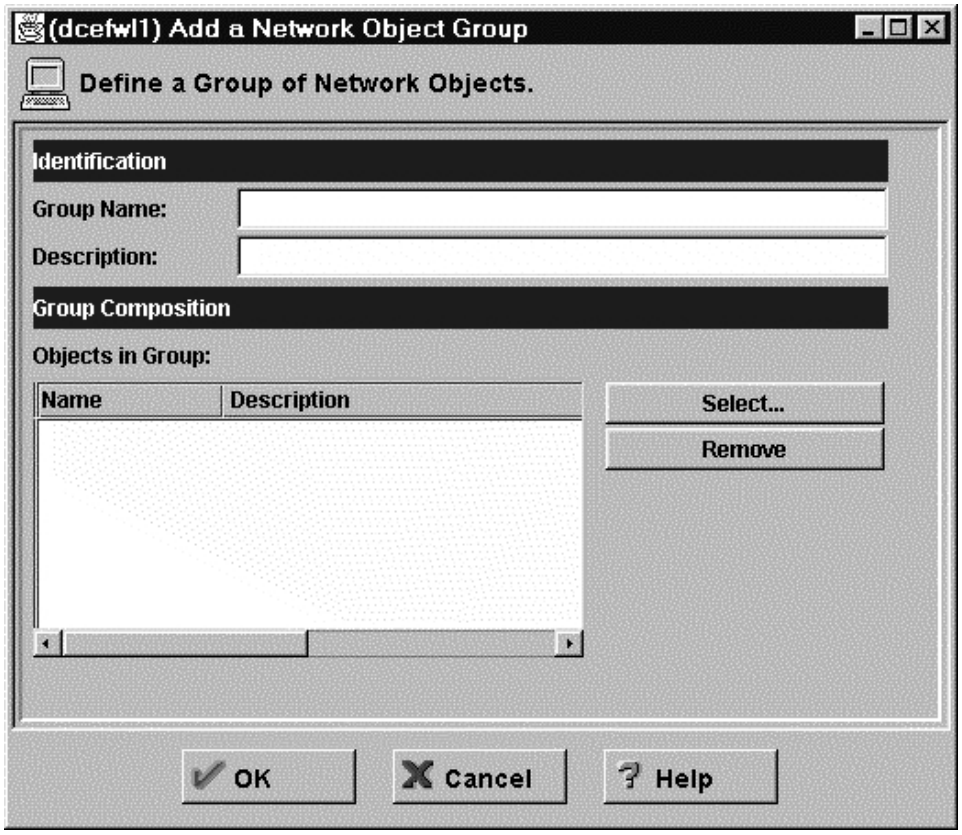


Figure 41. Add a Network Object Group

To define a group of network objects, select Network Objects from the configuration client navigation tree. The Network Objects dialog box appears. Double click NEW GROUP. The Add a Network Object dialog box appears.

1. Fill in the group name.
2. Fill in a description. This field is optional.
3. Click Select to select objects for the group.
4. Click OK.

Determining the Rule States

The OS/390 Firewall Technologies rules can be in one of these states:

1. The configuration is not active.

   You have not yet used the configuration client to activate the configuration or you have deactivated the configuration. This is the state of the configuration when you first install the OS/390 Firewall Technologies and IPL your system or deactivate filter rules. Default filters are in place to protect your network from intrusion when you first install the Firewall.

   Firewall Access:

   - The default filter configuration permits all local traffic.

2. The configuration is active but has errors.

   You have activated the configuration. Either there are errors (non-valid rules) in the configuration or nothing was configured. Errors and warnings are displayed in the Activation output window.

   Firewall Access:

- Permit all local traffic.

3. The configuration is active and valid. Note that there may have been some warnings, most notably, duplicate filter rules.

   You have activated the configuration that you defined using the traffic control section of the configuration client.

   Note: The configuration file can be valid and still contain no rules. In this case, an implied “deny all access” rule is in effect.

   Firewall Access:

   - Access determined by the configuration file.

     Each packet that is received by, or is about to be sent by, any network interface is examined and its contents compared against each rule in the generated connection rules. When a match is found, the action (permit or deny access) on that rule is carried out.
   - If no rules match the packet, there is an implied “deny all” rule that denies access.

Creating Filter Rules

Use this procedure to add a new rule to the list of available rule templates.

1. From the configuration client navigation tree, select Traffic Control and double-click the file folder icon. Select Connection Templates and then select Rules. On the Rules List dialog box, double-click NEW.

   The firewall displays an Add IP Rule dialog box, as shown in Figure 42 on page 116 so that you can define a rule.

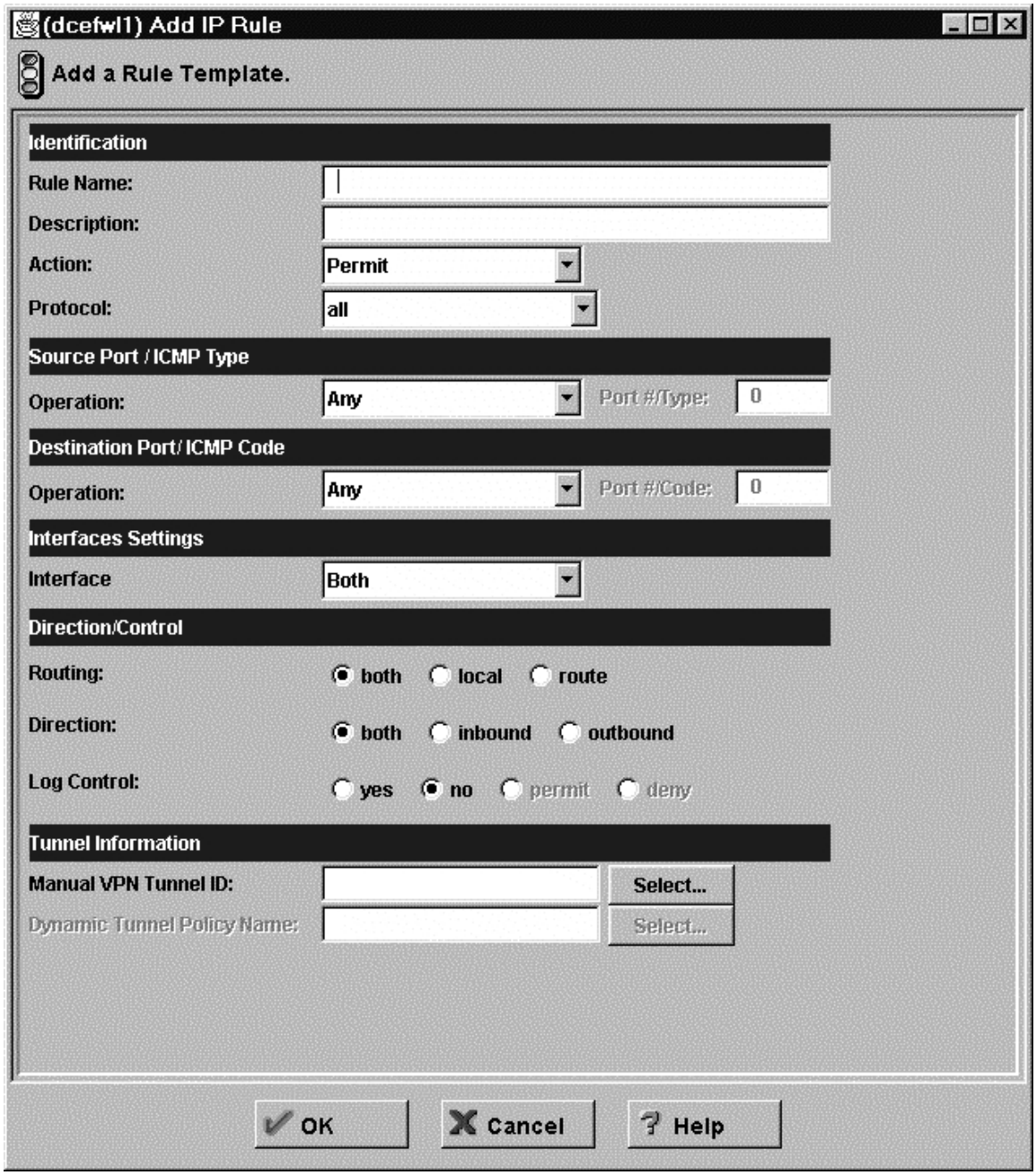


Figure 42. Add IP Rule

2. Enter the Rule Name.
3. Enter the Rule Description. This field is optional.
4. Click the action arrow and choose to either permit or deny access to the Firewall, or anchor rules to serve as a place holder for dynamic filter rules.
5. Click the protocol arrow and select from the following list:

   all Any protocol will match this rule.

   tcp The packet protocol must be transmission control protocol (TCP) to match this rule.

   tcp/ack The packet protocol must be TCP with acknowledgement to match this rule. (Not applicable for anchor rules.)

udp The packet protocol must be user packet protocol (UDP) to match this rule.

icmp The packet protocol must be internet control message protocol (ICMP) to match this rule.

ospf The packet protocol must be open shortest path first protocol (ospf) to match this rule. When **ospf** is specified as the protocol, the source port operation and source port value is used for the **ospf** record type value. Filtering can also be performed on the ospf type. A type value of Any can be specified and the destination port fields must be specified as Any 0. Anything else is ignored.

ipip The packet protocol must be IP-in-IP protocol (IPIP) to match this rule. When IPIP is specified, the port fields must be specified as Any 0.

esp The packet protocol must be encapsulating security protocol used by the virtual private network for sending encapsulated IP packets to match this rule.

ah Authentication header protocol is the packet protocol used by the virtual private network for sending IP packets which have an associated authentication token.

6. The operation and port number operands are used together. The source and operations state a relationship between the port number (destination or origin) for the packet and the source port# and destination port# operands. For example, if the packet destination port is port 20, and the destination operation and destination port# are “ge 15,” the packet matches. (20 is greater than or equal to 15).

   If you use a source or destination operation of Any, the filter does not look at the port number; any port will match. The port number cannot be changed in this case.

   For the ICMP protocol, rather than specifying a source port, specify an ICMP type and in place of a destination port, specify an ICMP code. The operation specified is applied to the type or code and, as for ports, an operator of *any* means that any type and code value will match the rule. The port number cannot be changed in this case.

   The values for operation are:

   - Any
   - Equal to (=)
   - Not equal to (≠)
   - Less than (<)
   - Greater than (>)
   - Less than or equal to (≤)
   - Greater than or equal to (≥)

   Here are some of the more important ports to protect. The values for port numbers must be in the range 1 through 65535:

Port	Use
20	FTP data
21	FTP control
23	Telnet
25	Mail
53	Domain Name Server
70	Gopher
80	HTTP
111	RPC
161	SNMP

1014 Configuration Server

1080 socks

Here are some of the ICMP types and codes:

Type	Code and Description
0	0 - Ping reply
8	0 - Ping request
3	1 - Host unreachable
3	3 - Port unreachable
5	1 - Redirect for host

7. Click the Interface arrow to select the type of interface (adapter).

   both For packets coming or going on either the secure or the nonsecure interface

   secure For packets coming or going on the secure interface

   nonsecure
   For packets coming or going on the nonsecure interface

8. Click the routing:

   both Applies to all traffic.

   local Implies that the packet is local to the firewall host. This means that:

   - Incoming local packets are packets that are received by the interface and are destined for this firewall host; they will not be routed to another host. Their destination is local.
   - Outgoing packets are transmitted from the interface, but originate on the firewall host. Their origin is local.

   route Implies that the packet is routed by the firewall host. This means that:

   - Incoming routed packets are packets that are received by the interface and are destined for some other host; they will not remain on the Firewall. Their destination is remote.
   - Outgoing packets are transmitted from the interface, and originated on some other host. Their origin is remote.

9. Click the direction: (not applicable to anchor rules)

   both For packets going out from or into the selected interface

   inbound For packets coming into the selected interface from the network

   outbound
   For packets going out from the selected interface to the network

10. Click the log control:

    - Yes - every packet that matches that rule is recorded in the **local4** log with priority level *Error*.
    - No - do not record a log record when the packet matches this rule.

    Anchor rules (only):

    - Permit - write a log record when this anchor rule has been matched and an associated dynamic filter rule has also been matched.
    - Deny - write a log record when this anchor rule has been matched and no associated dynamic filter rule has been matched.

11. When defining an anchor rule, the Dynamic Tunnel Policy Name must be specified. Click Select and choose a Dynamic Tunnel Policy object from the Dynamic Tunnel Policy object dialog list.

For a packet header to match a defined IP rule, the packet information must match all the parameters specified in the coded rule.

Note: The configuration client for OS/390 Firewall Technologies displays options for fragmentation, however, fragment options are ignored and not needed. All packet fragments will be reassembled before any filtering is performed.

Changing IP Rule Configuration Entry

To modify an IP rule that you have created:

1. Double click on an existing rule in the Rules List. The Modify IP Rule Configuration dialog box appears.
2. Modify the appropriate fields and click OK to apply the changes.

Deleting Rule Configuration Entry

To delete a rule select a rule from the Rules List and click Delete.

Defining Services

The firewall comes preloaded with a default set of services.

After you have defined a rule(s), you need to add the rule(s) to a service. Select Traffic Control from the configuration client navigation tree and double-click on Connection Templates, then select Services. The Services List dialog box appears. Double click NEW to get the Add Service dialog box, as shown in Figure 43 on page 120.

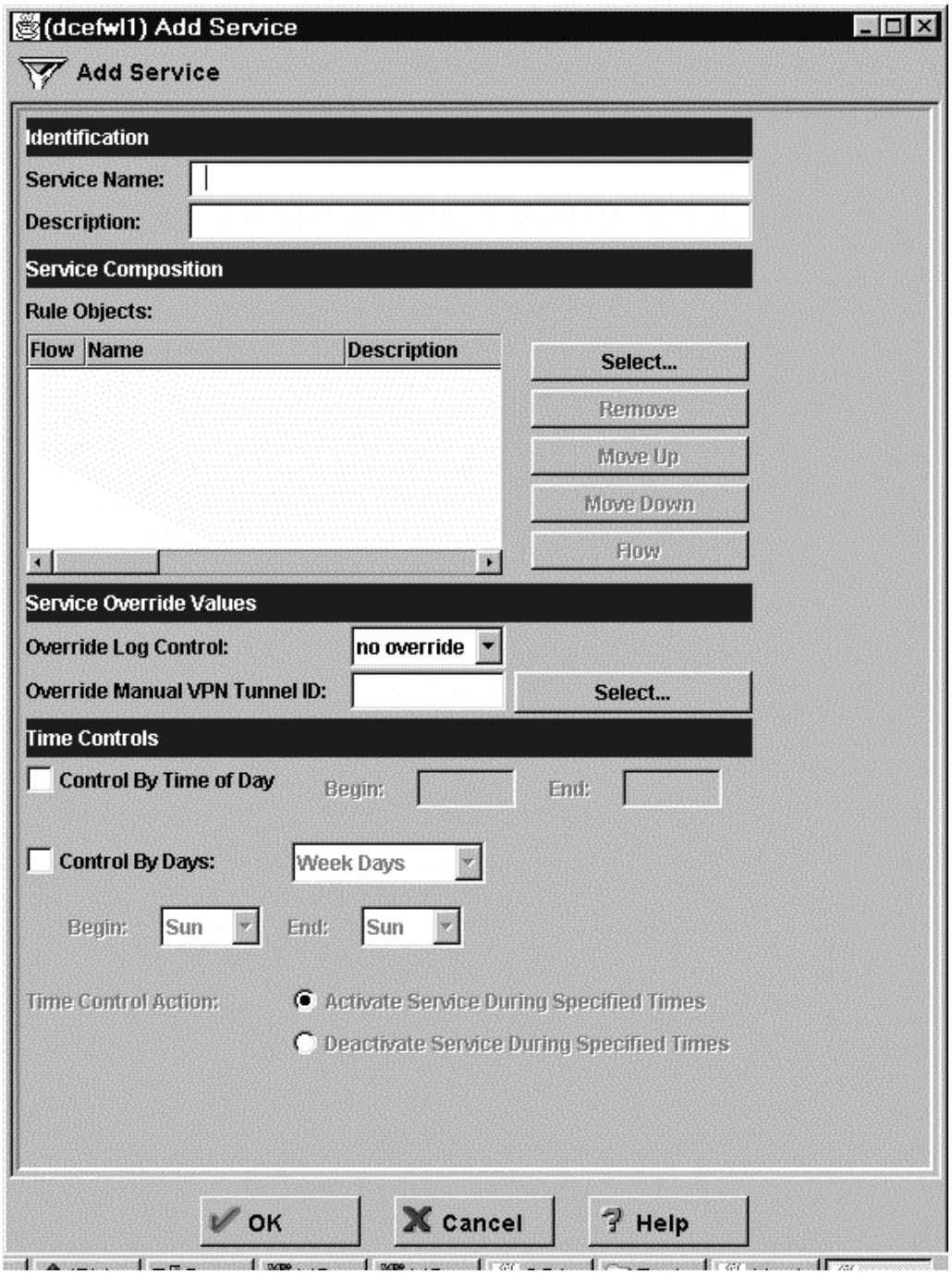


Figure 43. Add a Service

Creating a Service

After you have defined a rule, you need to add the rule to a service. Select Traffic Control from the configuration client navigation tree and double click on Connection Templates, then select Services The Services List dialog box appears. Double click NEW to get the Add Service dialog box, as shown in Figure 43.

1. Enter the service name.
2. Enter a description.

3. Click Select to choose the rules that comprise this service.
4. Use the Flow toggle to determine how the Source and Destination values of the Connection should be assigned to the filters as they get written to the Rule Base file.

   → Left to Right indicates that the Source and Destination of the Connection gets written directly to the rule as it is written to the Rule Base File.

   ← Right to Left indicates that the Source and Destination of the Connection, as specified on the connection panel (see "Building Connections Using Predefined Services" on page 125), gets reversed when it is written to the Rule Base File.
5. When a packet is received, the firewall compares the information in the packet to the rules in the rules configuration file starting at the top of the file. It stops comparing when the first match is found and performs the action contained in the rule.

   Once you have added a series of rules to the service, you can change their order. Select a rule from the Service Objects list and click the Move Up or Move Down buttons to reposition the rule. Or, you can remove a rule by clicking Remove. The configuration client displays a refreshed list of rules. Click OK to save your changes.
6. The Override Log Control field provides a means of overriding the log control setting in the rule templates that was selected for this service. For example, if you include a set of rule templates that ordinarily have log control set to no, you can override this setting to be yes for the purposes of this service. The override setting will act on all of the rules in this service. In the Override Log Control field, enter one of the following choices:

   - no override - override is turned off, the settings in the rules themselves still apply
   - yes - write a log record when any rule in this service is matched
   - no - do not write a log record when any rule in this service is matched

   When a log record is written for a filter rule, the values shown in the log record are the actual values from the IP packet. Logging matched filter rules can provide valuable information about the content of IP packets seen by the Firewall, for example, actual protocol and port numbers.
7. The Override Manual VPN Tunnel ID field provides a way to override the manual tunnel ID setting in the rule templates that were selected for this service. For example, if you include a set of rule templates that ordinarily have no Tunnel Setting, you can override this setting to include a Tunnel ID for all of the rules in this service. If you leave the field blank, override is turned off. The settings in the rules themselves still apply. In the Tunnel ID field, select a tunnel by clicking Select.
8. The time controls allow you to associate a time range with each service. Therefore, this service will only be valid in a specified time period. If there is no time specification for a service, that service is valid all the time.

   Note: You cannot specify a time that spans 2 days (wraps concurrent days). To do this, you must set one to 23:59 and start another at 00:00.

   Control by Time of Day

   Select if you want this service to be activated or deactivated according to begin and end times during the day. Use a 24-hour format. If this field is not enabled, the Time of Day fields will be in effect 24-hours a day.

   Control by Days

   Select if you want this service to be activated or deactivated according to a schedule based upon either days of the week or calendar dates. Note that whether a service is activated or deactivated depends on the value of the Time Control Action field.

Time Control Action

Choose Activate Service During Specified Times if you want this service to be activated during the specified times. This service will be deactivated during the times outside of those specified.

Choose Deactivate Service During Specified Times if you want this service to be deactivated during the specified times. This service will be activated during the times outside of those specified.

Configuring the Socks Server

Socks templates are rules that control security through the socks server. The socks templates allow you to customize, add to, copy, or delete existing socks templates. These socks templates, in turn, can be used in the definitions of connections on the Firewall in the same way rules templates are used.

Adding a New Socks Rule

To add a rule to the socks configuration file using a socks template provided by the configuration client, select Traffic Control from the configuration client navigation tree. Double click on the file folder icon to expand the view. Select Connection Templates. Double click on the file folder icon to expand the view. Select Socks. The Socks dialog box appears.

1. Double click NEW to add a new socks template.

   The Add a Socks Rule dialog box appears, as shown in Figure 44.

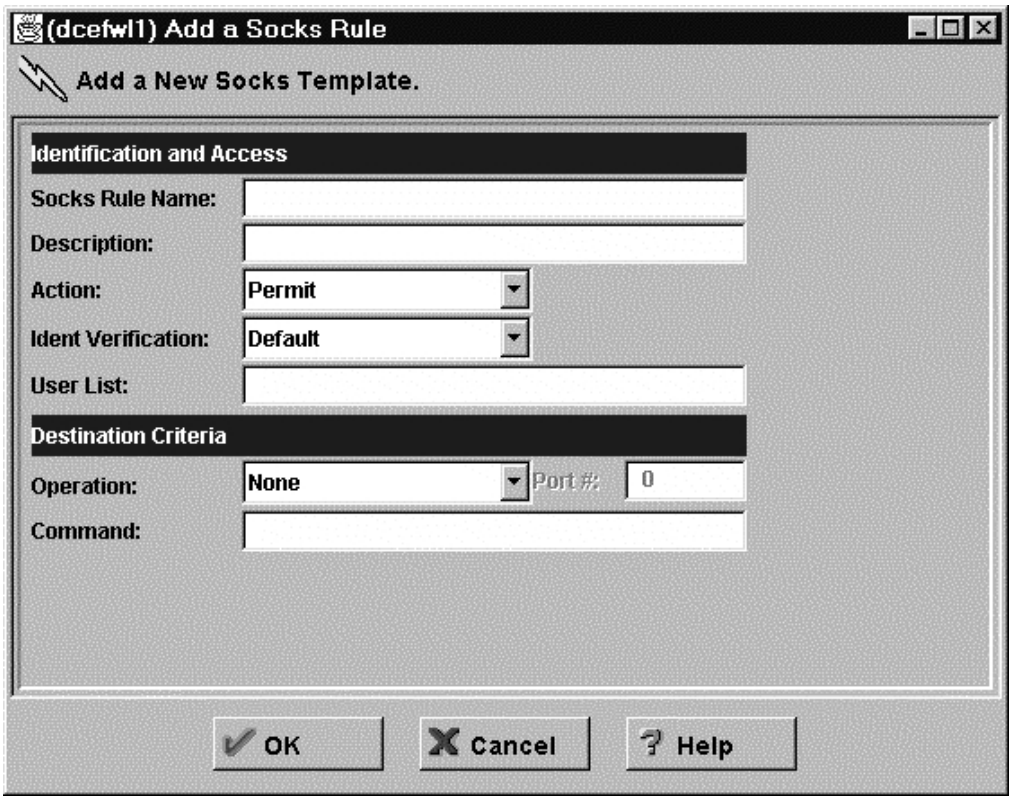


Figure 44. Add a Socks Rule

2. In the Socks Rule Name field, enter the name of the socks entry. This name must be unique and should not contain a vertical bar (|), a single quote (or apostrophe) character ('), or a double quote(") character. Use of these characters will result in unreliable data.
3. Fill in a description.
4. Click the Action arrow and choose to either Permit or Deny access from a source to a destination.

   When a datagram comes into the socks server, the server compares the datagram specifications to each rule in the configuration file starting with the first rule until it finds a rule that matches exactly. Then it stops searching and performs the relevant action (either permit or deny access) on that rule. If no match is found, access is denied automatically.
5. Specify whether Identd Verification should be used for this rule. The Identd server provides identification of users in your network. If you are using an Identd server, use this field to specify how you want to use the results of the identification. The User List can also be used by adding IDs that will be matched.

   Default Use the identification option specified in the configuration for **sockd**, if any is specified. See "fwdaemon" on page 201 for more information about displaying or changing server configuration.

   Always The Identd must be used to verify the user's identity. Access is denied if a connection to the client's `identd` fails or if the result does not match the user ID reported by the client program. This overrides the setting specified in the configuration for **sockd**. See "fwdaemon" on page 201 for more information about displaying or changing server configuration.

   If Configured
   Also specifies the use of `identd`, but denies access only if the client's `identd` reports a user ID different from what the client program claims. This overrides the setting specified in the configuration for **sockd**. See "fwdaemon" on page 201 for more information about displaying or changing server configuration.

   Do Not Use
   Do not use the `identd` program. This overrides the setting specified in the configuration for **sockd**. See "fwdaemon" on page 201 for more information about displaying or changing server configuration.
6. In the User List field, you can enter a user ID, a list of user IDs, a file name, or a list of file names. If you enter a list, separate the entries with commas. File names must be fully qualified (including the leading "/"). Do not use spaces, tabs, the vertical bar (|) or double quotes(") in the user list.
   - The user list is limited to 396 characters.
   - User IDs must be IDs of users on the requesting host, not those on the destination host or socks server host.
   - A user ID can consist of 1 to 8 characters, including:
     - a through z
     - A through Z
     - 0 through 9
     - _ (underscore)
7. A user ID cannot contain a vertical bar (|) or double quote character(").
8. If file names are used, they must be fully qualified (with the leading "/" to prevent their being interpreted as user IDs). Each file can contain a list of user IDs, with one or more per line, separated by commas, and optionally including a comment that is delimited with the # character. Full comment lines - those that begin with the # character are also supported. Each line in the file can be up to 1023 characters long and must be terminated by a "newline" character.

9. In the Operation field, enter the operation to be performed on the port number:
   - Equal to
   - Not equal to
   - Less than
   - Greater than
   - Less than or equal to
   - Greater than or equal to

   When used with Port Number, the operation establishes a relationship that must be met. For example, if you enter the Operation gt and Port Number 23, then the port number must be greater than 23 for the rule to be processed.
10. Enter the number of a port in the Port # field. The Port Number is used with the Operation to establish a relationship that must be met. For example, if you enter the Operation gt and Port Number 23, then the port number must be greater than 23 for the rule to be processed. If operation and port number are omitted, the rule applies to all destination port numbers.
11. In the Command field, enter a command string to be run when the conditions in this rule are met. The following substitutions occur before the string is presented to the UNIX shell for running:

```
%A        replaced by the client host's domain name if
          known, by its IP address otherwise
%a        replaced by the client host's IP address
%c        replaced by connect or bind, the command
          sockd is asked to execute
%p        replaced by the process id of sockd
%S        replaced by the service name (for example, ftp)
          if known, by the destination port number other-
          wise
%s        replaced by the destination port number
%U        replaced by the user-id reported by identd
%u        replaced by the user-id reported by the client
          program
%Z        replaced by the destination host's domain name
          if known, by its IP address otherwise
%z        replaced by the destination host's IP address
%%        replaced by a single %
```

    You can string together several shell commands in a line with a | or ; symbol.

Use this Add a Socks Rule dialog box to permit or deny firewall access to network hosts based on the IP address.

Modifying a Socks Rule

1. Double click on an entry on the Socks dialog box.

   The Modify a Socks Rule dialog box appears.
2. Change the appropriate fields as described in “Adding a New Socks Rule” on page 122 and click OK.

Deleting a Socks Rule

Select an entry from the Socks dialog box and click Delete You are asked to confirm that you want to delete this socks rule. Click OK to delete the rule.

Building Connections Using Predefined Services

To build a connection, select Traffic Control from the configuration client navigation tree and double-click the file folder icon to expand the view. Select Connection Setup. The Connections List dialog box appears. Select NEW and click Open. The Add a Connection dialog box appears, as shown in Figure 45 on page 125.

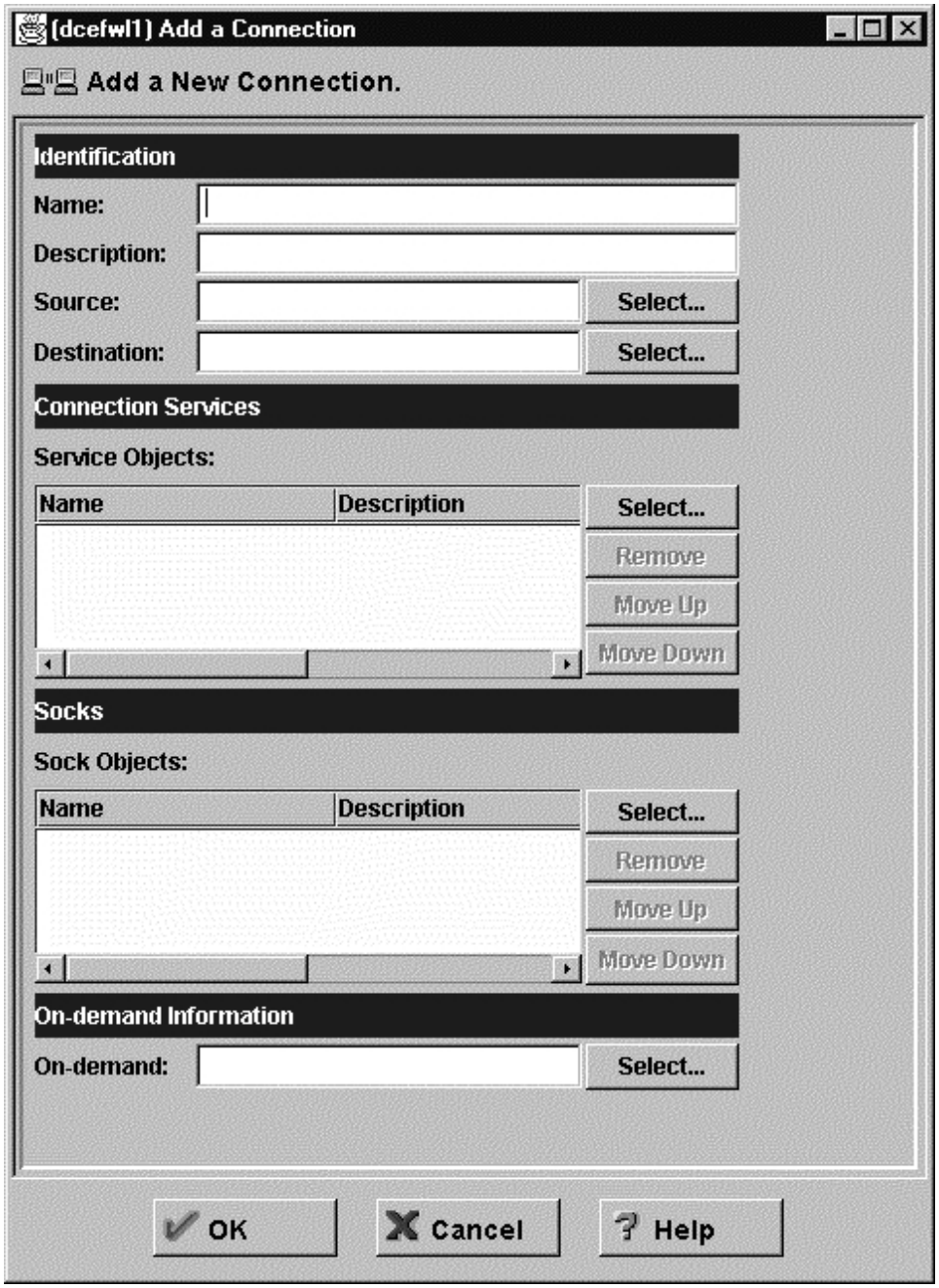


Figure 45. Add a Connection

1. Fill in a name for the connection.
2. Fill in a description of the connection.
3. For the source field, click Select and choose a network object from the Network Object dialog list.

4. For the destination field, click Select and choose a network object from the Network Object dialog list.
5. To choose the services objects for this connection, click Select and choose the type of traffic you want to control between the endpoints.
6. Choose one or more services from the list to add the service to the Connection.
7. You can reorder the list by selecting a service and clicking Move Up Move Down. See “Ordering Connections”
8. You can remove a service by selecting it and clicking Remove.
9. To choose Socks objects for this Connection, follow steps 5–7 to make connections for Socks.
10. To create an on-demand dynamic VPN connection, click Select and choose the on-demand definition which identifies the granularity and key server information to be associated with the connection.
11. After you have everything defined, click OK.
12. Activate all of your connections. See “Activating Connections.”

Ordering Connections

The more rules you have, the greater the impact there will be on performance.

When a packet is received at a network interface, whether going into or out of the firewall host, rules are applied starting at the top of the generated connection rules. When the information from the packet exactly matches the information in a rule, the action (permit or deny) is taken. If the entire file is searched without a match, the request is denied.

Note: Tip:

Place more specific connections closer to the top and less specific connections closer to the bottom. For example, you might have a Department ABC, with an address of (1.1.10.X) and a machine that is used as a server inside of Department ABC, with an address of (1.1.10.7). If you want to exclude machine 1.1.10.7 because it is a server that should not be used for telnet traffic, you must place the connection `Deny telnet for Dept ABC server` before the `Permit telnet for Dept ABC` connections. If you reverse the order of the connections, the deny connection will never be encountered.

Activating Connections

Activate the filter rules and the socks rules by activating connections.

1. Select Connection Activation client navigation tree to do any of the following:

   Regenerate and Activate Connection Rules
   The Firewall builds the generated connection rules from the connection configuration and activates that rule set.

   When pre-decap filtering is enabled, all AH and ESP packets are first verified to see if the AH and/or ESP packet is allowed to flow between the local host and the sender before it is decapsulated. If not enabled, an attempt to decapsulate all AH and ESP packets is done without consulting the filter rules. As a result, AH and ESP filter rules do not need to be specified for each host that the local host has an IPSec tunnel with.

   Deactivate Filter Rules
   The Firewall is now protected by the default rules.

   List Last Generated Filter Rules
   Displays the most recently generated filter rules.

List Current Active Filter Rules
Displays the filter rules based on what the TCP/IP stack considers active.

List Last Generated Socks Rules
You see the most recently generated socks rules.

Validate Rule Generation
The rules you have created are either valid or invalid.

Enable Connection Rules Logging
The Firewall logs selected traffic to the **local4** log facility.

Disable Connection Rules Logging
Stops the Firewall logging.

2. The Connection Activation dialog box appears, as shown in Figure 46 on page 128. Make a selection, then click Execute.

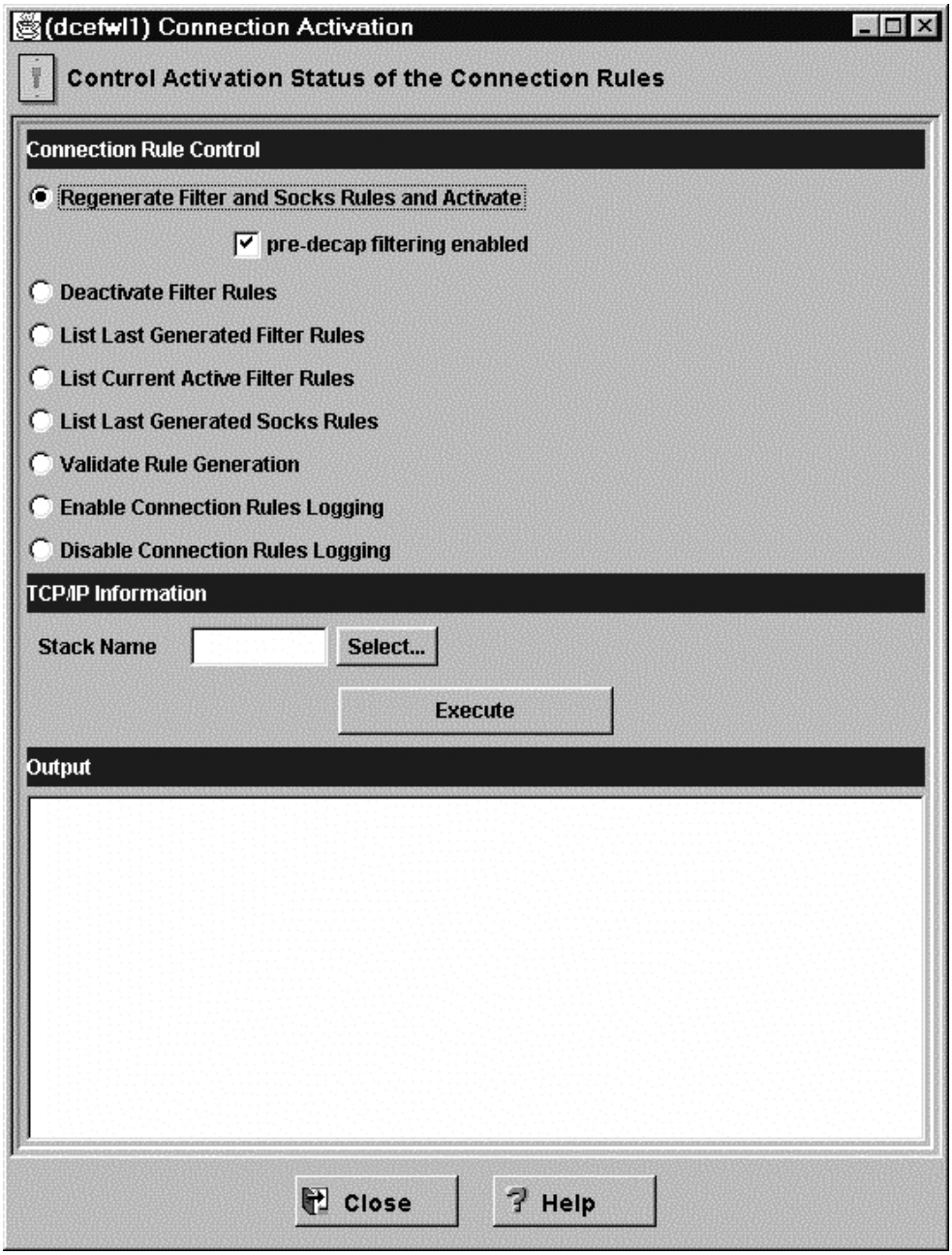


Figure 46. Connection Activation

Configuring Archiving

You can use the configuration client for log archive management. It is assumed that your available disk space is sufficient to contain all the log information. The Firewall generates info, notice and error information to the **local4** log facility. Administrative audit log information goes to the **local0** log facility.

For **fwlogtxt** to function properly, it is important that only **local4** log messages appear in their input files. No other facility should be directed to the same file as **local4** log, so set firewall logging accordingly.

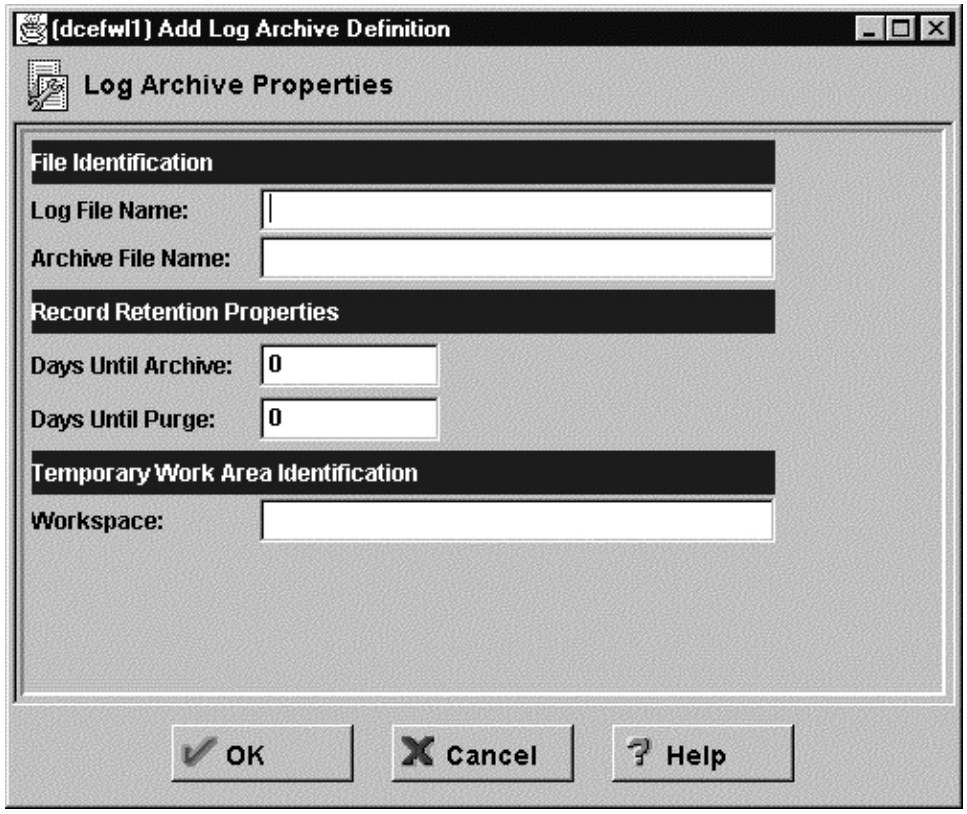


Figure 47. Add Log Archive Definition

Add a log archive definition by completing the following information:

1. Enter the log file name in the file identification section. This must be specified as a fully-qualified name (starting with a "/"), for example, `/tmp/local4/info`.
2. Enter the archive file name. The archive file name must be a fully-qualified name, and must end with a **.a** suffix.
3. Enter the number of days until archive. This number indicates how many days a **syslog** record will remain in the logfile before being moved to the archive file.
4. Enter the number of days until purged. This number indicates how many days a **syslog** record will remain in the archive file before being purged.
5. In the temporary work area identification section, you define a directory that the archiving program should use for temporary work files when archiving **syslog** files. This should be a fully-qualified directory name.
6. Click OK after filling in fields to create the log archive definition.

Configuring the Domain Name Server

To configure DNS, select System Administration from the configuration client navigation tree. Double click the file folder icon to expand the view. Select Domain Name Services. The firewall displays the current DNS configuration, which you can modify.

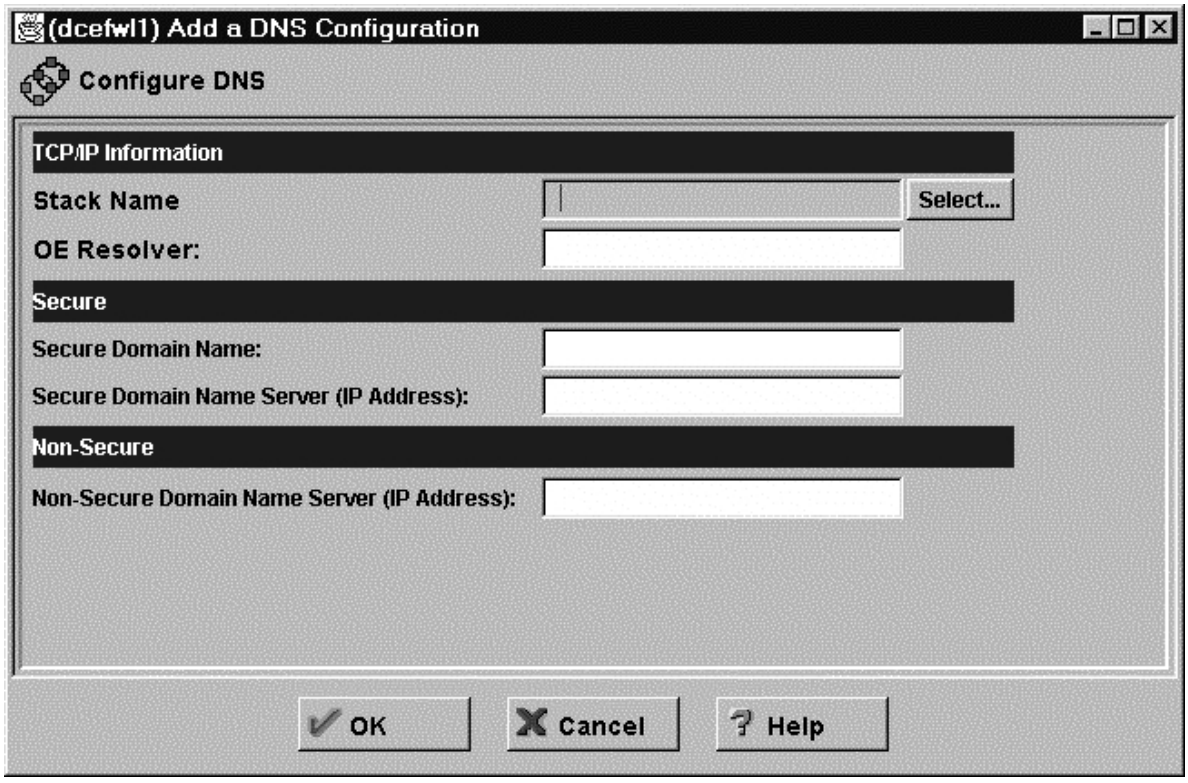


Figure 48. Domain Name Service

Note: When you add DNS, the firewall saves and renames any existing domain-name service configuration files.

1. The Stack Name field identifies the name of the TCP/IP stack which lets DNS know where the DNS configuration files are for each stack.
2. The OE Resolver field identifies the TCP/IP services resolver file.

   Note: This is optional and defaults to **/ect/security/fwstacks/**<*STACK*>**/fwresolv.conf**
3. The Secure Domain Name field identifies the domain name which the Firewall will append to any unqualified host names.
4. The Secure Domain Name Server field refers to the server that resolves names and IP addresses for the hosts protected from the Internet by the OS/390 Firewall Technologies. You must enter dotted-decimal IP addresses, separated by spaces.
5. The Nonsecure Domain Name Server field refers to the server(s) provided by your service provider to resolve information that about the nonsecure network. You must enter dotted-decimal IP addresses, separated by spaces.

Configuring Tunnels

This section describes how to use the configuration client to configure your tunnels on the firewall.

Configuring Manual Tunnels

Select Virtual Private Network from the configuration client navigation tree. Double click the Manual file folder icon to expand the view. Select Manual VPN Tunnel, to display the Manual VPN Administration panel, shown in Figure 49 on page 131.

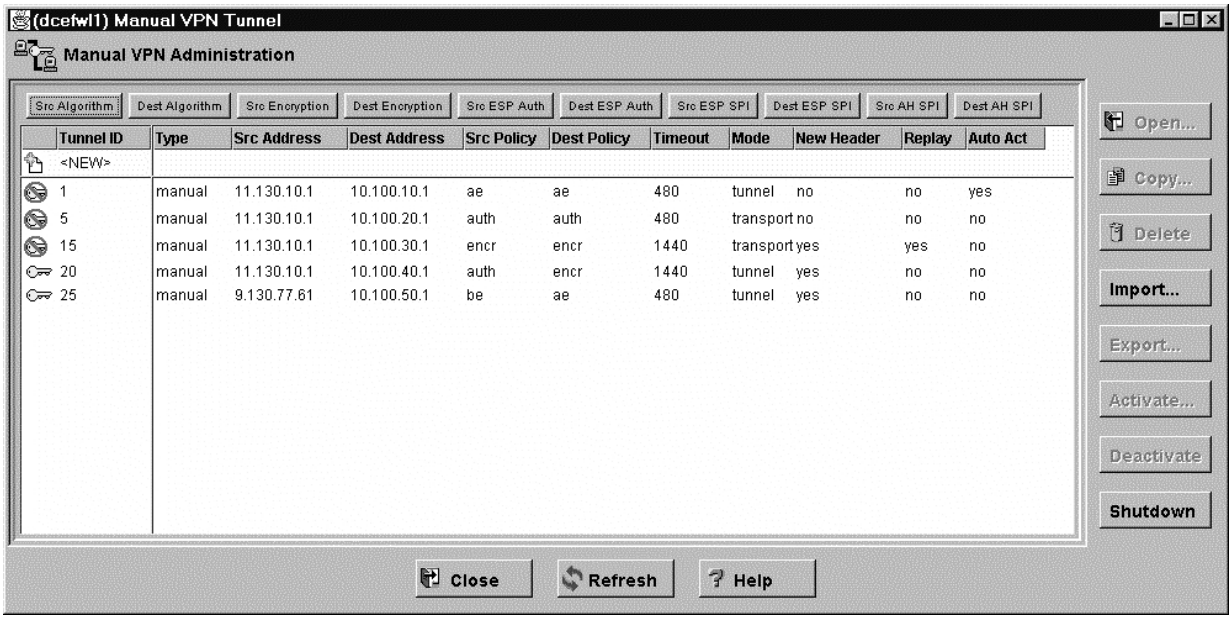


Figure 49. VPN Tunnel List

From here, you can add, delete, import, export, activate, deactivate, and shutdown a tunnel. The buttons across the top allow you to customize how the list of tunnels is displayed.

The *copy* function works the same way as adding a tunnel, except that it only copies the information from a tunnel that already exists.

Adding a Tunnel

1. Select NEW from the Tunnels dialog box and click Open.

   The three tabs across the top let you select the type of tunnel information you want to specify:

   General Generic tunnel information (Figure 50 on page 132)

   ESP Encryption tunnel information (Figure 51 on page 134)

   AH Authentication tunnel information (Figure 52 on page 137)

   Use the dialog boxes to specify the values required for a tunnel context ID specification.
2. Enter the values in the panel as required:
3. Click OK to add your entries to the Add Tunnel screens.

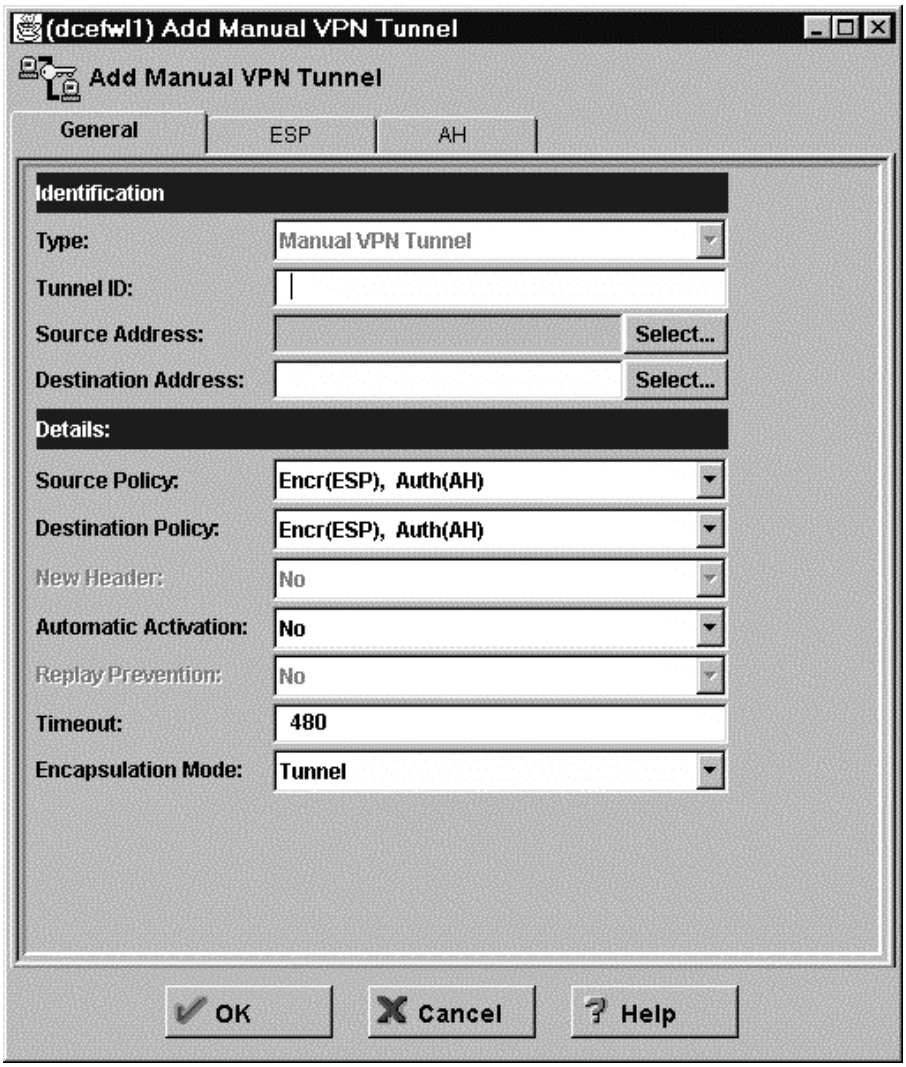


Figure 50. Add a Tunnel - General Information

Use the following parameters for specifying General tunnel information:

Value **Description**

Type Specifies a manual tunnel.

Tunnel ID

The identification number for the tunnel. Enter a number in the field. This number should be the same at both ends of the tunnel, with a value of 1 through 999999, but cannot be a number you have previously used.

Source Address

Specifies IP address of the local firewall nonsecure interface used by the tunnel. Click Select to get the Interface list. Select an interface and click Apply. The local address is added to the Tunnels screen.

Destination Address

Either enter a valid IP address of the destination host or click Select to get a list of all of the network objects. Use the list to select or create a network object. Click OK.

Source Policy

Specifies the source tunnel encryption and authentication classification. Click the arrow to select a policy. Values are:

Encr(ESP), Auth(AH)
authentication after encryption

Auth(AH), Encr(ESP)
authentication before encryption

Auth(AH) authentication

Encr(ESP)
encryption

Click OK and your selection is added to the Tunnels Definition screen.

Note: If the Encr(ESP) policy is specified, you may also specify an ESP authentication algorithm on the ESP panel.

Destination Policy

Specifies the destination tunnel encryption and authentication classification. Click the arrow to select a policy. Values are:

Encr(ESP), Auth(AH)
authentication after encryption

Auth(AH), Encr(ESP)
authentication before encryption

Auth(AH) authentication

Encr(ESP)
encryption

Note: If the Encr(ESP) policy is specified, you may also specify an ESP authentication algorithm on the ESP panel.

New header

Specifies whether the AH and ESP headers are built according to the RFCs 1825-1829 (**No**) or RFCs 2401-2406 and 2410 (**Yes**). New header is set by the algorithm selected and may be changed after the tunnel is created.

Notes:

1. **newheader** must be set to **No** (**No=RFCs 1825-1829**) when the source and destination AH authentication algorithms are KEYED_MD5
2. **newheader** must be set to **Yes** (**Yes=RFCs 2401-2406 and 2410**) when one or more of the following is true:
   - source or destination AH authentication algorithm is not KEYED_MD5
   - source or destination encryption algorithm is 3DES_CBC or ESP_NULL
   - replay is enabled
   - source or destination ESP authentication is specified

Automatic activation

Automatic activation defines manual tunnels that should be activated each time its associated Firewall TCP/IP stack is started. The valid values are Yes and No. The default value is No.

Replay prevention

Replay prevention monitors incoming IP packets to ensure that the system does not receive identical or old packets. Replay prevention is only valid when new header format is specified. The valid values are Yes and No. The default value is No.

Replay prevention cannot be added. Replay prevention may be set to **yes** if New header is set to **yes** after the tunnel is created.

Timeout Specifies the time in minutes that a manual tunnel is operational. Enter a value in the field. The default is 480 and the maximum time allowed is 44640 minutes (31 days). A value of 0 means the tunnel will never expire.

Encapsulation Mode

Specifies the operational mode of the tunnel.

transport Specifies that the tunnel operates in transport mode, which only protects the transport-layer packet (for example, TCP or UDP packet) inside an IP packet. This mode is used when the endpoints of the secure tunnel are the two communicating systems.

tunnel Specifies that the tunnel operates in tunnel mode, which protects the entire IP packet. Usually, this mode is used for a secure tunnel between two firewalls or between a firewall and a remote system; for example, when either of the two communicating systems is not an endpoint of the tunnel. The default value is **mode=tunnel**.

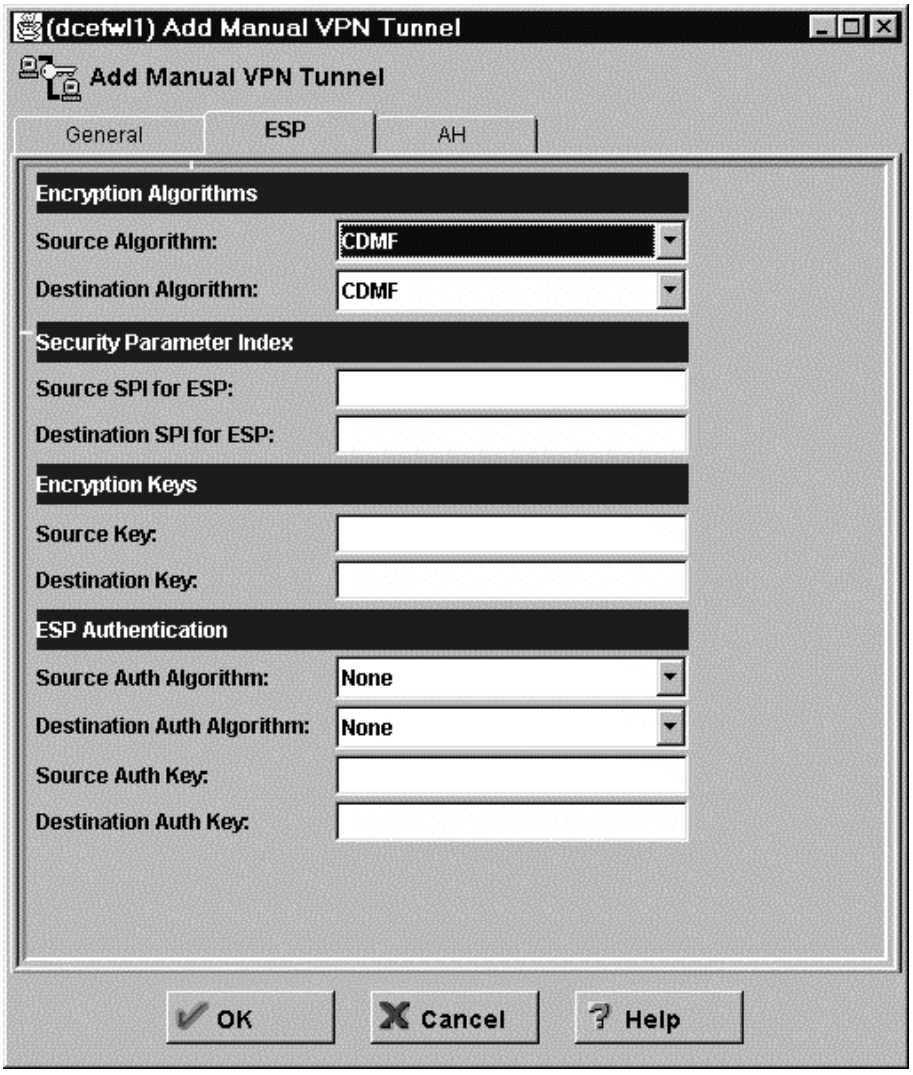


Figure 51. Add a Tunnel - Encryption Information

Encryption Algorithms: Encryption Algorithms specify the algorithm used to provide data confidentiality. If used, you must specify DES_CBC or Commercial Data Masking Facility (CDMF). Click the arrow to choose from CDMF, DES_CBC_4, DES_CBC_8, or 3DES_CBC. DES_CBC_4 uses a 32 bit initialization vector, DES_CBC_8 uses a 64 bit initialization vector, and 3DES_CBC or triple DES, executes the DES encryption algorithm three times and uses a 24 byte key.

Select one of the following for Source Algorithm and Destination Algorithm. Click OK and the encryption algorithm you choose is added to the tunnels screen.

Note: The algorithms are ordered from least to most secure.

ESP_NULL
No encryption is used. This encryption type is only valid in combination with a source ESP authentication algorithm. Use of ESP_NULL requires that the new header format parameter be specified as Yes.

CDMF Commercial Data Masking Facility is a limited version of Data Encryption Standard (DES) that uses a 40 bit key during encryption.

DES_CBC_4
DES encryption is used with a 56 bit key and a 32 bit initialization vector.

DES_CBC_8
DES encryption is used with a 56 bit key and a 64 bit initialization vector.

3DES_CBC
Triple DES executes the DES encryption algorithm three times and uses a 24 byte key. Use of 3DES_CBC requires that the new header format parameter be specified as Yes.

Use the following parameters for specifying Encryption (ESP) tunnel information:

Value Description

Source Algorithm
Specifies the source encryption algorithm used to provide data confidentially. You can only select an algorithm if you have a source policy that includes encryption.

Destination Algorithm
Specifies the destination encryption algorithm used to provide data confidentiality. You can only select an algorithm if you have a destination policy that includes encryption.

Source Key
Specifies the source ESP encryption key. If specified, the key must be a hexadecimal string. Each byte of the key represents a value 00-ff. The length of the key is determined by the source encryption algorithm specified in the source encryption field. The key length (in bytes) for each algorithm type are: ESP_NULL (no key), CDMF/DES_CBC_4/DES_CBC_8 (8) and 3DES_CBC (24).

This parameter is optional. If a key is not specified a system generated key is assigned.

Destination Key
Specifies the destination ESP encryption key. If specified, the key must be a hexadecimal string. Each byte of the key represents a value 00-ff. The length of the key is determined by the destination encryption algorithm specified in the destination encryption field. The key length (in bytes) for each algorithm type are: ESP_NULL (no key), CDMF/DES_CBC_4/DES_CBC_8 (8) and 3DES_CBC (24).

This parameter is optional. If a key is not specified a system generated key is assigned.

ESP Authentication
Specifies the authentication algorithm associated with a policy of encryption only. If a policy other than encryption only is specified, these fields are not selectable.

Source Auth Algorithm

Specifies the source ESP authentication algorithm. When specified, ESP authentication also requires that the new header format parameter be specified as Yes.

Note: The algorithms are ordered from least to most secure.

HMAC_MD5

computes the authentication checksum by combining a 128 bit key, the Hash-based Message Authentication Code (HMAC) authentication algorithm and the MD5 hash algorithm.

HMAC_SHA

computes the authentication checksum by combining a 160 bit key, the HMAC authentication algorithm and the Secure Hash Algorithm (SHA) hash algorithm.

Destination Auth Algorthim

Specifies the destination ESP authentication algorithm. When specified, ESP authentication also requires that the new header format parameter be specified as Yes.

Source Auth Key

Specifies the source ESP authentication key. The key must be specified in hexadecimal. Each byte of the key represents a value 00-ff. The length of the key is determined by the source ESP authentication algorithm specified in the Source Auth Algorithm description. The key length (in bytes) for each algorithm type are: HMAC_MD5 (16), HMAC_SHA (20).

This parameter is optional. If a key is not specified a system generated key is assigned.

Destination Auth Key

Specifies the destination ESP authentication key. The key must be specified in hexadecimal. Each byte of the key represents a value 00-ff. The length of the key is determined by the destination ESP authentication algorithm specified in the Destination Auth Algorithm field. The key length (in bytes) for each algorithm type are: HMAC_MD5 (16), HMAC_SHA (20).

This parameter is optional. If a key is not specified a system generated key is assigned.

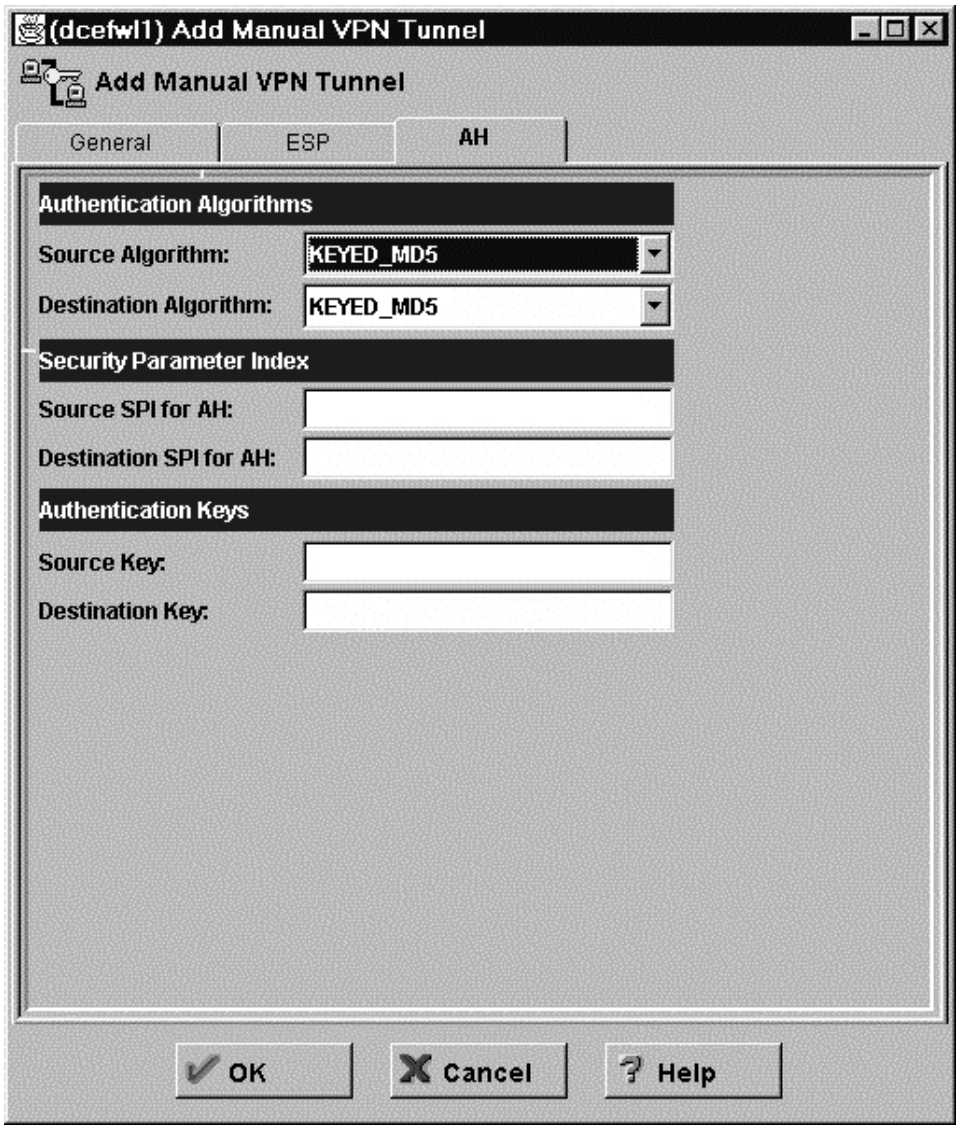


Figure 52. Add a Tunnel - Authentication Information

Authentication Algorithms: Authentication Algorithms specify the algorithm used to produce a checksum for IP packet authentication and integrity verification. You can only select an algorithm if you have a source policy that includes authentication.

Note: The algorithms are ordered from least to most secure. **KEYED_MD5** requires that the new header format parameter be specified as No. **HMAC_MD5** and **HMAC_SHA** requires that the new header format parameter be specified as Yes.

Select one of the following for Source Algorithm and Destination Algorithm. Click OK and the encryption algorithm you choose is added to the tunnels screen.

Note: The algorithms are ordered from least to most secure.

KEYED_MD5
computes the authentication checksum by combining a 128 bit key with the Message Digest-5 (MD5) hash algorithm.

HMAC_MD5
computes the authentication checksum by combining a 128 bit key, the Hash-based Message Authentication Code (HMAC) authentication algorithm and the MD5 hash algorithm.

HMAC_SHA
computes the authentication checksum by combining a 160 bit key, the HMAC authentication algorithm and the Secure Hash Algorithm (SHA) hash algorithm.

Use the following parameters for specifying Authentication (AH) tunnel information:

Value Description

Source Algorithm
Specifies the source authentication algorithm used to produce a checksum used for IP packet authentication and integrity verification.

Destination Algorithm
Specifies the destination authentication algorithm used to produce a checksum used for IP packet authentication and integrity verification. For more information on the authentication algorithms, see Authentication Algorithms.

Source SPI for AH
Specifies the source AH Security Parameter Index. Valid values are 1 - 4294967294. This parameter is optional. If not specified, the value in Source SPI for ESP is used.

Destination SPI for AH
Specifies the destination AH Security Parameter Index. Valid values are 1 - 4294967294. This parameter is optional. If not specified, the value in Destination SPI for ESP is used.

Source Key
Specifies the source AH authentication key. The key must be specified in hexadecimal. Each byte of the key represents a value 00-ff. The length of the key is determined by the source authentication algorithm specified in the Source Algorithm field. The key length (in bytes) for each algorithm type are: KEYED_MD5 and HMAC_MD5 (16), HMAC_SHA (20).

This parameter is optional. If a key is not specified, a system generated key is assigned.

Destination Key
Specifies the destination AH authentication key. The key must be specified in hexadecimal. Each byte of the key represents a value 00-ff. The length of the key is determined by the destination AH authentication algorithm specified in the Destination Algorithm field. The key length (in bytes) for each algorithm type are: KEYED_MD5 and HMAC_MD5 (16), HMAC_SHA (20).

This parameter is optional. If a key is not specified, a system generated key is assigned.

Changing a Tunnel

You cannot modify an active tunnel; you have to deactivate the tunnel first.

1. Select a tunnel from the Tunnels dialog box and click Open.
2. Modify the fields on the Modify Tunnel dialog box and click OK.

Deleting a Tunnel

You cannot delete an active tunnel; you have to deactivate the tunnel first.

If the tunnel that you want to delete is used in a rule or service, you must first remove the reference to the tunnel on the Services panel. To do this, double-click Traffic Control from the configuration client navigation tree. Double click Connection Templates. Select Services. Double click on the service you want to modify on the Services List panel. The Modify Service dialog box appears. Click Select and choose a tunnel that is not used in any service, from the Select a Tunnel dialog box. Click Apply to place the tunnel ID in the Override Tunnel ID field. This removes the original tunnel reference and you are now able to safely delete the tunnel.

1. Select the tunnel you want to delete from the Tunnels dialog box and click Delete.

   The configuration client asks you to confirm your request.

2. Click Yes to confirm the delete.

   The configuration client confirms your request.

Note: If you delete a tunnel and then add it again, you have to export it again.

Exporting Tunnel Definition Files

As a tunnel owner, after you have defined a set of tunnel context definitions, you will export one or more of these definitions to a tunnel partner. See Figure 53.

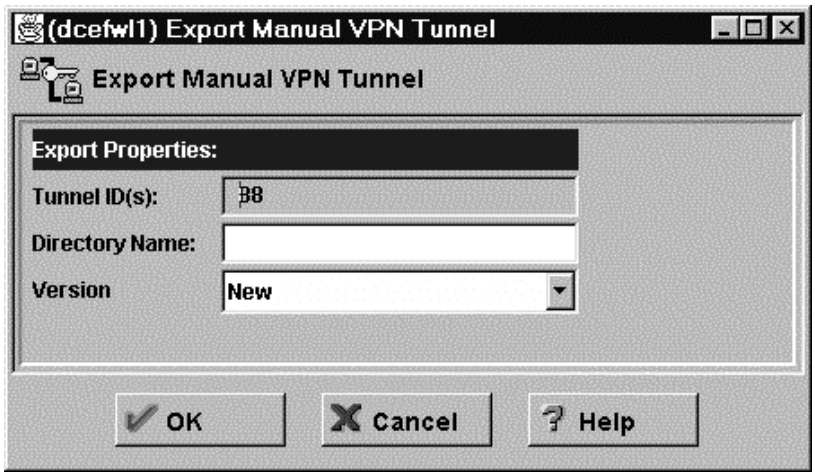


Figure 53. Export properties

1. Select a tunnel or tunnels from the tunnel first panel and click Export.
2. Enter a directory name in the field.

   The directory is the name of an existing directory where files containing tunnel information is temporarily placed for export to a partner. There cannot already be export (or import) files in the directory.

3. The Version field allows an administrator to create a tunnel export file in the format for Version 2 Release 6 (**V2R6**) and prior, or Version 2 Release 7 (**NEW**) and later.
4. When you complete this operation, the directory contains the name of the files that need to be moved to your tunnel partner's machine. If a directory name of `tmp` was used, the format for the files to be exported are:

```
/tmp/fwexpmctx.manual
/tmp/fwexppolicy
```

5. Move the files to your tunnel partner's machine.

Importing Tunnel Definition Files

1. Select Traffic Control, Virtual Private Network, and click Import.

   A dialog box appears as in Figure 54.

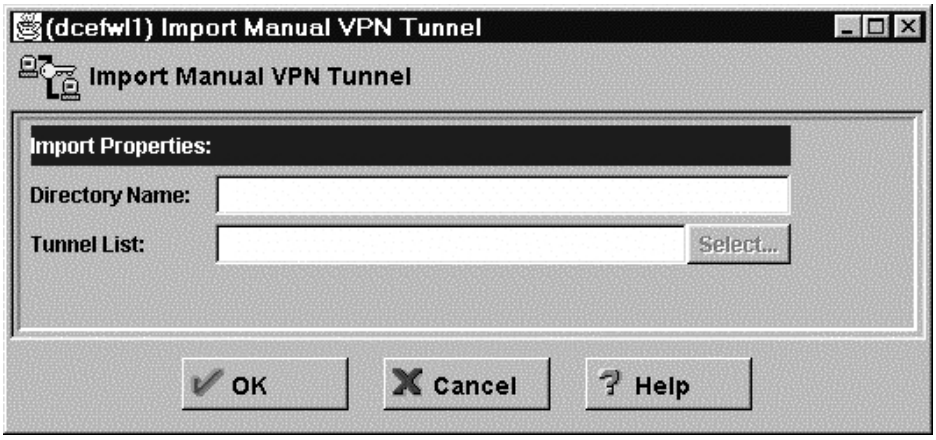


Figure 54. Import properties

2. Enter the name of the directory where you have restored the files you have imported from the firewall.
3. Either enter a list of tunnels and click Apply, or click Select and select the tunnels and click Apply.
4. Click OK after making all selections.

Activating a Tunnel

Use this procedure to activate or deactivate a tunnel(s). When you activate a tunnel, the firewall enables the use of that tunnel. Any filter rules that reference the activated tunnel will be operable.

1. Select a tunnel or tunnels from the Tunnels list and click Activate.

   The Activate a Tunnel dialog box appears.

   One or more items can be selected.
2. Click OK after making all selections.

Deactivating a Tunnel

Use this procedure to stop communication of a tunnel.

1. Select a tunnel from the Tunnels list and click Deactivate.
2. Select the item.

   One or more items can be selected.
3. Click OK after making all selections.

   The tunnel ID to be deactivated is displayed.

Shutting Down a Tunnel

Use shutdown only if you want to stop all tunnel activity for an extended period of time. Or use shutdown for security reasons if you need to stop all tunnel activity immediately.

1. Click Shutdown from the tunnels list panel.
2. The configuration client asks you to confirm the request.
3. Click Yes to shut down.

Building Dynamic Virtual Private Network (VPN) Connections

Building a dynamic VPN connection involves:

- Defining policies - Key Management, Data Management and Dynamic Tunnels
- Defining key server
- Defining authentication information
- Defining anchor rules
- Defining locally activated dynamic connections.

To build a dynamic VPN connection, all information except for defining anchor rules can be found by under the Dynamic folder. To locate the Dynamic folder, select Virtual Private Network from the configuration client navigation tree. Double click the file folder icon to expand the view. Select Dynamic to display the main folder involved with configuring a dynamic VPN connection.

Defining Key Management Information

Dynamic tunnel key management consists of Key Transform(s), Key Proposal and Key Policy information. Key Management is responsible for determining how key negotiations between ISAKMP servers are to be protected and how each server authenticates each other.

Select VPN Connection Templates from the configuration client navigation tree. Double click the file folder icon to expand the view. Double click on the Key Management folder icon to expand the view.

Defining a Key Transform

A key transform defines the protection mechanism used to secure key exchanges between key servers. The transform specifies how the Internet Key Exchange (IKE) protocol should be exploited. Exploitation is based upon an authentication method, hash algorithm, encryption method and Diffie-Hellman group.

Select Key Transform to display the Key Transform Administration panel. Double click on NEW. The Add Key Transform dialog box appears, as shown in Figure 55 on page 142.

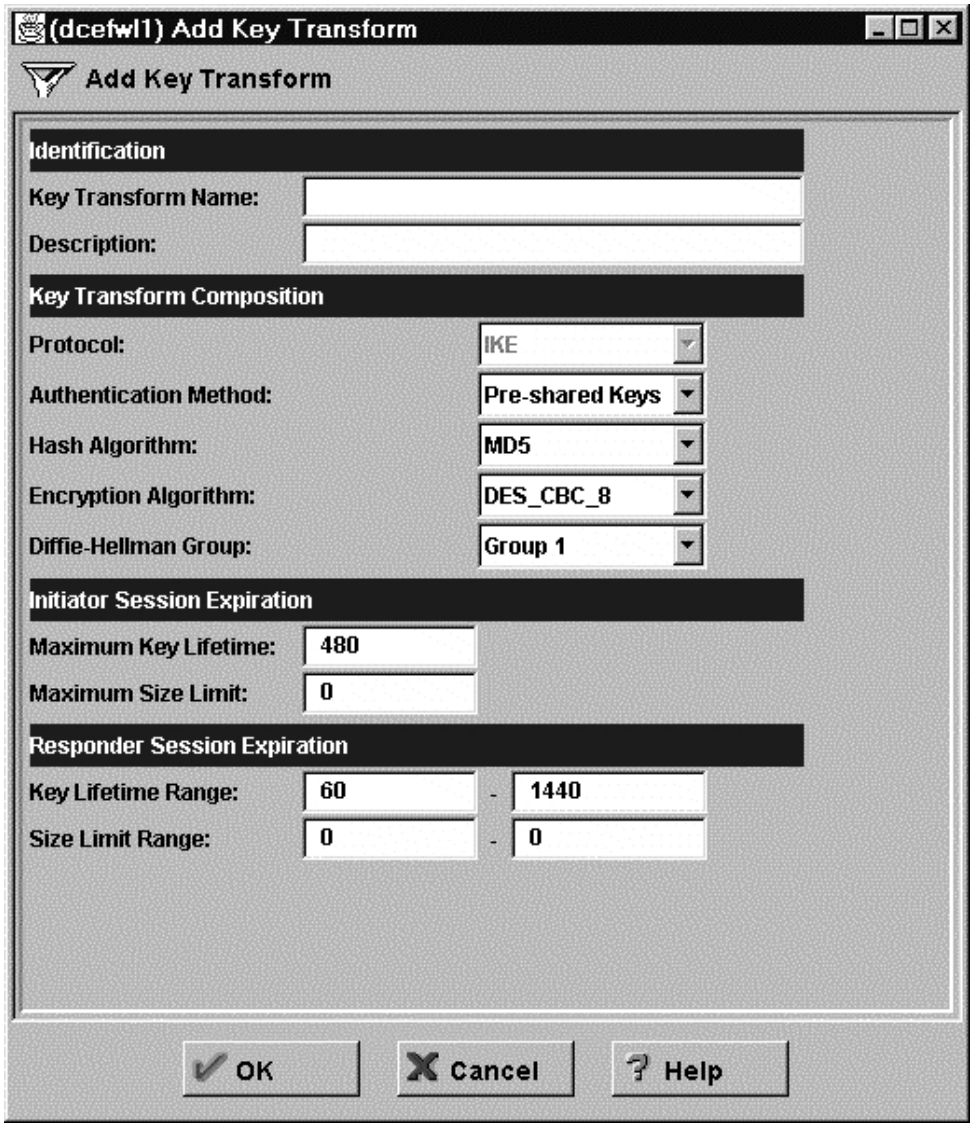


Figure 55. Add Key Transform

1. Fill in the Key Transform Name.
2. Fill in the description. This field is optional.
3. Currently, the only supported protocol is Internet Key Exchange (IKE).
4. Click the Authentication Method arrow and select the authentication method to be used to authenticate the sender of ISAKMP messages.

   Pre-shared Keys

   RSA Signatures
5. Click the Hash Algorithm arrow and select the algorithm to be used to authenticate the key exchange.

   MD5

   SHA
6. Click the Encryption Method arrow and select the method to be used to encrypt the key exchange data.

DES_CBC_8 - DES computes the authentication checksum by combining a 128 bit key, the Hash-based Message Authentication Code (HMAC) authentication algorithm and the MD5 hash algorithm.

3DES_CBC - Triple DES computes the authentication checksum by combining a 160 bit key, the HMAC authentication algorithm and the Secure Hash Algorithm (SHA) hash algorithm.

7. Click the Diffie_Hellman Group arrow and select the group to be used to derive the key information.

   GROUP1 modular exponentiation group with a 768-bit modulus

   GROUP2 modular exponentiation group with a 1024-bit modulus

8. Enter the Initiator's Session Maximum Key Lifetime in minutes. This is the proposed lifetime that the negotiated key remains valid. The key will be renegotiated by the key server prior to this limit being reached. The valid values are 1 to 9999 minutes. This value is used when the local key server initiates the negotiation. The maximum value should be within the Responder Session Key Lifetime Range.
9. Enter the Initiator's Session Maximum Size Limit in kilobytes. This is the proposed amount of data in kilobytes that will be transferred through the tunnel while the negotiated keys remain valid. The key will be renegotiated by the key server prior to this limit being reached. The valid values are 0 to 4194300 kilobytes. Specifying 0 means that the amount of data transferred is not considered when negotiating keys. The maximum value should be within the Responder Session Size Limit Range.
10. Enter the Responder's Session Key Lifetime Range in minutes. This is the range of lifetime values that will be acceptable as a responder. The lifetime that the initiator specifies must fall within this range. The valid values are 1 to 9999 minutes for each value in the range.
11. Enter the Responder's Session Size Limit Range in kilobytes. This is the range size values that will be acceptable as a responder. The valid values are 1 to 4194300 kilobytes or 0. Specifying 0 means that the amount of data transferred is not considered when renegotiating key exchanges. You can not specify 0 in one field and a number in the other field.
12. Click OK after filling in all of the required fields to create the key transform object. You may change any of the fields prior to clicking the OK button.

    After clicking the OK button, you will be returned to the Key Transform Administration panel.

Changing a Key Transform Configuration Entry

To modify a key transform that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Key Transform dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying a Key Transform Configuration Entry

To copy a key transform:

1. Select the entry and click the Copy button. The Copy Key Transform dialog box appears.
2. Modify the appropriate fields, enter a unique name for the transform and Click OK to create a new transform.

Deleting a Key Transform Configuration Entry

To delete a key transform that you have created:

1. Select the key transform entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the key transform is reference by a key proposal, the key transform cannot be deleted until it is removed from the key proposal.

Defining a Key Proposal

Key proposals define a set of ordered transforms to be used by this proposal during key management negotiation.

After you have defined a key transform, you need to add the transform to a key proposal. Select Key Proposal to display the Key Proposal Administration panel. Double click on NEW. The Add Key Proposal dialog box appears, as shown in Figure 56.

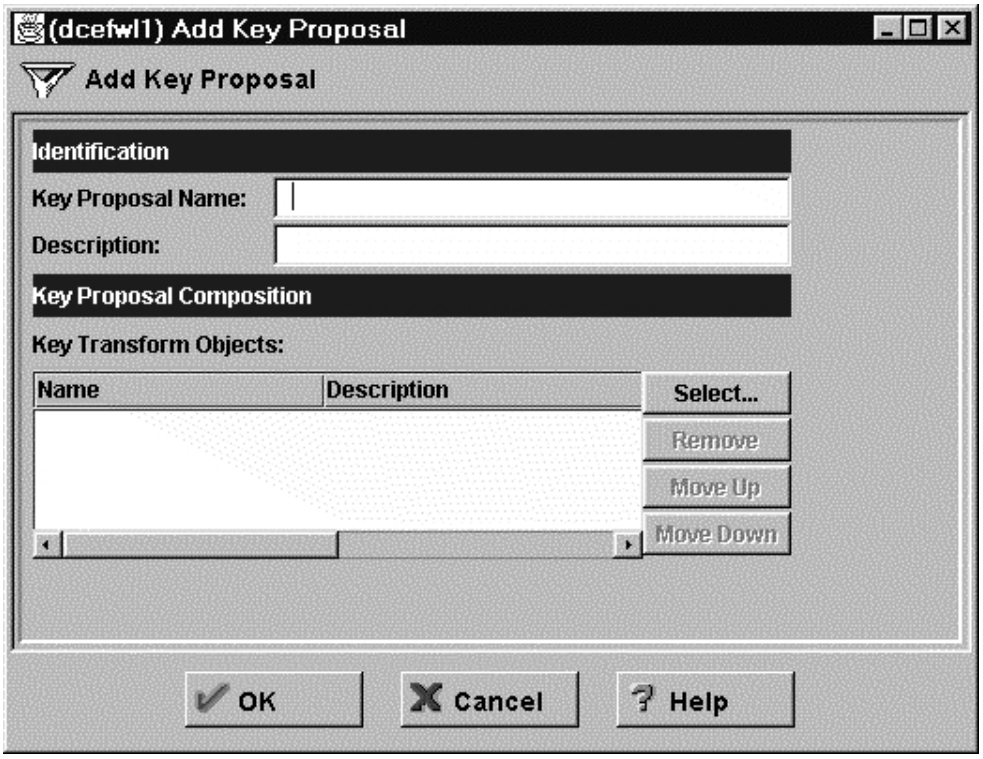


Figure 56. Add Key Proposal

1. Fill in the Key Proposal Name.
2. Fill in the description. This field is optional.
3. To choose the key transforms for this key proposal, click Select and choose the type of key management mechanism you want to use when exchanging keys between the key servers.

   Order of the key transforms is very important. During negotiation, this list is search from top to bottom until an acceptable key transform is found. You can reorder the list by selecting a key transform and clicking Move Up or Move Down.

   You can remove a key transform by selecting it and clicking Remove.

4. Click OK after filling in all of the required fields to create the key proposal object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the Key Proposal Administration panel.

Changing a Key Proposal Configuration Entry

To modify a key proposal that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Key Proposal dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying a Key Proposal Configuration Entry

To copy a key proposal:

1. Select the entry and click the Copy button. The Copy Key Proposal dialog box appears.
2. Modify the appropriate fields, enter a unique name for the proposal and Click OK to create the proposal.

Deleting a Key Proposal Configuration Entry

To delete a key proposal that you have created:

1. Select the key proposal entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the key proposal is reference by a key policy, the key proposal cannot be deleted until it is removed from the key policy.

Defining a Key Policy

Key policy objects define information required when negotiating keys for data exchanges.

After you have defined your key proposal, you need to add the proposals to a key policy. Select Key Policy to display the Key Policy Administration panel. Double click on NEW. The Add Key Policy dialog box appears, as shown in Figure 57 on page 146.

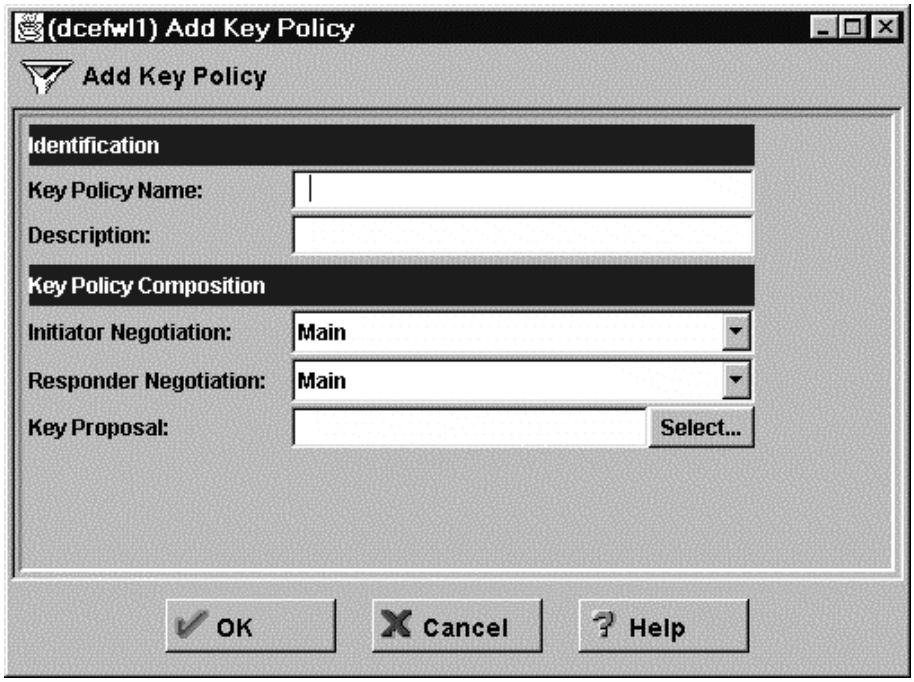


Figure 57. Add Key Policy

1. Fill in the Key Policy Name.
2. Fill in the description. This field is optional.
3. Click the Initiator Negotiation arrow and select the negotiation mode to be used when acting as the initiator of a negotiation request.

   Main Identity protection is to be used during key management negotiation

   Aggressive Identity protection is not to be used during key management negotiation.

   Note: Refer to Appendix H, "Initializing Security Associations with ISAKMP and IKE" on page 463 for a more detailed description of Main and Aggressive mode.
4. Click the Responder Negotiation arrow and select the negotiation mode to be used when acting as the responder to a negotiation request.

   Main Identity protection is to be used during key management negotiation

   Aggressive Identity protection is not to be used during key management negotiation.

   Any Allow negotiation with either identity protection or not.
5. To choose the key proposal for this key policy, click Select and choose the key proposal.
6. Click OK after filling in all of the required fields to create the key policy object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the Key Policy Administration panel.

Changing a Key Policy Configuration Entry

To modify a key policy that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Key Policy dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying a Key Policy Configuration Entry

To copy a key policy:

1. Select the entry and click the Copy button. The Copy Key Policy dialog box appears.
2. Modify the appropriate fields, enter a unique name for the policy and Click OK to create the policy.

Deleting a Key Policy Configuration Entry

To delete a key policy that you have created:

1. Select the key policy entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the key policy is reference by a key server group, the key policy cannot be deleted until it is removed from the key server group.

Defining Data Management Policies

Data management consists of AH Transform(s) and/or ESP Transforms, Data Proposal(s) and Data Policy information. Data Management is responsible for determining how data sent in a dynamic VPN is to be protected.

Select VPN Connection Templates from the configuration client navigation tree. Double click the file folder icon to expand the view. Double click on the Data Management folder icon to expand the view.

Defining an ESP transform

ESP transforms define the protection mechanism used to secure exchanges between the data endpoints via encryption and optionally with authentication. A protection mechanism consists of an encapsulation mode, an authentication algorithm and an encryption algorithm.

Select ESP transform to display the ESP Transform Administration panel. Double click on NEW. The Add ESP Transform dialog box appears, as shown in Figure 58 on page 148.

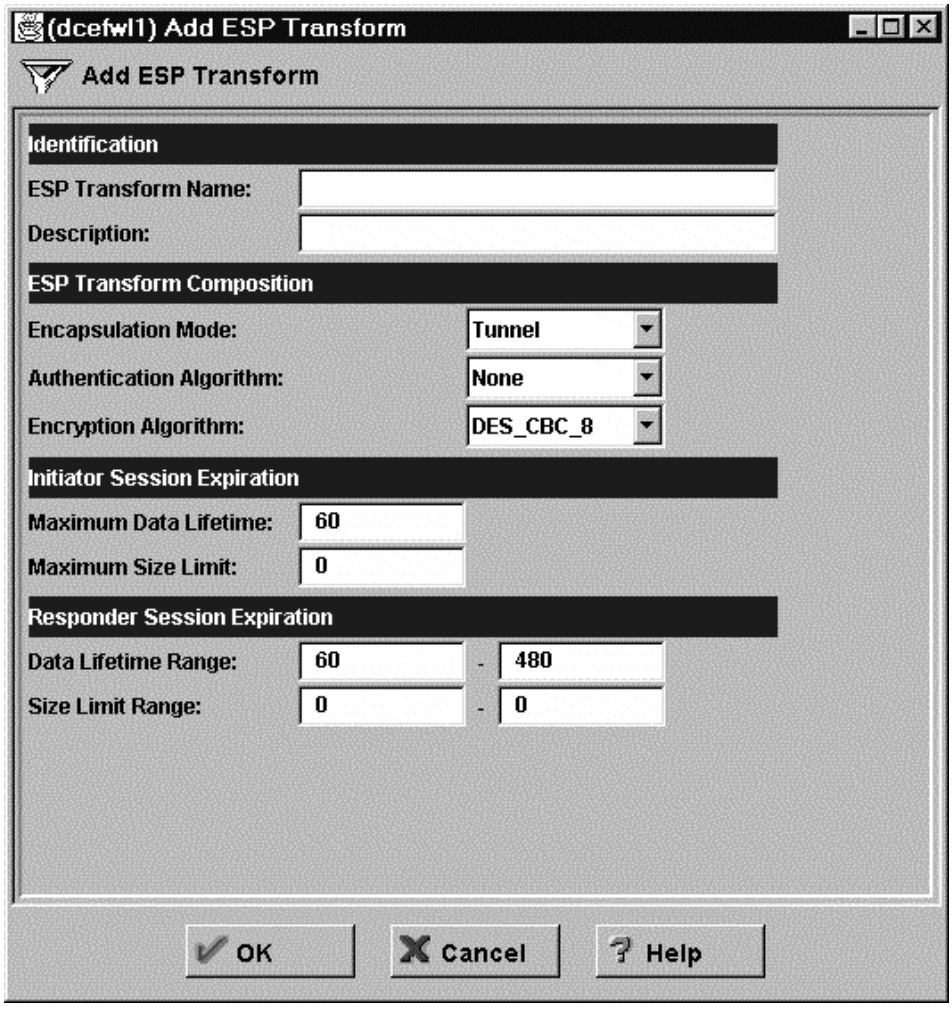


Figure 58. Add ESP Transform

1. Fill in the ESP Transform Name.
2. Fill in the description. This field is optional.
3. Click the Encapsulation Mode arrow and select the type of encapsulation to be applied to the dynamic tunnel.

   Tunnel specifies that the tunnel operates in tunnel mode, which protects the entire IP packet. Usually, this mode is use a secure tunnel between two firewalls or between a firewall and a remote system; for example, when either of the two communicating systems is not an endpoint of the tunnel.

   Transport specifies that the tunnel operates in transport mode, which only protects the transport-layer packet (for example, TCP or UDP packet) inside an IP packet. This mode is used when endpoints of the secure tunnel are the two communicating systems.
4. Click the Authentication Algorithm arrow and select the type of authentication to be performed when data is exchange through the tunnel.

   None specifies that no authentication to be performed

   HMAC_MD5 computes the authentication checksum by combining a 128 bit key, the Hash-based Message Authentication Code (HMAC) authentication algorithm and the MD5 hash algorithm.

HMAC_SHA computes the authentication checksum by combining a 160 bit key, the HMAC authentication algorithm and the Secure Hash Algorithm (SHA) hash algorithm.

5. Click the Encryption Algorithm arrow and select the type of encryption algorithm to be used when encrypting/decrypting data that will be transmitted through the tunnel.

   DES_CBC_8 DES encryption is used with a 56 bit key and a 64 bit initialization vector

   3DES_CBC DES encryption is used with a 24 byte key and a 64 bit initialization vector

   ESP_NULL specifies that no encryption should be done to the data (can only be specified if an authentication algorithm is specified).

6. Enter the maximum data lifetime in minutes. This value represents the proposed initiator lifetime for this security association. The valid range of values is 1 to 9999.
7. Enter the Maximum Size Limit in kilobytes. This value represents the proposed number of kilobytes of data allowed to be transferred through the tunnel by the initiator for this security association. The valid range of values is 0 to 4194300. A value of 0 indicates no size limit.
8. Enter the Responder Data Lifetime range in minutes. This is the lifetime range that is acceptable for the security association responder. The initiator's lifetime must fall within this range. The valid values are 1 to 9999 for each side of the range.
9. Enter the Responder Data Size Limit range in kilobytes. This is the acceptable range by the security association responder for the amount of data in kilobytes allowed through the tunnel. The initiator's size limit must fall within this range. The valid values are 0 to 4194300 for each side of the range. A value of 0 indicates no size limit.
10. Click OK after filling in all of the required fields to create the ESP transform object. You may change any of the fields prior to clicking the OK button.

    After clicking the OK button, you will be returned to the ESP Transform Administration panel.

Changing an ESP Transform Configuration Entry

To modify an ESP transform that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify ESP Transform dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying an ESP Transform Configuration Entry

To copy a ESP transform:

1. Select the entry and click the Copy button. The Copy ESP Transform dialog box appears.
2. Modify the appropriate fields, enter a unique name for the transform and Click OK to create the transform.

Deleting an ESP Transform Configuration Entry

To delete an ESP transform that you have created:

1. Select the ESP Transform entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the ESP transform is referenced by a data proposal, the transform cannot be deleted until it is removed from the data proposal.

Defining an AH transform

AH transforms define protection mechanism used to authenticate data exchanges between the data endpoints. A protection mechanism consists of an encapsulation mode and an authentication algorithm.

Select AH transform to display the AH Transform Administration panel. Double click on NEW. The Add AH Transform dialog box appears, as shown in Figure 59 on page 150.

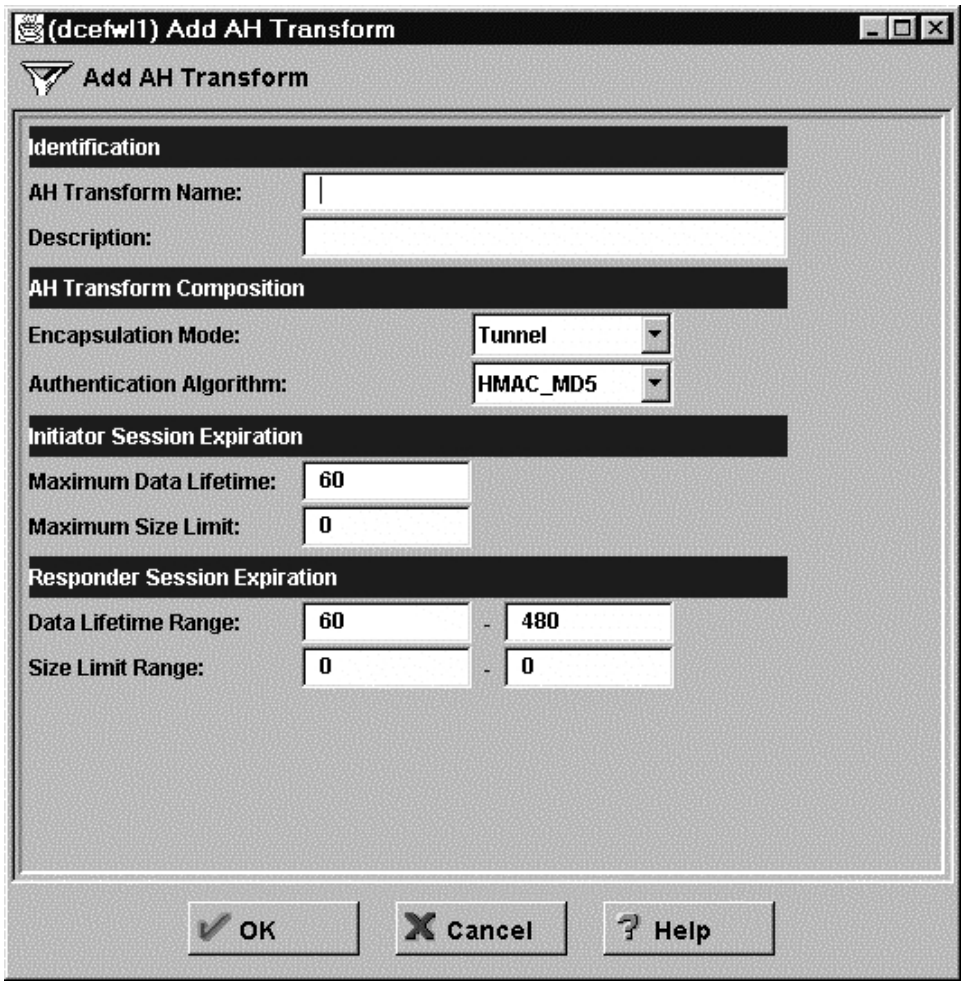


Figure 59. Add AH Transform

1. Fill in the AH Transform Name.
2. Fill in the description. This field is optional.
3. Click the Encapsulation Mode arrow and select the type of encapsulation to be applied to the dynamic tunnel.

   Tunnel specifies that the tunnel operates in tunnel mode, which protects the entire IP packet. Usually, this mode is use a secure tunnel between two firewalls or between a firewall and a remote system; for example, when either of the two communicating systems is not an endpoint of the tunnel.

   Transport specifies that the tunnel operates in transport mode, which only protects the transport-layer packet (for example, TCP or UDP packet) inside an IP packet. This mode is used when endpoints of the secure tunnel are the two communicating systems.

4. Click the Authentication Algorithm arrow and select the type of authentication to be performed to validate the data endpoints.

   HMAC_MD5 computes the authentication checksum by combining a 128 bit key, the Hash-based Message Authentication Code (HMAC) authentication algorithm and the MD5 hash algorithm.

   HMAC_SHA computes the authentication checksum by combining a 160 bit key, the HMAC authentication algorithm and the Secure Hash Algorithm (SHA) hash algorithm.

5. Enter the maximum data lifetime in minutes. This value represents the proposed initiator lifetime for this security association. The valid range of values is 1 to 9999.
6. Enter the Maximum Size Limit in kilobytes. This value represents the proposed number of kilobytes of data transferred through the tunnel for this security association initiator. The valid range of value is 0-4194300. A value of 0 indicates no size limit.
7. Enter the Responder Data Lifetime range in minutes. This is the range that is acceptable for the security association responder. The initiator's lifetime must fall within this range. The valid values are that is acceptable for the security association responder. The initiator's lifetime must fall within this range. The valid values are 1 to 9999 for each side of the range.
8. Enter the Responder Data Size Limit range in kilobytes. This is the acceptable range by the security association responder for the amount of data in kilobytes allowed through the tunnel. The initiator's size limit must fall within this range. The valid values are 0 to 4194300 for each side of the range. A value of 0 indicates no size limit.
9. Click OK after filling in all of the required fields to create the AH transform object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the AH Transform Administration panel.

Changing an AH Transform Configuration Entry

To modify an AH transform that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify AH Transform dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying an AH Transform Configuration Entry

To copy an AH transform:

1. Select the entry and click the Copy button. The Copy AH Transform dialog box appears.
2. Modify the appropriate fields, enter a unique name for the transform and Click OK to create the transform.

Deleting an AH Transform Configuration Entry

To delete an AH transform that you have created:

1. Select the AH Transform entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the AH transform is reference by a data proposal, the transform cannot be deleted until it is removed from the data proposal.

Defining a Data Proposal

Data Proposals define a set of ordered AH and/or ESP transforms to be used by this proposal.

After you have defined an AH and/or ESP transform, you need to add the transform(s) to a data proposal. Select Data proposal to display the Data Proposal Administration panel. Double click on NEW. The Add Data Proposal dialog box appears, as shown in Figure 60 on page 152.

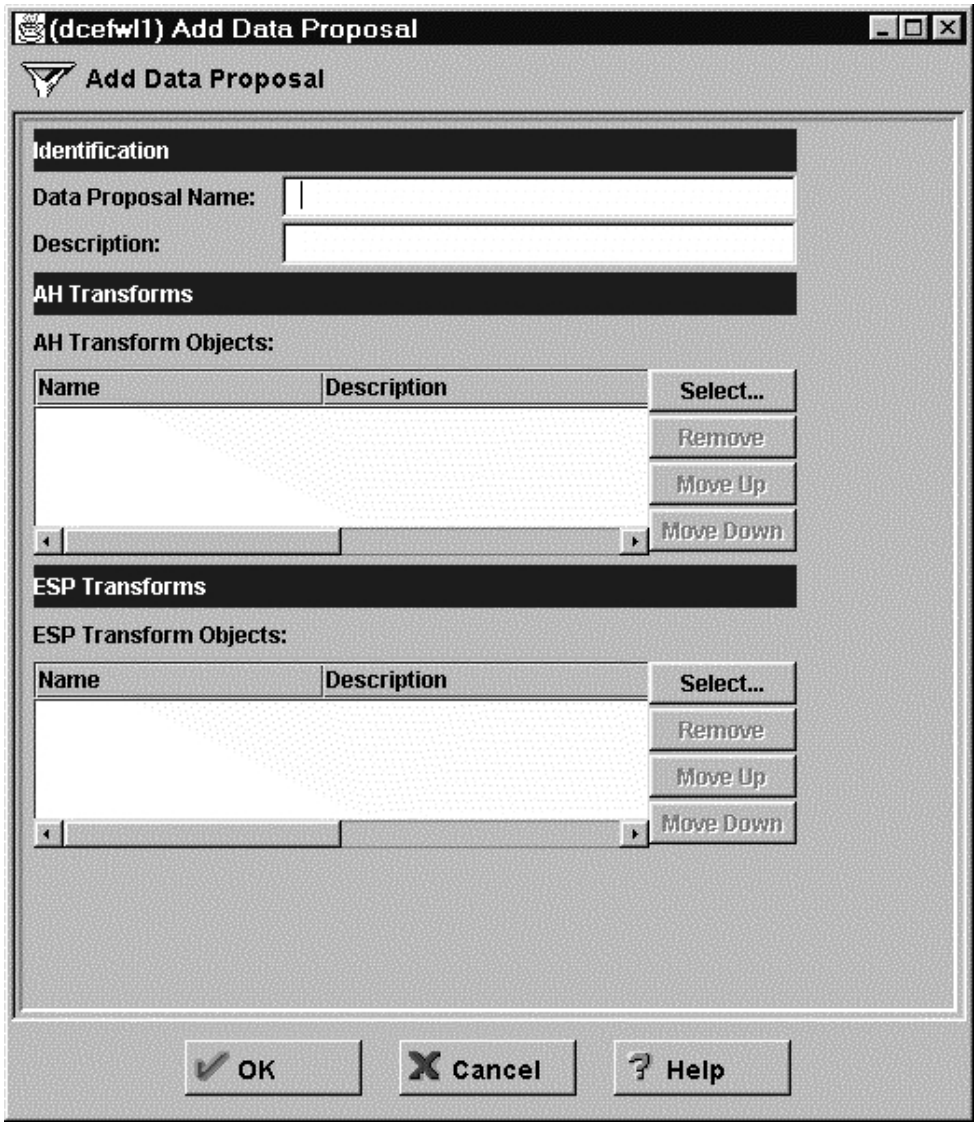


Figure 60. Add Data Proposal

1. Fill in the Data Proposal Name.
2. Fill in the description. This field is optional.
3. To choose the AH transforms for this data proposal, click Select and choose the type of authentication protection mechanism you want to perform between the two data endpoints.

   You can reorder the list by selecting an AH transform and clicking Move Up or Move Down.

   You can remove an AH transform by selecting it and clicking Remove.

4. To choose the ESP transforms for this data proposal, click Select and choose the type of encryption protection mechanism (optional authentication) you want to perform between the two data endpoints.

   You can reorder the list by selecting an ESP transform and clicking Move Up or Move Down.

   You can remove an ESP transform by selecting it and clicking Remove.

5. Click OK after filling in all of the required fields to create the Data Proposal object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the Data Proposal Administration panel.

Changing a Data Proposal Configuration Entry

To modify a data proposal that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Data Proposal dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying a Data Proposal Configuration Entry

To copy a data proposal:

1. Select the entry and click the Copy button. The Copy Data Proposal dialog box appears.
2. Modify the appropriate fields, enter a unique name for the proposal and Click OK to create the proposal.

Deleting a Data Proposal Configuration Entry

To delete a data proposal that you have created:

1. Select the data proposal entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the data proposal is reference by a data policy, the proposal cannot be deleted until it is removed from the data policy.

Defining a Data Policy

Data Policy objects define information required when negotiating keys for data exchanges.

After you have defined your data proposals, you need to add the proposals(s) to a data policy. Select Data policy to display the Data Policy Administration panel. Double click on NEW. The Add Data Policy dialog box appears, as shown in Figure 61 on page 154.

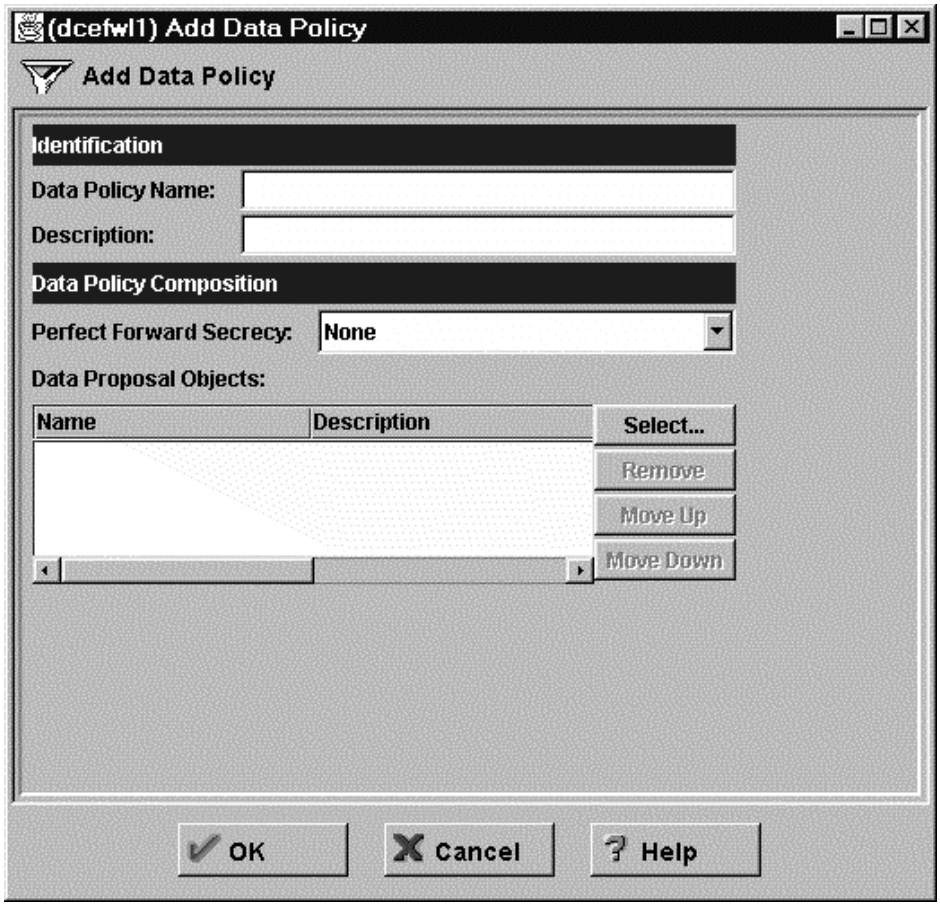


Figure 61. Add Data Policy

1. Fill in the Data Policy Name.
2. Fill in the description. This field is optional.
3. Select whether perfect forward secrecy should be used or not. Perfect forward secrecy means that the key used to protect transmission of data and the material used to derive the key are not used to derive additional keys. If perfect forward secrecy is requested, the appropriate Diffie-Hellman group needs to be selected.

   None No perfect forward secrecy

   Group1 modular exponentiation group with a 768-bit modulus

   Group2 modular exponentiation group with a 1024-bit modulus
4. To choose the data proposals for this data policy, click Select and choose the type of data proposals.

   You can reorder the list by selecting a data proposal and clicking Move Up or Move Down.

   You can remove a data proposal by selecting it and clicking Remove.
5. Click OK after filling in all of the required fields to create the Data Policy object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the Data Policy Administration panel.

Changing a Data Policy Configuration Entry

To modify a data policy that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Data Policy dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying a Data Policy Configuration Entry

To copy a data policy:

1. Select the entry and click the Copy button. The Copy Data Policy dialog box appears.
2. Modify the appropriate fields, enter a unique name for the policy and Click OK to create the policy.

Deleting a Data Policy Configuration Entry

To delete a data policy that you have created:

1. Select the data policy entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the data policy is reference by a dynamic tunnel policy, the data policy cannot be deleted until it is removed from the dynamic tunnel policy.

Dynamic Tunnel Policy

Dynamic Tunnel Policy objects define generic information relative to a set of dynamic VPNs. These objects are then associated with an anchor filter rule to govern traffic through the dynamic VPN.

Select VPN Connection Templates from the configuration client navigation tree. Double click the file folder icon to expand the view.

Defining Dynamic Tunnel Policy

Select Dynamic Tunnel Policy to display the Dynamic Tunnel Policy Administration panel. Double click on NEW. The Add Dynamic Tunnel Policy dialog box appears, as shown in Figure 62 on page 156.

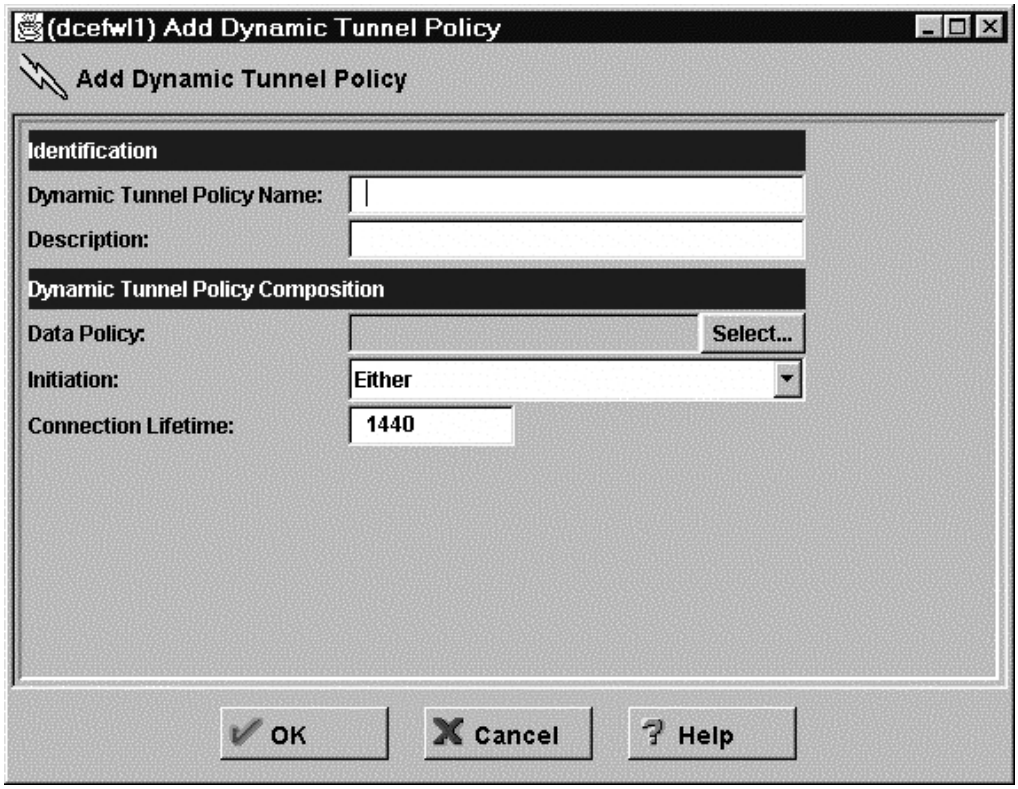


Figure 62. Add Dynamic Tunnel Policy

1. Fill in the Dynamic Tunnel Policy Name.
2. Fill in the description. This field is optional.
3. To choose the data management policy to be used for this tunnel, click Select and choose the policy. The policy defines how the data being transmitted through the tunnel should be protected.
4. Select which systems will be allowed to initiate dynamic VPN connections using this dynamic tunnel policy object.

   Local only the local system may initiate dynamic VPN connections

   Remote only the remote system may initiate dynamic VPN connections

   Either either the local or remote system may initiate dynamic VPN connections
5. Enter the connection lifetime value in minutes. This is the number of minutes that the connection should remain active. The valid values are 0 to 525600 minutes (365 days). A value of 0 means that the connection will not expire.
6. Click OK after filling in all of the required fields to create the Dynamic Tunnel Policy object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the Dynamic Tunnel Policy Administration panel.

Changing a Dynamic Tunnel Policy Configuration Entry

To modify a dynamic tunnel policy that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Dynamic Tunnel Policy dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying a Dynamic Tunnel Policy Configuration Entry

To copy a dynamic tunnel policy:

1. Select the entry and click the Copy button. The Copy Dynamic Tunnel Policy dialog box appears.
2. Modify the appropriate fields, enter a unique name for the policy and Click OK to create the policy.

Deleting a Dynamic Tunnel Policy Configuration Entry

To delete a dynamic tunnel policy that you have created:

1. Select the dynamic tunnel policy entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the dynamic tunnel policy is reference by an anchor filter rule, the dynamic tunnel policy cannot be deleted until it is removed from the anchor filter rule.

Defining Key Server Information

Key servers negotiate security associations using the Internet Security Association Key Management Protocol (ISAKMP). A key server object defines an authentication identity by which a key server is known. A key server may have multiple identities. A key server group consists of a key policy as well as a local key server and a group of remote key servers which can negotiate with the local key server.

Select VPN Key Servers from the configuration client navigation tree. Double click the file folder icon to expand the view. The folder consists of key server, key server group and authentication entries.

Defining a Key Ring

A Key ring is used to store certificates that will be used by the local key server. The key ring is identified through a RACF userid and name. Double click on the Authentication folder icon to expand the view. Select Key Ring to display the Configure Key Ring Parameters dialog box, as shown in Figure 63.

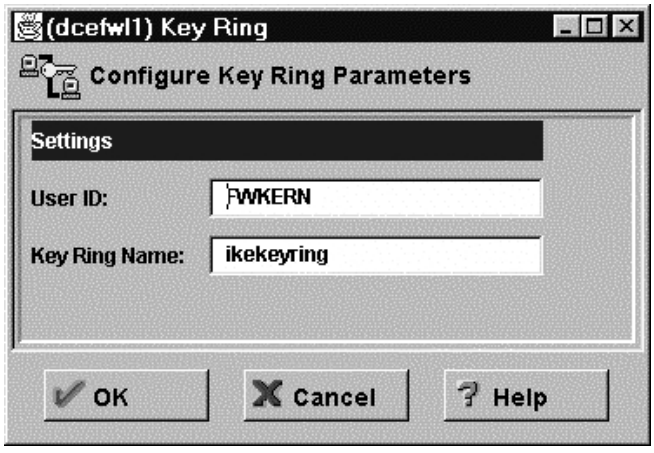


Figure 63. Configure Key Ring Parameters

1. Fill in the RACF userid that owns the key ring
2. Fill in the name of the key ring. The name cannot contain an asterisk ('*'). The key ring name is case sensitive.

3. Click OK after filling in all of the required fields to create the key ring object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will need to return to the Configure Key Ring Parameters to make any changes.

Defining a Certificate Authority

Certificate authority objects contain the RACDCERT label needed to identify a certificate authority to be used when requesting certificates of negotiating peers.

Double click on the Authentication folder icon to expand the view. Select Certificate Authority to display the Certificate Authority Administration panel. Double click on NEW. The Add Certificate Authority dialog box appears, as shown in Figure 64.

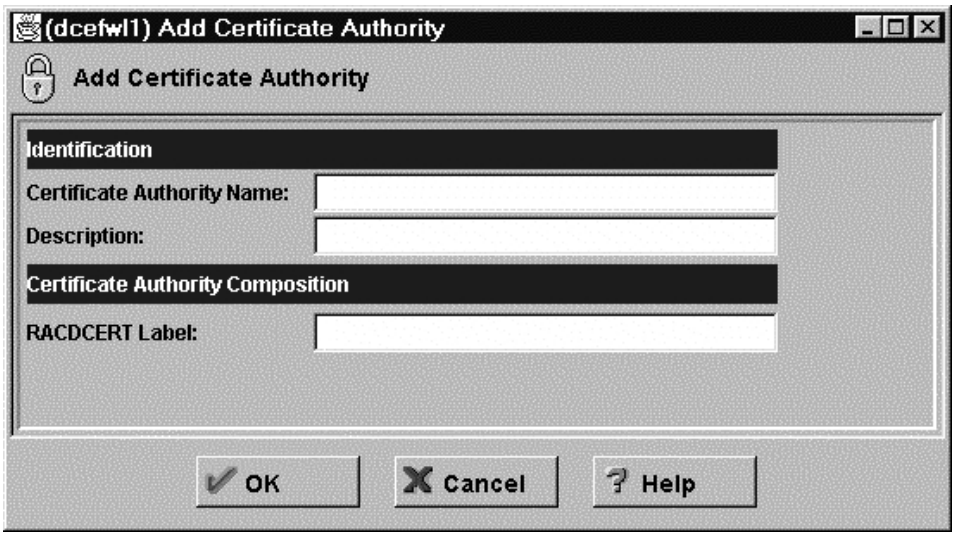


Figure 64. Add Certificate Authority

1. Fill in the Certificate Authority Name.
2. Fill in the description. This field is optional.
3. Fill in the RACDCERT label which identifies the Certificate Authority to RACF.
4. Click OK after filling in all of the required fields to create the certificate authority object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the Certificate Authority Administration panel.

Changing a Certificate Authority Configuration Entry

To modify a certificate authority that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Certificate Authority dialog box appears.
2. Modify the appropriate fields and Click OK to apply the changes.

Copying a Certificate Authority Configuration Entry

To copy a certificate authority:

1. Select the entry and click the Copy button. The Copy Certificate Authority dialog box appears.
2. Modify the appropriate fields, enter a unique name for the certificate authority and Click OK to create the object.

Deleting a Certificate Authority Configuration Entry

To delete a certificate authority that you have created:

1. Select the certificate authority entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the certificate authority is referenced by an authentication information object, the certificate authority cannot be deleted until it is removed from the authentication information object.

Defining a Key Server

Key Server objects define information about key servers. Key servers negotiate security associations using the Internet Security Association Key Management Protocol (ISAKMP). A key server object defines an authentication identity and an optional IP address or host name by which a key server is known.

Select Key Server to display the Key Server Administration panel. Double click on NEW. The Add Key Server dialog box appears, as shown in Figure 65.

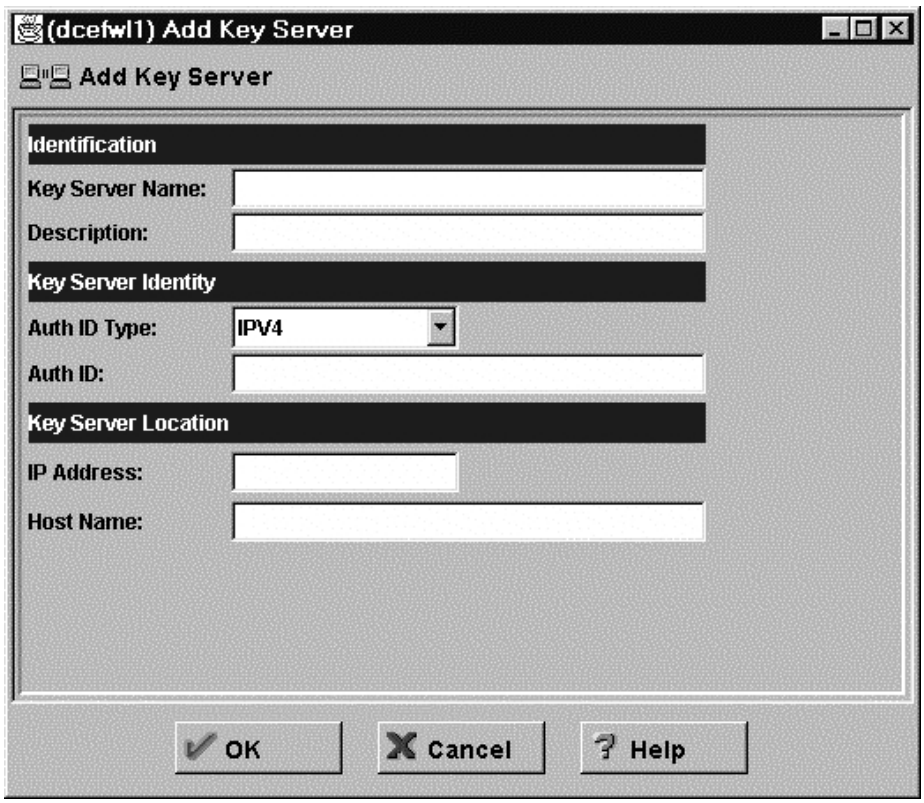


Figure 65. Add Key Server

1. Fill in the Key Server Name.

2. Fill in the description. This field is optional.
3. Select the authentication ID type for the key server object. Click the Auth ID Type arrow to see the ID types that can be used.

   IPV4 an IP address in dotted-decimal format. For example, 1.2.3.4

   FQDN a fully qualified domain name. For example, vnet.ibm.com

   USER@FQDN a user at a fully qualified domain name. The user name cannot contain a blank. For example, ibm@vnet.ibm.com

   X500_DN an X.500 distinguished name (DN). For example, CN=Alicia Reger, OU=\" 1st Grade\",OU=Ms. Jone's class, O=St. Anthony's Elementary,ST=NY,C=usa
4. Enter the Auth ID used to identify the key server in the format specified in the Auth ID Type field.
5. When defining a key server for this system, either the server's IP address or host name needs to be specified. If you choose to identify the server using its IP address, enter the IP address in IPV4 format in the IP Address field. If you choose to identify the server using its host name, enter the host name in the Host Name field. The host name must resolve to a single IP address through the Domain Name Server. IP address and host name cannot both be specified to identify the local key server.

   When defining a key server for a remote system which will initiate requests to this system, either the server's IP address or host name needs to be specified. If you choose to identify the server using its IP address, enter the IP address in IPV4 format in the IP Address field. If you choose to identify the server using its host name, enter the host name in the Host Name field. The host name must resolve to a single IP address through the Domain Name Server. IP address and host name cannot both be specified to identify the local key server.
6. Click OK after filling in all of the required fields to create the Key Server object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the Key Server Administration panel.

Changing a Key Server Configuration Entry

To modify a key server that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Key Server dialog box appears.
2. Modify the appropriate fields and Click OK to apply the changes.

Copying a Key Server Configuration Entry

To copy a key server:

1. Select the entry and click the Copy button. The Copy Key Server dialog box appears.
2. Modify the appropriate fields, enter a unique name for the key server and Click OK to create the object.

Deleting a Key Server Configuration Entry

To delete a key server that you have created:

1. Select the key server entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the key server is referenced by a dynamic VPN connection or key server group or authentication information object, the key server cannot be deleted until it is removed from the referencing object.

Defining Authentication Information

Authentication information objects define information required to authenticate the identity of the communicating peer. This information includes the remote key server object and the pre-shared key and/or certificate authority to be used by the remote key server.

Double click on the Authentication folder icon to expand the view. Select Authentication Information to display the Authentication Information Administration panel. Double click on NEW. The Add Authentication Information dialog box appears, as shown in Figure 66 on page 161.

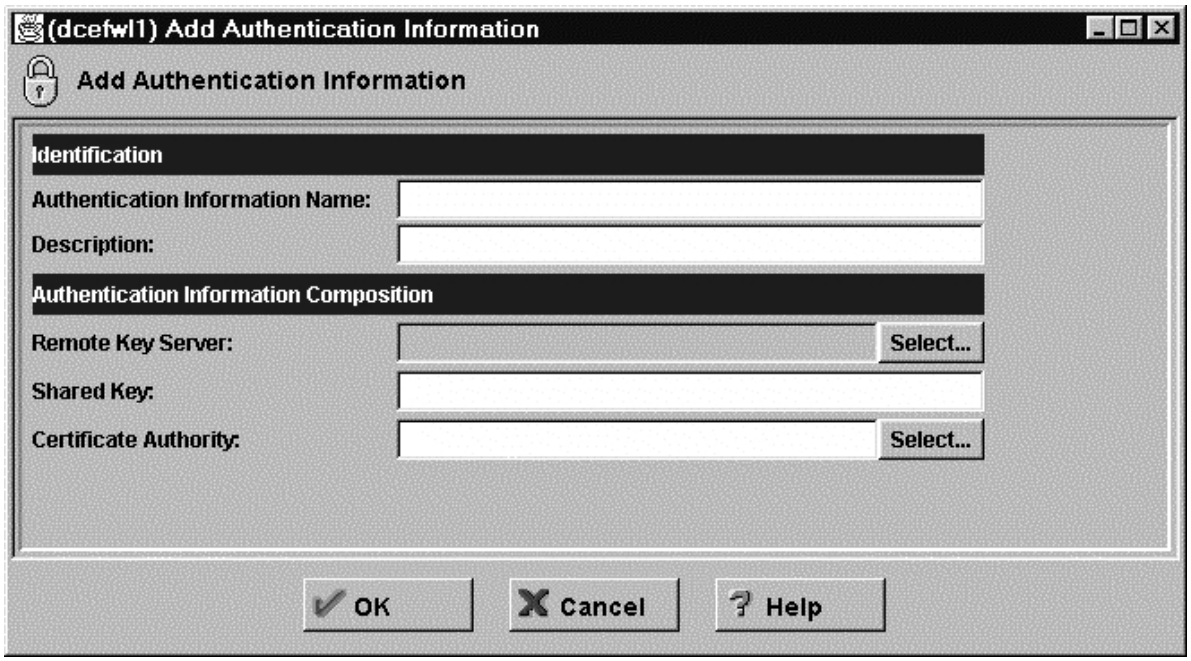


Figure 66. Add Authentication Information

1. Fill in the Authentication Information Name.
2. Fill in the description. This field is optional.
3. To choose the Remote Key Server, click Select and choose the remote key server for which the authentication information will be used.
4. Fill in the Shared Key value if authentication is to be performed using a pre-shared key. The shared key must be entered as byte data and must consist of an even number of hexadecimal digits.
5. If authentication is to be performed using certificates, choose the certificate authority by clicking Select and choosing a certificate authority from the list.
6. Click OK after filling in all of the required fields to create the Authentication Information object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the Authentication Information Administration panel.

Changing an Authentication Information Configuration Entry

To modify a authentication information object that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Authentication Information dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying an Authentication Information Configuration Entry

To copy an authentication information object:

1. Select the entry and click the Copy button. The Copy Authentication Information dialog box appears.
2. Modify the appropriate fields, enter a unique name for the authentication information object and Click OK to create the object.

Deleting an Authentication Information Configuration Entry

To delete an authentication information object that you have created:

1. Select the authentication information entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted.

Defining a Key Server Group

Key Server Group objects define the association of a local key server with a list of remote key servers. This establishes which combinations of local and remote key servers can negotiate key management information and the key policy which will be used during these negotiations. Key Server Groups are an ordered list. Searches will find the first instance of the key server found in all key server group objects. Therefore, the order of the groups is very important.

Select Key Server Group to display the Key Server Group Administration panel. Double click on NEW. The Add Key Server Group dialog box appears, as shown in Figure 67 on page 163.

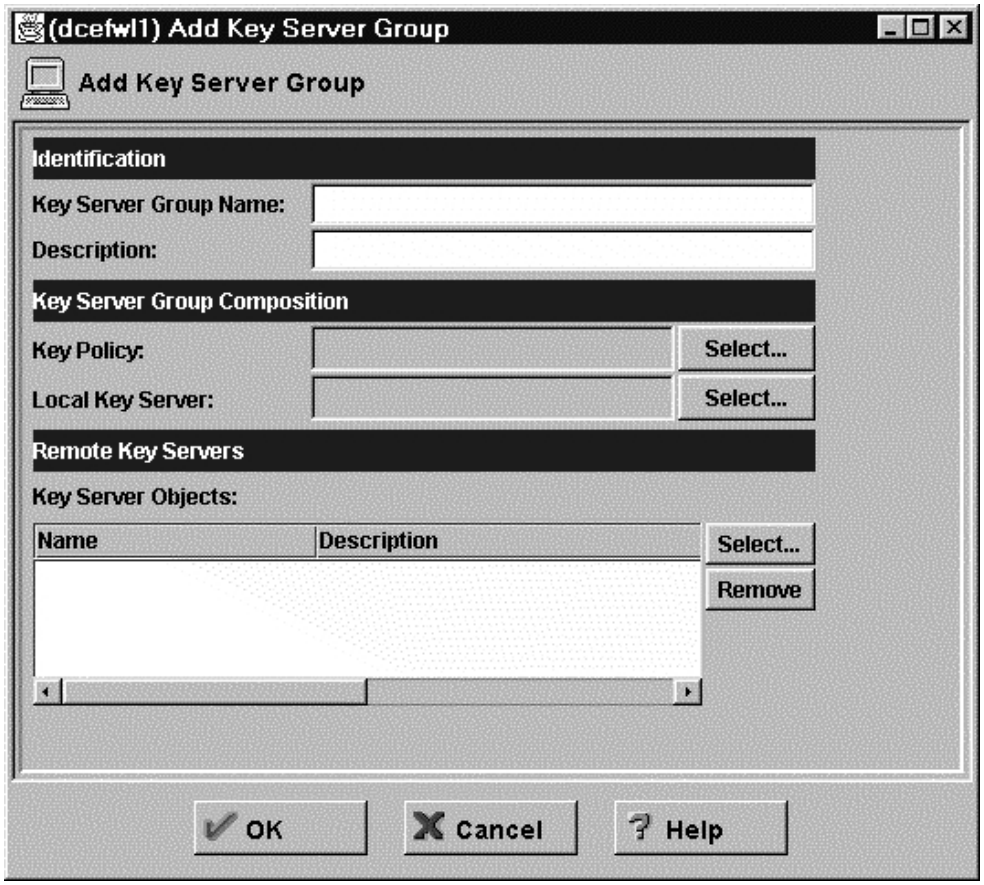


Figure 67. Add Key Server Group

1. Fill in the Key Server Group Name.
2. Fill in the description. This field is optional.
3. To choose the key policy to be used by this group when establishing the key management security association between the local key server and a member of the remote key server list, click Select and choose the key policy.
4. To choose the local key server, click Select an choose the key server to represent the local key server for this group object.
5. To choose the Remote Key Servers with which the specified local key server can negotiate, click Select and choose the Key Servers,

   The order of the remote key servers does not manner.

   You can remove a remote key server by selecting it and clicking Remove.
6. Click OK after filling in all of the required fields to create the key server group object. You may change any of the fields prior to clicking the OK button.

   After clicking the OK button, you will be returned to the Key Server Group Administration panel.

Ordering Key Server Group Objects

When determining whether two key servers are allowed to negotiate and the key policy to be used during negotiation, the list of key server groups are searched in a top down fashion. When a match is found, the search is completed. If no match is found, the negotiation fails. To order the list,

1. Select the to-be-moved item.
2. Hold down the Ctrl key and select the item above where you want the to-be-moved item to be placed. You should now have exactly 2 items selected and the Move button should be enabled. If you wish for the item to be moved to the top of the list, you can select the NEW item as the location item. Click Move to place the item in its new location.

Changing a Key Server Group Configuration Entry

To modify a authentication information object that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Key Server Group dialog box appears.
2. Modify the appropriate field and Click OK to apply the changes.

Copying a Key Server Group Configuration Entry

To copy a key server group object:

1. Select the entry and click the Copy button. The Copy Key Server Group dialog box appears.
2. Modify the appropriate fields, enter a unique name for the key server group object and Click OK to create the object.

Deleting a Key Server Group Configuration Entry

To delete a key server group object that you have created:

1. Select the key server group entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted. If the key server group object is referenced dynamic VPN connection, the key server group cannot be deleted until it is removed from the dynamic VPN connection object.

Defining Anchor Rules

As mentioned in previous chapters, filter rules are an ordered set of rules that govern what type of IP traffic is permitted or denied to flow through a Firewall TCP/IP stack. An anchor rule is a place holder for another type of filter rule called a dynamic filter rule. For information on defining anchor rules, see “Creating Filter Rules” on page 115. Anchor filter rules are similar to permit and deny IP filter rules except that the action parameter is set to anchor. As with permit and deny IP filter rules, anchor rules must be added to services which in turn are added to connections and activated through the **fwfilter** command.

Defining an On-demand VPN Connection

The On-demand object describes the attributes applied to a dynamic on-demand connection. The on-demand object describes what level of granularity to apply to the connection being established.

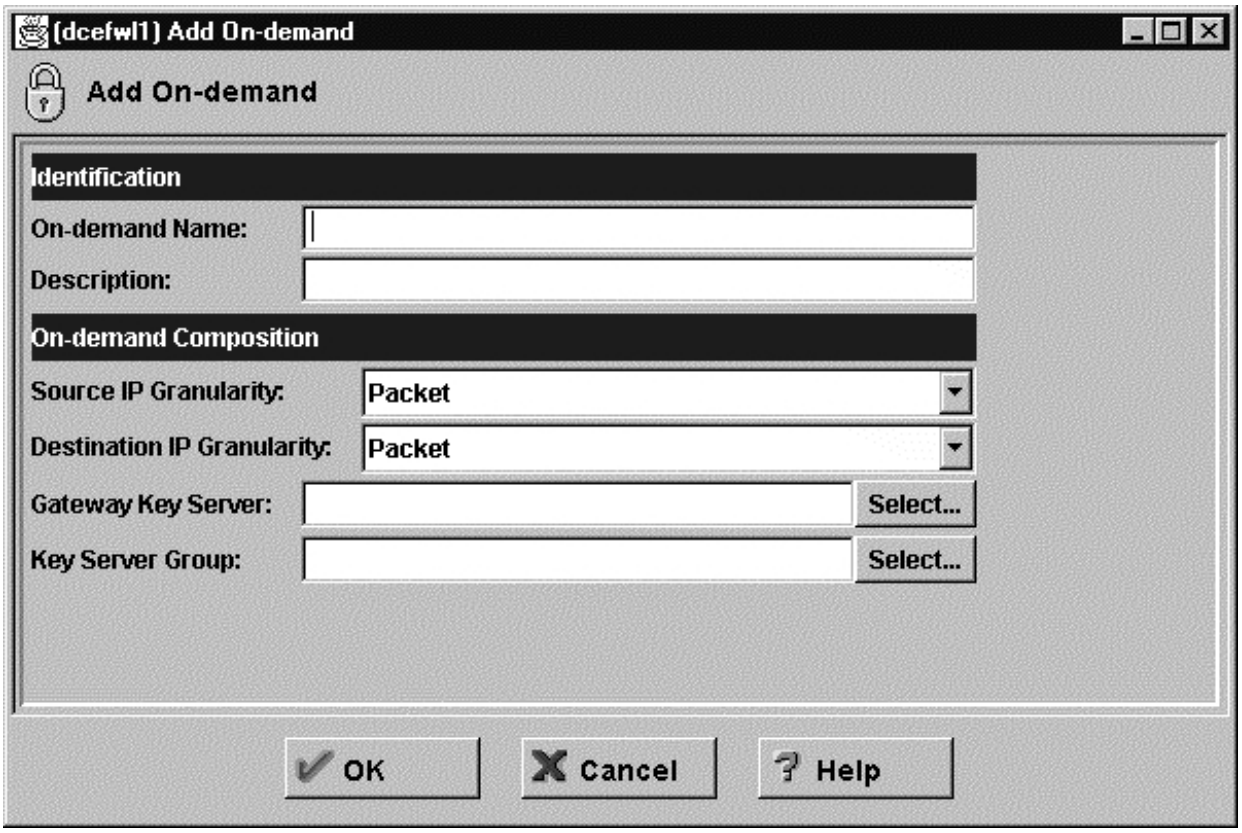


Figure 68. Add On-demand

Add an on-demand connection by completing the following information:

1. Enter the name of the on-demand object. The maximum length is 100 characters.
2. Enter a description of the on-demand object. Maximum length of this field is 100 characters, and it is optional.
3. Choose the source IP granularity. This is the granularity that will be applied to the source IP address in the dynamic connection being established. The default source granularity is **packet**.
4. Choose the destination IP granularity. This is the granularity that will be applied to the destination IP address in the dynamic connection being established. The default destination granularity is **packet**.
5. Select the gateway key server that is remote to the system being configured. It is the encryption/decryption point, or tunnel endpoint. This parameter is optional. If the parameter is not specified, the destination address in the IP packet is used as the gateway key server.
6. Select the key server group used for the connection.
7. Click the OK button after making your selections.

Defining Locally Activated Dynamic Connections

This action involves the defining of the dynamic connection and the manual activation of the connection.

Defining a Dynamic VPN connection

The Dynamic VPN Connection object defines information that is used to activate a specific connection between data endpoints. This information includes the source and destination objects, source and destination ports, and protocol supported for this connection along with the remote key server and key server group objects and an indicator of whether to autoactivate the connection.

Select VPN Connection Setup from the configuration client navigation tree to display the Dynamic VPN connection Administration panel. Double click on NEW. The Add Dynamic VPN Connection dialog box appears, as shown in Figure 69.

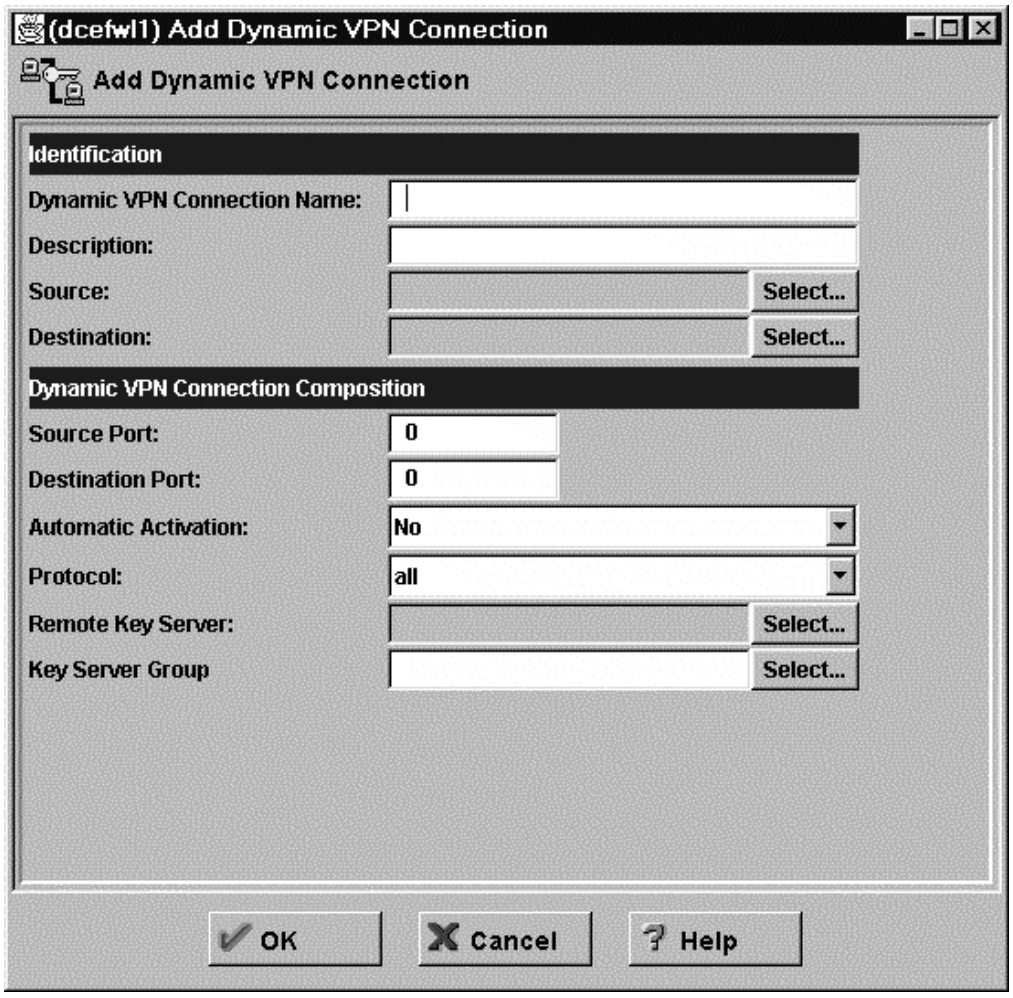


Figure 69. Dynamic VPN Connection

1. Fill in the Dynamic VPN Connection Name.
2. Fill in the description. This field is optional.
3. To choose the network object to use as the source IP address(es) for this connection, click Select and choose the network object from the list.
4. To choose the network object to use as the destination IP address(es) for this connection, click Select and choose the network object from the list.
5. Enter the port number to use as the source port for this connection. A value of 0 means that all ports will be used.

6. Enter the port number to use as the destination port for this connection. A value of 0 means that all ports will be used.
7. Select the activation mode for the dynamic VPN connection by clicking the Automatic Activation arrow and selecting a choice.

   No Do not autoactivate the connection when either the Firewall TCP/IP stack or ISAKMP daemon is initialized.

   Yes Start the connection when either the Firewall TCP/IP stack or ISAKMP daemon is initialized.
8. Select the internet protocol type allowed to flow in a protected mode through this connection by clicking the Protocol arrow and choosing the type from the list.

   all Any specified protocol

   tcp transmission control protocol (TCP)

   udp user datagram protocol (UDP)

   icmp internet control message protocol (ICMP)

   ospf open shortest path first protocol (ospf). This protocol uses the source port number for the ospf record type.

   ipip IP-in-IP protocol (IPIP). This protocol requires the source and destination port numbers to be 0.

   esp Encapsulating security payload protocol (ESP). This protocol datagram is used for sending encrypted IP packets.

   ah Authentication header protocol (AH). This protocol datagram is used for sending IP packets, which have an associated authentication token.
9. To choose the remote key server to be negotiated with for this dynamic connection, click Select and choose a key server from the list. Once a key server is chosen, the Key Server Group (if empty) will be filled in with the first key server group for which the remote key server is a member.

   To change the Key Server Group, click Select and choose from the list of key server groups. The remote key server must be member of the key server group.
10. Click OK after filling in all of the required fields to create the dynamic VPN connection object. You may change any of the fields prior to clicking the OK button.

    After clicking the OK button, you will be returned to the Dynamic VPN Connection Administration panel.

Changing a Dynamic VPN Connection Configuration Entry

To modify a dynamic VPN connection that you have created:

1. Double click on an existing entry or select the entry and click the Open button. The Modify Dynamic VPN Connection dialog box appears.
2. Modify the appropriate fields and Click OK to apply the changes.

Copying a Dynamic VPN Connection Configuration Entry

To copy a dynamic VPN connection:

1. Select the entry and click the Copy button. The Copy Dynamic VPN Connection dialog box appears.
2. Modify the appropriate fields, enter a unique name for the dynamic VPN connection and Click OK to create the object.

Deleting a Dynamic VPN Connection Configuration Entry

To delete a dynamic VPN connection that you have created:

1. Select the dynamic VPN connection entry and click the Delete button.

   You will be prompted with a confirmation message prior to the definition being deleted.

Activating a Dynamic VPN connection

Activation of a dynamic VPN connection is accomplished through the Dynamic VPN Connection Administration Panel.

Select VPN Connection Setup from the configuration client navigation tree to display the Dynamic VPN Connection Administration panel. Select the connection to be activated and click the Activate button.

Managing Activate Dynamic VPN Connections

Once a dynamic VPN connection has been negotiated and activated between two endpoints, the Dynamic VPN Connection Activation Administration panel allows you to manage the activate connections.

Select VPN Connection Activation from the configuration client navigation tree to display the Dynamic VPN Connection Activation Administration panel, as shown in Figure 70.

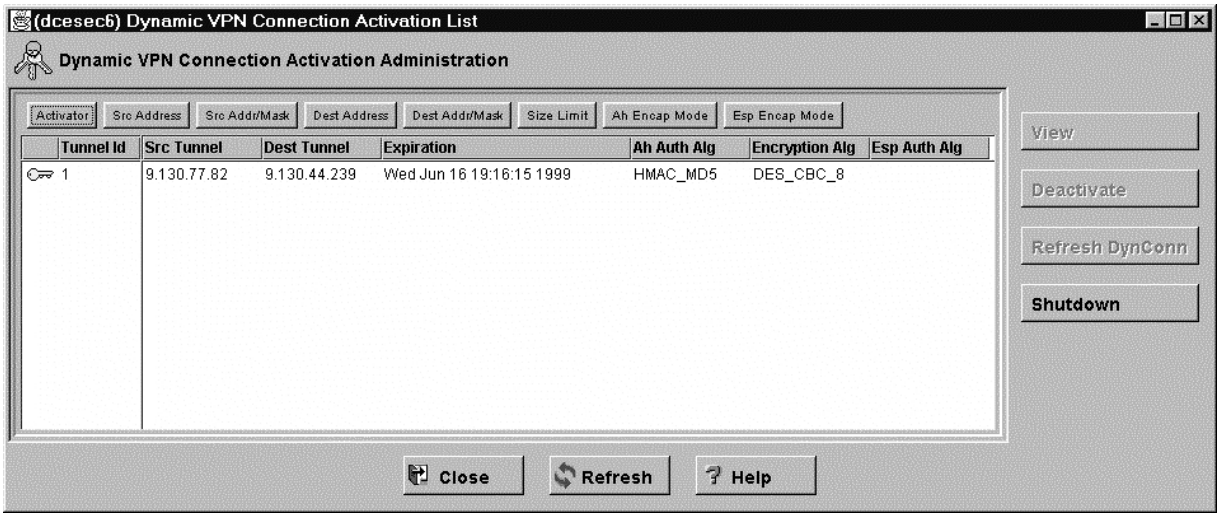


Figure 70. Dynamic VPN Connection Activation

This panel allows you to deactivate, view and refresh selected connections as well as shutdown all active connections.

Configuring Network Address Translation

1. From the configuration client navigation tree, double-click the Address Translation file folder icon to expand the view. Double click the NAT file folder icon to expand the view.
2. Select NAT Setup to configure the Network Address Translation module.

   The Network Address Translation List appears, as shown in Figure 71 on page 169.

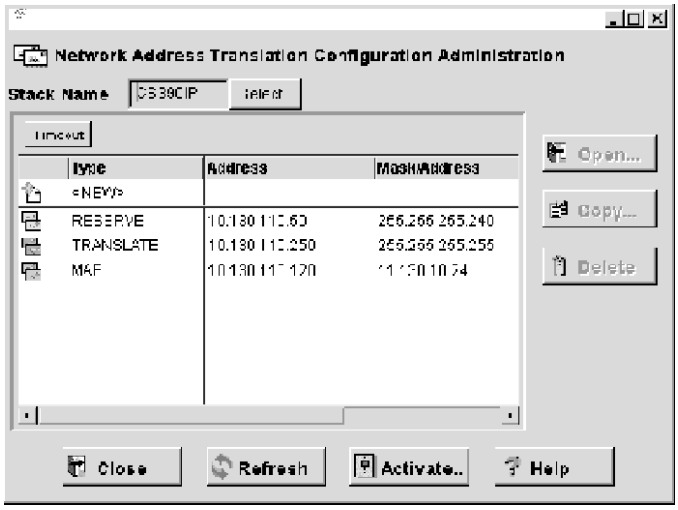


Figure 71. Network Address Translation List

3. Select the TCP/IP stack name from the drop down box.
4. Network Address Translation entries contained in the NAT configuration file are displayed on this dialog box. You can also add, change, or delete NAT entries.

Adding a NAT Entry

1. Select NEW from the Network Address Translation List and click Open to add new entries to the NAT configuration file.

   The Add NAT dialog box appears, as shown in Figure 72.

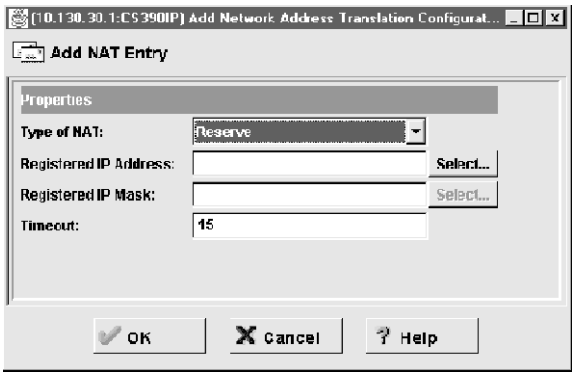


Figure 72. Add NAT Configuration

2. From the Add NAT dialog box, click the arrow in the Type of NAT field and select from the following:
   - **Reserve** Registered Network Address: Adds the IP addresses specified to the registered address pool.

- **Translate** Secured Network Address: Specifies a range of secure IP addresses that require network address translation.
- **Exclude** Secured Network Address: Specifies a range of secure IP addresses that should be excluded from network address translation.
- **Map** Secured Network Address: Defines a one-to-one secure-to-registered IP address static translation.

Reserve Registered Network Address: If you selected Reserve from the Add NAT screen, enter the following values:

Registered IP Address

Specify a dotted-decimal IP address that identifies a range of registered IP addresses to be added to the registered address pool.

Choose a network object by clicking Select to get the Select Network Object dialog box. Select a network object and click OK. The network object is added to the Network Object field on the Add NAT Configuration dialog box.

Registered IP Address Mask

Refer to Chapter 15, "Network Address Translation" on page 103 for information on specifying an IP address mask.

Timeout Value

Enter the number of minutes an address translation can remain idle before NAT can free the registered IP address. This timeout value only applies to the address translation that uses a registered IP address within the range of IP addresses specified by this entry.

The default is 15 minutes and is the minimum value allowed. The range of values is 15 through 45.

Translate Secured Network Address: If you selected Translate from the Add NAT screen, enter the following values:

Secured IP Address

Specify a dotted-decimal IP address that identifies a range of secure IP addresses that require network address translation.

Choose a network object by clicking Select to get the Select Network Object dialog box. Select a network object and click OK. The network object is added to the Network Object field on the Add NAT Configuration dialog box.

Secured IP Address Mask

Refer to Chapter 15, "Network Address Translation" on page 103 for information on specifying an IP address mask.

Exclude Secured Network Address: If you selected Exclude from the Add NAT dialog screen, enter the following values:

Secured IP Address

Specify a dotted-decimal IP address that identifies a range of secure IP addresses that should be excluded from network address translation.

Choose a network object by clicking Select to get the Select Network Object dialog box. Select a network object and click OK. The network object is added to the Network Object field on the Add NAT Configuration dialog box.

Secured IP Address Mask

Refer to Chapter 15, “Network Address Translation” on page 103 for information on specifying an IP address mask.

Map Secured Network Address: If you selected Map from the Add NAT Configuration dialog box, enter the following values:

Value **Description**

Secured IP Address

A dotted-decimal IP address that should be translated into a specified registered IP address.

You can choose a network object by clicking Select to get the Select Network Object dialog box. Select a network object and click OK. The network object is added to the Network Object field on the Add NAT Configuration dialog box.

Registered IP Address field

A dotted-decimal IP address into which a specified secure IP address should be translated.

You can choose a network object by clicking Select to get the Select Network Object dialog box. Select a network object and click OK. The network object is added to the Network Object field on the Add NAT Configuration dialog box.

Changing NAT Entry

1. Select an existing NAT entry from the NAT Configuration dialog box.
2. Click Open to change Network Translation entries in the NAT configuration file.

Deleting NAT Entry

1. Select an existing NAT entry from the NAT Configuration dialog box and click Delete to remove a Network Translation entry from the NAT configuration file.

   A confirmation dialog box appears.
2. Select Yes or No.

Activating NAT

1. From the configuration client navigation tree, double-click the Address Translation file folder icon to expand the view. Double click the NAT file folder icon to expand the view.
2. Select NAT Activation and a dialog box similar to the one shown in Figure 73 on page 172 appears.

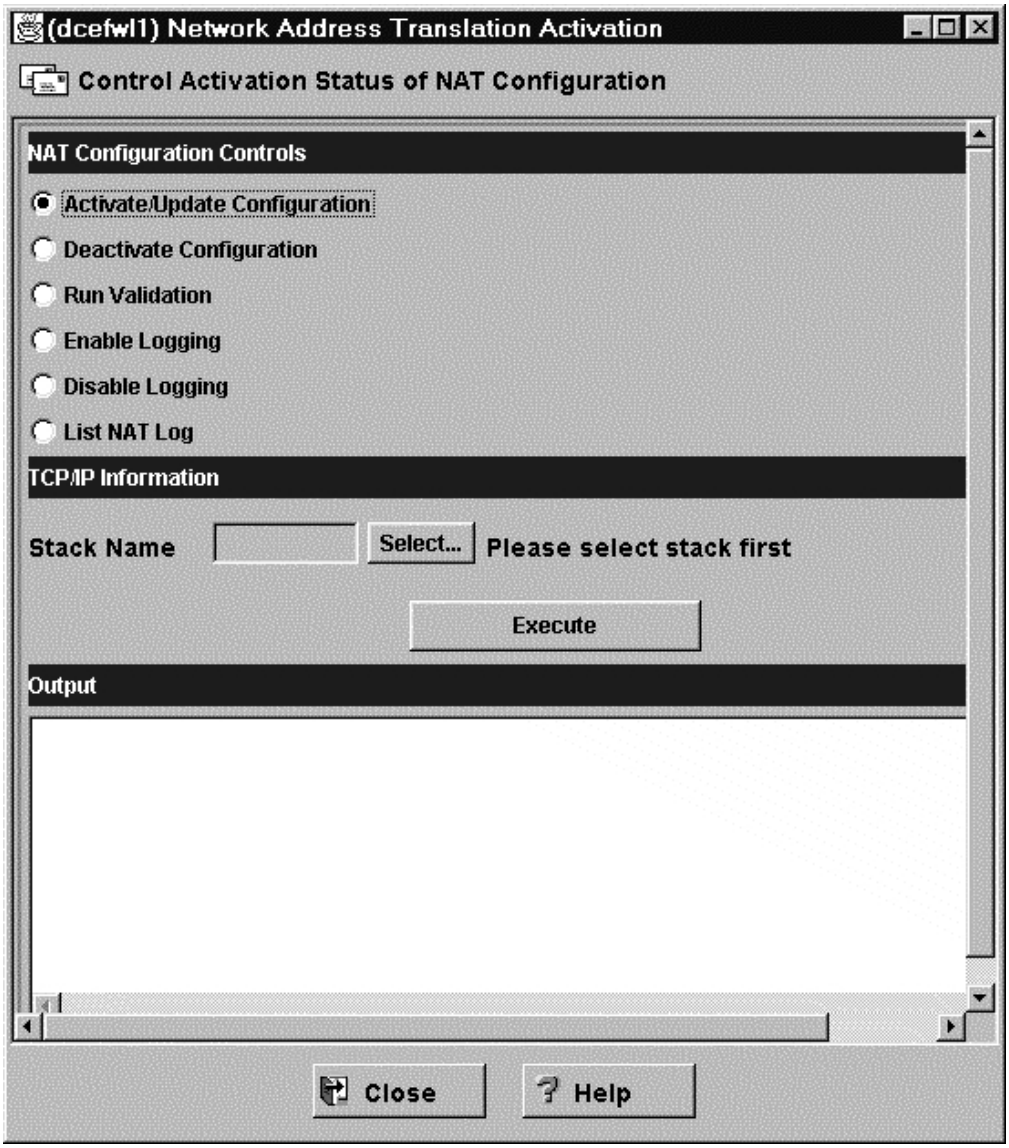


Figure 73. NAT Activation

3. Select the TCP/IP stack name from the drop down box.
4. You can select any of the following and then click Execute:
   - Validate network address translation entries contained in a specified NAT configuration file.
   - Activate/Update the configuration to display Network Address Translation entries currently used by the NAT module.
   - Deactivate NAT to disable network address translation.
   - Enable Logging to enable network address translation logging.
   - Disable Logging to disable network address translation logging.
   - List NAT Log.

Logging takes place at the initial mapping phase and is not dynamic. It does not show each actual mapping in the kernel upon request. You must enable logging before activation. If not, you could lose log entries. If you refresh the log or erase it, you will lose log entries.

Creating Filter Rules for NAT

After you have completed the NAT configuration, you have to create the filter rules for the connections that are going to use NAT. Review Chapter 8, “Controlling Traffic Through the Firewall with Filters” on page 51 and use the predefined services that are for direct connections. Examples of predefined services that are for direct connections are:

- HTTP direct out
- Telnet direct out

See “Building Connections Using Predefined Services” on page 125 for more information.

If you want a service to come directly into your network, you will have to create one. See “Creating a Service” on page 120 for information on how to do this.

Note: Filter rules are not created using the translated addresses; they are created using the original addresses.

Configuring RealAudio

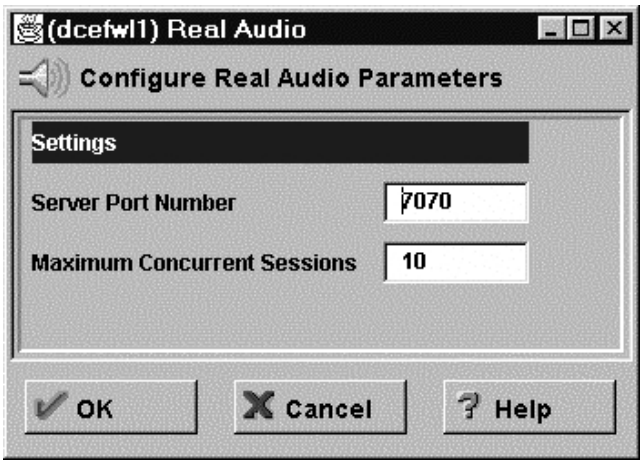


Figure 74. Configuring RealAudio

Select System Administration from the configuration client navigation tree. Double click the file folder icon to expand the view. Select Real Audio. The Real Audio dialog box appears:

1. Fill in the server port number for RealAudio. The RealAudio default server port number is 7070. However, you can reconfigure it to any valid TCP port number.
2. Fill in the maximum concurrent sessions allowed for RealAudio. The default is 10. Valid entries are 0 through 2147483647. The sessions number is based per stack, each stack having the same number. With 3 stacks, each with 10 sessions maximum, you would have 30 sessions going (10 per stack).
3. Click OK.

Part 4. Commands and Messages

© Copyright IBM Corp. 1997, 2000

Chapter 18. Commands

This section describes commands used to start the firewall and configure firewall services.

Note: With the exception of **fwkern**, which is issued from the OS/390 operator console, these commands are valid in UNIX services only; TSO is not supported. In addition, all commands except **fwkern** must be issued either by a **superuser** or a user in the **fwgrp** group defined by the External Security Manager.

Command Syntax

To see the correct syntax of a command, enter the command without parameters on the Unix System Services command line.

Syntax Notes

Syntax notes that apply to specific commands are listed within the syntax description of the command. General syntax notes that apply to all commands are listed here:

1. Keywords and options may be specified in any order (except for names) and are not case-sensitive, including values for **protocol, srcopcode, destopcode, interface, routing, direction** and **log**.
2. Blanks are not allowed in a parameter unless in double quotes within the text.
3. The **change** parameter only changes the values specified in a command; attributes not specified remain unchanged. To **delete** an optional value, specify a blank value for that attribute.
4. The **stack** parameter is an optional parameter when there is only one firewall stack. Where more than one firewall stack exists, the **stack** parameter is required.
5. The **cmd=delete force=yes** option is backward compatible. If the object referenced is a V2 R8 object, you will receive a message that the object cannot be deleted because it is in use by an object that does not allow force.

Common Parameters

NAME and DESCRIPTION: The **name** and **description** parameters can each be up to 100 characters long, including blanks. To include blanks in a value, surround it with double quotes. For example,

"`This is a name with blanks`**"**

A vertical bar (|) cannot be used within a name or description. Some commands have a parameter whose value is specified as either a name or an ID.

NAME or ID: Many commands have a **name | id** parameter. The commands use the following logic to find an object when you specify a value which could either be a **name** or **id**:

- If value is numeric, then the configuration file is searched for this ID. If found, that object is used.
- The configuration file is searched for this name. If found, that object is used.
- If no matches are found, an error message is returned.

If a name is numeric and happens to match another object's ID, then the object with the matching ID is found and used, instead of the object with the matching name. For example, assume you have the following 3 objects:

OBJECTA(501,"name1")

© Copyright IBM Corp. 1997, 2000

OBJECTB(502,“name2”)
OBJECTC(503,“501”)

Searching for “501” will find OBJECTA, and **not** OBJECTC. Therefore, we recommend that you do not give objects numeric names.

Common Return Codes

0	successful
2-100	warning
255	error

fwadapter

Purpose

Lists or changes the state of a secure interface.

Format

```
fwadapter cmd=list
          [addr=x.x.x.x]

fwadapter cmd=change
          addr=x.x.x.x
          state={secure|nonsecure}
```

Keywords and Parameters

cmd=list
Lists all active non-loopback adapters attached to this machine and identifies each as being either a secure or a nonsecure adapter. If the optional *addr* parameter is specified, only that adapter is listed.

addr=*x.x.x.x*
is the IP address of an adapter in dotted-decimal format.

cmd=change
Sets the secure or nonsecure state of the adapter answering to the specified IP address.

addr=*x.x.x.x*
is the IP address of an adapter in dotted-decimal format.

state={secure|nonsecure}
identifies the adapter as either secure or non-secure.

Examples

1. List all active adapters

```
   User:    fwadapter cmd=list
System:     99.100.77.77      Non-Secure Interface  TCPIPAA   TR1
            99.100.50.250     Secure Interface      TCPIPAA   TR3
            99.100.55.55      Non-Secure Interface  TCPIPAA   TR4
```

2. List a specified adapter

```
   User:    fwadapter cmd=list addr=99.100.50.250
System:     99.100.50.250     Secure Interface      TCPIPAA   TR3
```

3. Change the adapter from secure to nonsecure

```
   User:    fwadapter cmd=change addr=99.100.50.250 state=nonsecure
System:
   User:    fwadapter cmd=list addr=99.100.50.250
System:     99.100.50.250     Non-Secure Interface  TCPIPAA   TR3
```

4. Restore the adapter to the secure state

```
   User:    fwadapter cmd=change addr=99.100.50.250 state=secure
System:
   User:    fwadapter cmd=list addr=99.100.50.250
System:     99.100.50.250     Secure Interface      TCPIPAA   TR3
```

fwahtran

Purpose

Configures AH transforms.

Format

```
fwahtran cmd=add
         name=name
        [desc=description]
        [mode={tunnel|transport}]
         authalg={HMAC_MD5|HMAC_SHA}
        [itime=minutes]               (minutes:   1-9999)
        [isize=kb]                    (kilobytes: 0-4194300)
        [rtime=minminutes-maxminutes] (minutes:   1-9999)
        [rsize={minkb-maxkb|0|0-0}]   (kilobytes: 0-4194300)

         defaults: mode=tunnel, itime=60,    isize=0,    rtime=60-480, rsize=0

fwahtran cmd=change
         ahtran=ah_transform_id_or_name
        [desc=[description]]
        [mode={tunnel|transport}]
        [authalg={HMAC_MD5|HMAC_SHA}]
        [itime=minutes]               (minutes:   1-9999)
        [isize=kb]                    (kilobytes: 0-4194300)
        [rtime=minminutes-maxminutes] (minutes:   1-9999)
        [rsize={minkb-maxkb|0|0-0}]   (kilobytes: 0-4194300)

fwahtran cmd=delete
         ahtran=ah_transform_id_or_name

fwahtran cmd=list
        [ahtran=ah_transform_id_or_name]
        [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds an AH transform.

name=*name*
the name of the AH transform. Maximum length is 100 characters.

desc=*description*
the description of the AH transform. Maximum length is 100 characters. This parameter is optional.

mode={tunnel|transport}
specifies the operational mode of the security association negotiation. The default is **tunnel**.

tunnel
specifies that the tunnel operates in tunnel mode, which protects the entire IP packet. Usually, this mode is used for a secure tunnel between two firewalls or between a firewall and a remote system; for example, when either of the two communicating systems is not an endpoint of the tunnel.

transport
specifies that the tunnel operates in transport mode, which only protects the transport-layer packet (for example, TCP or UDP packet) inside an IP packet. This mode is used when the endpoints of the secure tunnel are the two communicating systems.

authalg={HMAC_MD5|HMAC_SHA}
the authentication algorithm to use when exchanging user data.

itime=*minutes*
the security association lifetime to propose when negotiating a security association using this transform. Valid values are 1-9999. The default is 60 minutes.

isize=*kb*
the security association lifesize to propose when negotiating a security association using this transform. Valid values are 0-4194300. A value of 0 indicates no size limit.

rtime=*minminutes-maxminutes*
the security association lifetime range allowed when responding to a negotiation request using this transform. Valid values, for each value are 1-9999.

rsize={*minkb-maxkb|0|0-0***}**
the security association lifesize range allowed when responding to a negotiation request using this transform. Valid values, for each value are 0-4194300. A value of 0 indicates no size limit.

Note: **rsize=0-0** is equivalent to **rsize=0**, and it will be stored as 0, not as 0-0.

cmd=change
changes an AH transform. The parameters are the same as for **add** except that **ahtran** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes an AH transform.

ahtran=*ah_transform_id_or_name*
the ID or name of the AH transform to delete.

cmd=list
lists one or all AH transforms.

ahtran=*ah_transform_id_or_name*
the ID or name of the AH transform to list. If **ahtran** is not specified, all AH transforms are listed. This parameter is optional.

format={short|long|wide|raw}
the format of the listing. If **ahtran** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`
long	formatted as keyword = *value*, one parameter per line.

```
      id = id
    name = name
    desc = description
    mode = mode
 authalg = authalg
   itime = minutes
   isize = kb
   rtime = minminutes-maxminutes
   rsize = minkb-maxkb
```

wide 132 columns formatted as:

```
ID name desc
```

raw the values are separated by vertical bars, in the following order:

```
id|name|desc|mode|authalg|itime|isize|rtime|rsize
```

For example,

```
501|ahtran1|ah trans. desc.|tunnel|HMAC_MD5|60|9999|60-480|9999-99999
```

Usage

The AH transform is a description of a user data authentication method that is acceptable to the owners of this system. It is used when negotiating a security association for data transmission.

Use the **fwahtran cmd=list** command to list the AH transforms that are supplied with OS/390 Firewall Technologies. Carefully consider the listed AH transforms, then define all other AH transforms needed on this system.

After an AH transform is defined, associate it with a data proposal using the **fwdataprop** command.

Security association life values must be specified for both initiator and responder roles. The initiator role must have an expiration time. It may also have an expiration size. The responder role must have an expiration time range. It may also have an expiration size range. The initiator role values should be within the range specified for the responder.

Examples

1. Create an AH transform that uses transport mode and HMAC_MD5 authentication. This AH transform will use default values for the other parameters.

```
  User: fwahtran cmd=add name="AH Transform A" desc="This is AH transform A" \
        mode=transport authalg=HMAC_MD5
System:
  User: fwahtran cmd=list format=long
System:
                    id = 501
                  name = AH Transform A
                  desc = This is AH transform A
                  mode = transport
               authalg = HMAC_MD5
                 itime = 60
                 isize = 0
                 rtime = 60-480
                 rsize = 0
```

2. Create an AH transform that uses HMAC_SHA authentication. This AH transform will have a lifetime of 480 minutes and an unlimited lifesize when used to initiate a negotiation and a lifetime between 480 and 720 minutes and a maximum lifesize of 1 million kilobytes when used to respond to a negotiation request.

```
   User:     fwahtran cmd=add name="AH Transform B" desc="This is AH transform B" \
             authalg=HMAC_SHA itime=480 isize=0 rtime=480-720 rsize=1-1000000
System:
   User:     fwahtran cmd=list ahtran="AH Transform B"
System:
                     id = 502
                   name = AH Transform B
                   desc = This is AH transform B
                   mode = tunnel
                authalg = HMAC_SHA
                  itime = 480
                  isize = 0
                  rtime = 480-720
                  rsize = 1-1000000
```

3. Change an AH transform so that there is no limit on the security association responder lifesize (rsize). Note that if rsize is specified as '0-0' in the command, it shows in the list as '0'.

```
   User:     fwahtran cmd=change ahtran="AH Transform B" rsize=0-0
System:
   User:     fwahtran cmd=list ahtran="AH Transform B"
System:
                     id = 502
                   name = AH Transform B
                   desc = This is AH transform B
                   mode = tunnel
                authalg = HMAC_SHA
                  itime = 480
                  isize = 0
                  rtime = 480-720
                  rsize = 0
```

fwarchive

Purpose

Configure the archive attributes for Firewall syslog records.

Format

```
fwarchive cmd=add
          logfile=LogFileName
          logtime=DaysToKeepInLog      (Days=0-9999)
          arcfile=ArchiveFileName
          arctime=DaysToKeepInArchive  (Days=0-9999)
          workspace=WorkSpaceDirectory

fwarchive cmd=change
          index=Number
         [logfile=LogFileName]
         [logtime=DaysToKeepInLog]     (Days=0-9999)
         [arcfile=ArchiveFileName]
         [arctime=DaysToKeepInArchive] (Days=0-9999)
         [workspace=WorkSpaceDirectory]

fwarchive cmd=delete
          index=Number

fwarchive cmd=list
         [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
creates an archive log management definition.

logfile=*LogFileName*
identifies the file containing the Firewall syslog records. The file name must be specified as a fully-qualified name (starting with a "/" character), for example: `/tmp/local4.info`.

logtime=*DaysToKeepInLog*
indicates how many days a **syslog** record will remain in the logfile before being moved to the archive file. Valid values are 0-9999.

arcfile=*ArchiveFileName*
specifies the archive file. ArchiveFileName must be a fully-qualified file name, and must end with a **.a** suffix.

arctime=*DaysToKeepInArchive*
indicates how many days a **syslog** record will remain in the archive file before being purged. Valid values are 0-9999.

workspace=*WorkspaceDirectory*
specifies the directory that the archiving program should use for temporary work files when archiving **syslog** files. It must be a fully-qualified directory name.

cmd=change
changes an archive log management definition. The parameters are the same as **add**, except that **index** parameter must be specified.

index=*Number*
the numeric value associated with the archive log management definition. This value is displayed as part of the **fwarchive cmd=list** output.

cmd=delete
deletes an archive log management definition.

index=*Number*
the numeric value associated with the archive log management definition. This value is displayed as part of the **fwarchive cmd=list** output.

cmd=list
lists all archive definitions.

format={short|long|wide|raw}
the format of the listing. This parameter is optional the default is **short**. The formats available are:

short 80 columns formatted as:

```
index logfile arcfile
```

long
formatted as *keyword=value*, one parameter per line:

```
    index=number
  logfile=LogFileName
  logtime=DaysToKeepInLog
  arcfile=ArchiveFileName
  arctime=DaysToKeepInArchive
workspace=WorkspaceDirectory
```

wide
132 columns formatted as:

```
index logfile arcfile
```

raw
the values are separated by vertical bars, in the following order:

```
index|logfile|logtime|arcfile|arctime|workspace
```

Usage

The **fwarchive** command provides configuration definitions for record retention. It allows you to define the number of days a log record will remain in the log file before being archived. It also allows you to define the number of days a log record will remain in the archive file before it is purged. The movement and purging of log records is done by the **fwlogmgmt** command.

Examples

1. Add an archive record to move all records in the /var/fw/fwdata/local4.log log file that are older than 7 days to archive file /archive/local4.a, where they will live for 30 days.

```
    User:   fwarchive cmd=add logfile=/var/fw/fwdata/local4.log
            logtime=7 arcfile=/archive/local4.a arctime=30 workspace=/tmp
System:
    User:   fwarchive cmd=list
System:
                    index = 1
                  logfile = /var/fw/fwdata/local4.log
                  logtime = 7
                  arcfile = /archive/local4.a
                  arctime = 30
                workspace = /tmp
```

2. Change an existing archive definition to remove the archive data after 60 days.

```
    User:   fwarchive cmd=change index=1 arctime=60
System:
    User:   fwarchive cmd=list
System:
                    index = 1
                  logfile = /var/fw/fwdata/local4.log
                  logtime = 7
                  arcfile = /archive/local4.a
                  arctime = 60
                workspace = /tmp
```

3. List the archive definitions in short format.

```
    User:   fwarchive cmd=list format=short
System:     1     /var/fw/fwdata/local4.log     /archive/local4.a
```

4. Delete archive definition number 1.

```
    User:   fwarchive cmd=delete index=1
System:
```

fwaudio

Purpose

Configures RealAudio support.

Format

```
fwaudio cmd=list

fwaudio cmd=change
        {[port=[port_num]] [sessions=[max_sessions]]}
        (port_num=1-65535 max_sessions=0-2147483647)
```

Keywords and Parameters

cmd=change
changes the server port number and/or maximum concurrent sessions allowed.

port=*portnum*
server port number. The default server port number is 7070, but you can reconfigure it to any valid TCP port number in the range 1-65535.

sessions=*sessions*
maximum number of concurrent sessions that can be configured. Specify a non-negative integer from 0-2147483647. The default is 10.

Note:

The same port and session is passed to each firewall stack defined by **fwstack**. Therefore, each stack shares the same port number, and the session is the maximum number of sessions for each stack.

cmd=list
lists the server port number and number of concurrent sessions that can be configured.

Examples

1. List the current real audio information

```
   User:  fwaudio cmd=list
System:   port=7070
          sessions=10
```

2. Change the real audio port and session values

```
   User:  fwaudio cmd=change port=8080 sessions=100
System:
   User:  fwaudio cmd=list
System:   port=8080
          sessions=100
```

3. Restore the real audio values back to the default values

```
   User:  fwaudio cmd=change port= sessions=
System:
   User:  fwaudio cmd=list
System:   port=7070
          sessions=10
```

fwauthinfo

Purpose

Configures authentication information.

Format

```
fwauthinfo cmd=add
           name=name
          [desc=description]
           remkeysrv=key_server_id_or_name
           {shkey=preshared_key and/or
           certauth=cert_auth_id_or_name}

fwauthinfo cmd=change
           authinfo=auth_info_id_or_name
          [desc=[description]]
          [remkeysrv=key_server_id_or_name]
          [shkey=[preshared_key]]
          [certauth=[cert_auth_id_or_name]]

fwauthinfo cmd=delete
           authinfo=auth_info_id_or_name

fwauthinfo cmd=list
          [authinfo=auth_info_id_or_name]
          [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds authentication information for a remote server.

name=*name*
the name of the authentication information definition. Maximum length is 100 characters.

desc=*description*
the description of the authentication information definition. Maximum length is 100 characters. This parameter is optional.

remkeysrv=*key_server_id_or_name*
ID or name of the remote key server that will use this authentication information. This must represent a single IP address through the use of an **fwkeysrv** object.

certauth=*cert_auth_id_or_name*
ID or name of the certificate authority object to use with the specified remote server. Either **shkey** or **certauth** must be specified.

shkey=*preshared_key*
key to use with the specified remote server. The shared key must be entered as byte data and must consist of an even number of hexadecimal digits. Maximum length is 900 characters. Either **shkey** or **certauth** must be specified.

cmd=change
changes authentication information. The parameters are the same as for **add**, except that **authinfo** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes an authentication information object.

authinfo=*auth_info_id_or_name*
the ID or name of the authentication information to delete.

cmd=list
lists one or all authentication information objects.

authinfo=*auth_info_id_or_name*
the ID or name of the authentication information object to list. If **authinfo** is not specified, all authentication information objects are listed. This parameter is optional.

format={short|long|raw|wide}
the format of the listing. If **authinfo** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`remkeysrv = remotekeyserverid`
	`certauth = certauthid`
	`shkey = key`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|remkeysrv|certauth|shkey`
	For example,
	`501|sharedkey1|key desc.|504|502|01000a200d0303040e100302`

Usage

Authentication information is an association of a shared key and/or a certificate authority with a specific remote key server. This information is used to authenticate the remote key server. Before using the **fwauthinfo** command, you must define a remote key server using the **fwkeysrv** command. Before specifying a certificate authority, it must be defined with the **fwcertauth** command.

The shared key must be entered as byte data and must consist of an even number of hexadecimal digits. For example, if the key specified by your peer is represented in ASCII as:

`GOOD4YOU`

it must be entered as

`474f4f4434594f55`

There may be only one shared key and certificate authority associated with each key server. The remote key server should refer to a single ISAKMP server through an IP address or host name.

The **certauth** parameter is only applicable to the RSA signature mode of authentication. It is used by the local ISAKMP server to help guide its peer to pick a certificate that has been signed by the specified certificate authority. Details about the RSA signature mode of authentication are described in "Authentication in ISAKMP" on page 86.

The local ISAKMP server will be unable to help guide its peer to pick a certificate if any of the following conditions are true:

- No **authinfo** entry exists for the peer
- An **authinfo** entry exists for the peer but there is no **certauth** information specified or the **certauth** information references a RACDCERT label that doesn't exist on the key ring specified by the **fwkeyring** command.
- The OS/390 ISAKMP server is acting as the responder in main mode when using RSA signature authentication and the key server entry representing the peer does not contain the peer's IP address or host nname. Additional details about this contition can be found in the third scenario of "ISAKMP Main Mode Limitations" on page 101.

If the local ISAKMP server is unable to help guide the peer to pick its certificate, the local ISAKMP server may not be able to verify the peer's signature.

The **certauth** parameter has no influence over which certificate the local ISAKMP server will use when it creates its own signature. This certificate is determined by the identity of the local ISAKMP server and by any certificate authority guidance sent to us by our peer. If our peer does not send any certificate authority guidance, then the local ISAKMP server will use the first certificate on the RACF key ring that matches its identity.

Examples

1. Create two authentication information definitions: one that uses a certificate authority and one that uses a pre-shared key.

```
   User:   fwauthinfo cmd=add name="Authentication Information A" \
           desc="This is authentication information A" remkeysrv="Key Server 1" \
           certauth="Certificate Authority 1"
System:
   User:   fwauthinfo cmd=add name="Authentication Information B" \
           desc="This is authentication information B" remkeysrv="Key Server 2" \
           shkey=AB12CD34EF56
System:
   User:   fwauthinfo cmd=list authinfo="Authentication Information A"
System:
                     id = 504
                   name = Authentication Information A
                   desc = This is authentication information A
              remkeysrv = 513
                  shkey =
               certauth = 501
   User:   fwauthinfo cmd=list authinfo="Authentication Information B"
System:
                     id = 506
                   name = Authentication Information B
                   desc = This is authentication information B
              remkeysrv = 515
                  shkey = ab12cd34ef56
               certauth =
```

2. Change the remote key server that will use this authentication information. Also change this authentication information definition so that it refers to a certificate authority as well as to a pre-shared key.

User:
```
fwauthinfo cmd=change authinfo="Authentication Information B" \
remkeysrv="Key Server 3" certauth="Certificate Authority 2"
```
System:

User:
```
fwauthinfo cmd=list authinfo="Authentication Information B"
```
System:
```
                id = 506
              name = Authentication Information B
              desc = This is authentication information B
         remkeysrv = 516
             shkey = ab12cd34ef56
          certauth = 502
```

fwcertauth

Purpose

Configures certificate authorities.

Format

```
fwcertauth cmd=add
           name=name
          [desc=description]
           label="RACDCERT label"

fwcertauth cmd=change
           certauth=cert_auth_id_or_name
          [desc=[description]]
          [label={"RACDCERT label"}]

fwcertauth cmd=delete
           certauth=cert_auth_id_or_name

fwcertauth cmd=list
          [certauth=cert_auth_id_or_name]
          [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds a certificate authority.

name=*name*
the name of the certificate authority object. Maximum length is 100 characters.

desc=*description*
the description of the certificate authority object. Maximum length is 100 characters. This parameter is optional.

label=*"RACDCERT label"*
the RACDCERT label of the certificate authority. The maximum length of this string is 256 characters. If the label contains a blank, it must be enclosed in quotes.

cmd=change
changes a certificate authority. The parameters are the same as for **add**, except that **certauth** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a certificate authority by ID or name

cert=*cert_auth_id_or_name*
the ID or name of the certificate authority to delete.

cmd=list
lists one or all certificate authorities.

cert=*cert_auth_id_or_name*
the ID or name of the certificate authority. If **certauth** is not specified, all certificate authorities are listed. This parameter is optional.

format={short|long|raw|wide}
the format of the listing. If **certauth** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns, formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`label = label`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|label`
	For example,
	`501|Cert1|certificate for ISAKMP|ISAKMP Certificate`

Usage

After a certificate authority is stored in RACF, via RACDCERT, use the **fwcertauth** command to add the certificate to the VPN configuration.

When a certificate authority is added to the VPN configuration, the ISAKMP server is notified so that it may be added to the server's cache. If the server is not running when the certificate authority is added, the add will complete and the certificate authority will be available when the server is started. When a certificate authority's configuration information is changed, the server's cache is also changed. When a certificate authority is deleted from the VPN configuration, its entry in the server's cache is also deleted.

There are no predefined certificate authorities supplied with OS/390 Firewall Technologies.

Examples

1. Create a certificate authority.

```
   User: fwcertauth cmd=add name="Certificate Authority A" \
         desc="This is certificate authority A" label=SomeLabel
System:
   User: fwcertauth cmd=list format=long
System:
                     id = 501
                   name = Certificate Authority A
                   desc = This is certificate authority A
                  label = SomeLabel
```

2. Change the RACDCERT label of the certificate authority.

User: **fwcertauth cmd=change certauth="Certificate Authority A" label="Some Label"**
System:
User: **fwcertauth cmd=list certauth="Certificate Authority A"**
System:

```
                id = 501
              name = Certificate Authority A
              desc = This is certificate authority A
             label = Some Label
```

fwconns

Purpose

Configures firewall filtering connections.

Format

```
fwconns cmd=create
        name=name
       [desc=description]
        source=network_id_or_name
       [destination=network_id_or_name]
       {servicelist=service_id1,service_id2,...,service_idn and/or
        sockslist=socks_id1,socks_id2,...,socks_idn}
       [ondemand=ondemand_id_or_name]

fwconns cmd=change
       {name=name | id=id}
       [desc=[description]]
       [source=network_id_or_name]
       [destination=[network_id_or_name]]
       [servicelist=[service_id1,service_id2,...,service_idn]]
       [sockslist=[socks_id1,socks_id2,...,socks_ idn]]
       [ondemand=[ondemand_id_or_name]]

fwconns cmd=delete
       {name=name | id=id}

fwconns cmd=list
       [name=name | id=id]
       [format={short|long|wide|raw}]

fwconns cmd=move
        connection=connection_id_or_name
        after=[connection_id_or_name]

fwconns cmd=addservice
        servicelist=service_id1,service_id2,...,service_idn
        connection=connection_id_or_name

fwconns cmd=removeservice
        servicelist=service_id1,service_id2,...,service_idn
        connection=connection_id_or_name

fwconns cmd=moveservice
        service=service_id_or_name
        after=[service_id_or_name]
        connection=connection_id_or_name
```

```
fwconns cmd=addsocks
        sockslist=socks_id1,socks_id2, ...,socks_idn
        connection=connection_id_or_name

fwconns cmd=removesocks
        sockslist=socks_id1,socks_id2, ...,socks_idn
        connection=connection_id_or_name

fwconns cmd=movesocks
        socks=socks_id_or_name
        after=[socks_id_or_name]
        connection=connection_id_or_name
```

Keywords and Parameters

cmd=create
creates a connection.

name=*name*
name of the connection to be created. Maximum length is 100 characters.

desc=[*description***]**
description of the connection to be created. Maximum length is 100 characters. This parameter is optional.

source=*network_id_or_name*
ID of source network object or network object group for the connection.

destination=*network_id_or_name*
ID of destination network object or network object group for the connection. This parameter is optional only when socks rules are specified.

servicelist=*service_id1,service_id2,...,service_idn* **and/or**
sockslist=*socks_id1,socks_id2,...,socks_idn*
ID list of the filter rules and/or socks rules to apply to this connection.

ondemand=*ondemand_id_or_name*
ID or name of the on-demand object that will be associated with any anchor rules that belong to the services defined in the connection. This parameter is optional. An on-demand object will be ignored when a service contains non-anchor rules.

cmd=change
changes the definition of a connection. The parameters are the same as for **create** above, except that all parameters except **name** or **id** are optional. In addition, they may be specified as blank to delete existing values.

Note: Both **servicelist** and **sockslist** cannot be blank.

cmd=delete
deletes the connection specified by **name** or **id**.

cmd=list
lists a specific connection, or all connections that have been defined. If **name** or **id** is not specified, all services are listed.

name=*name*
name of the connection. This parameter is optional.

id=*ID*
ID of the connection. This parameter is optional.

format={short|long|wide|raw}
is the format of the list. The listing can be in any of the following formats:

Format	Example
short	`id(id1), name(16), desc(44)`

long	is formatted as: keyword = *value*, one parameter per line.
	`  id = id`
	`name = name`
	`desc = description`
	`⋮`

wide	same layout as short, but 132 characters wide.
raw	displays values in the same order as the files separated by vertical bars, in the following order: `id|name|desc|source|destination|servicelist|sockslist|ondemand`

Note: The default is **short** if no ID or name is specified; **long** if ID or name is specified.

cmd=move
moves a connection (re-orders connection objects within the list of all connections). Connections can be ordered, because they are applied to incoming IP traffic starting at the top of the connections file. The first rule whose information matches that of the incoming packet has its action enacted.

connection=*connection_id_or_name*
ID or name of the connection to be moved.

after=[*connection_id_or_name***]**
ID or name of the connection after which this connection should now appear; if a blank value is specified, move the connection to the top of the list.

cmd=addservice
adds one or more services to an already-defined connection.

servicelist=*service_id1,service_id2,...,service_idn*
the ID or list of IDs of the service or services to add to the connection object.

connection=*connection_id_or_name*
ID or name of the connection to which the service or services are to be added.

cmd=removeservice
removes a service from a connection.

servicelist=*service_id1,service_id2,...,service_idn*
ID of the service to be removed.

connection=*connection_id_or_name*
ID or name of the connection from which the service is to be removed

cmd=moveservice
moves (re-orders) services within a connection.

service=*service_id_or_name*
ID or name of the service to be moved.

after=[*service_id_or_name***]**
ID or name of the service after which the above service should now appear; if a blank value is specified, move the service to the top of the list.

connection=*connection_id_or_name*
ID or name of the connection in which the services are to be re-ordered.

cmd=addsocks
adds socks rules to a connection.

sockslist=*socks_id1,socks_id2, ...,socks_idn*
list of IDs of the socks rules to be added.

connection=*connection_id_or_name*
ID or name of the connection to which the rules are to be added.

cmd=removesocks
removes socks rules from a connection.

sockslist=*socks_id1,socks_id2, ...,socks_idn*
list of IDs of the socks rules to be removed.

connection=*connection_id_or_name*
ID or name of the connection from which the rules are to be removed.

cmd=movesocks
moves (re-orders) socks rules within a connection.

socks=*socks_id_or_name*
ID or name of the socks rule to be moved.

after=[*socks_id_or_name***]**
ID or name of the rule after which the above rule should now appear. If a blank value is specified, move socks rule to the top of the list.

connection=*connection_id_or_name*
ID or name of the connection in which the rules are to be re-ordered

Usage

The firewall connection is an association of the network objects or network object groups to specific services. Taking the service and tieing it to the objects forms a connection. Before using the **fwconns** command, you must define the list of services and list of network objects or network object groups that you want associated. There are no predefined IP filtering connections supplied with OS/390 Firewall Technologies

After all connections are defined, use the **fwfilter cmd=update** to generate the filter and socks rules files.

A **connection** is a communications definition. Connections are set up between network objects using **services** (socks and filter rules) that govern the types of communication allowed between them.

This command refers to services that you define using the **fwservice** command and socks rules defined using the **fwsrule** command.

Connections are one component of IP filtering. See Chapter 8, “Controlling Traffic Through the Firewall with Filters” on page 51 for more information about how filtering works.

Examples

1. Create a firewall connection with both service and socks definitions.

```
   User:   fwconns cmd=create name="Example Connection 1" source="Example Object 1" \
           destination="Example Object 2" servicelist=501,502 sockslist=504
System:
```

2. List the connection by name.

```
   User:   fwconns cmd=list name="Example Connection 1"
System:
                     id = 501
                   name = Example Connection 1
                   desc =
                 source = 501
            destination = 502
            servicelist = 501,502
              sockslist = 504
               ondemand =
```

3. Show all the connections, in raw format.

```
   User:   fwconns cmd=list format=raw
System:   501|Example Connection 1||501|502|501,502|504
```

4. Change the source and destination values in the connection.

```
   User:   fwconns cmd=change id=501 source=502 destination=501
System:
   User:   fwconns cmd=list id=501
System:
                     id = 501
                   name = Example Connection 1
                   desc =
                 source = 502
            destination = 501
            servicelist = 501,502
              sockslist = 504
               ondemand =
```

5. Add a service to the connection.

```
   User:   fwconns cmd=addservice servicelist=503 connection=501
System:
   User:   fwconns cmd=list id=501
System:
                     id = 501
                   name = Example Connection 1
                   desc =
                 source = 502
            destination = 501
            servicelist = 501,502,503
              sockslist = 504
               ondemand =
```

6. Move service 502 after service 503 within the servicelist of the connection.

```
   User:     fwconns cmd=moveservice service=502 after=503 connection=501
System:
   User:     fwconns cmd=list id=501
System:
                     id = 501
                   name = Example Connection 1
                   desc =
                 source = 502
            destination = 501
            servicelist = 501,503,502
              sockslist = 504
               ondemand =
```

7. Remove service 503 from the connection.

```
   User:     fwconns cmd=removeservice servicelist=503 connection=501
System:
   User:     fwconns cmd=list id=501
System:
                     id = 501
                   name = Example Connection 1
                   desc =
                 source = 502
            destination = 501
            servicelist = 501,502
              sockslist = 504
               ondemand =
```

8. Delete the connection.

```
   User:     fwconns cmd=delete id=501
System:
```

fwdaemon

Purpose

Stores basic daemon configuration information and start up options.

Format

```
fwdaemon cmd=change
         daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}
        [started={yes|no}]
        [timeout=[seconds]]              (seconds=60-1800)
        [restart=[seconds]]              (seconds=60-1800)
        [maxconns=connections]           (connections=1-15000)
        [runopts=ropts]
        [daemonopts=dopts]
        [outputfile=dddef_or_filename]

fwdaemon cmd=list
        [daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}]
        [format={short|wide|long|raw}]

fwdaemon cmd=query
        [daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}]

fwdaemon cmd=start
        [daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}]

fwdaemon cmd=stop
        [daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}]
```

Keywords and Parameters

cmd=change
changes the configuration attributes for a daemon. **Daemon** is required. All other parameters are optional.

daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}
the name of the daemon being changed.

started=*{yes | no}*
indicates whether the daemon will be started by FWKERN

- yes - daemon will be started by FWKERN
- no - daemon will not be started by FWKERN

timeout=seconds
number of seconds before daemon start up will time out. The default timeout value is 300 seconds. The range of valid values is 60 to 1800 seconds.

restart=seconds
number of seconds daemon must remain active to attempt a restart when the daemon ends. The default restart value is 300 seconds. The range of valid values is 60 to 1800 seconds.

maxconns=connections
maximum number of connections allowed for the specific daemon. This parameter is applicable for the SOCKD and PFTPD daemons only. The range of valid values is 1 to 15000 connections.

runopts=ropts
runtime options to be handled by the start procedure (the environment variables settings, etc.)

daemonopt=dopts
start up options to be handled by the specified daemon.

outputfile=dddef_or_filename
specifies where the daemon's standard output will be redirected. This can be either a **dd** definition or a fully qualified **hfs** file.

cmd=list
lists the attributes of a specific daemon or all daemons.

daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}
name of the daemon to be listed. This parameter is optional. If name is not specified, all daemons are listed.

format={short | wide | long | raw}
the format of the output. This list can be in any of the following formats.

short in the format: daemon_name, started status, timeout, restart, maxconns, daemon options

wide in the format: daemon_name, started status, timeout, restart, maxconns, daemon options

long in the format: keyword = *value*, one parameter per line

raw values are separated by vertical bars. The raw format on the **cmd=list format=raw** displays in the same order as the file ordering.

If a daemon name is not specified, the default is short. If a daemon name is specified, the default is long.

cmd=query
lists the FWKERN state of a specific daemon or all daemons. For the SOCKD and PFTPD daemons, information is also displayed about their individual processes.

daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}
name of the daemon to be queried. The parameter is optional. If name is not specified, all daemons are queried.

cmd=start
starts a specific daemon or all daemons.

daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}
name of the daemon to be started. The parameter is optional. If name is not specified, all daemons are started.

cmd=stop
stops a specific daemon or all daemons.

daemon={SOCKD|PFTPD|CFGSRV|ISAKMPD|FWSTACKD}
name of the daemon to be stopped. The parameter is optional. If name is not specified, all daemons are stopped.

Usage

fwdaemon allows the firewall administrator the ability to list or change daemon configuration attributes as well as query daemon status and start and stop daemons. To issue a **cmd=start** or **cmd=stop** command, the user must be permitted to the FWKERN.START.REQUEST facility (refer to “Granting Users and Groups Authority to Firewall Objects” on page 24).

fwdaemon provides a consistent interface to control the daemon configuration data, start up parameters, and environment variables. **fwdaemon** stores basic configuration information for each of the daemons, daemon start up options and environment variables that control the behavior of the daemons.

When determining your *maxconns* setting, one address space is created for every 300 connections configured.

Examples

1. List the current firewall daemon configuration, in short format

```
   User:  fwdaemon cmd=list
System:   SOCKD      Yes 300  300  300
          PFTPD      Yes 300  300  300        -ns
          CFGSRV     No  300  300  1
          FWSTACKD   Yes 300  300  1
          ISAKMPD    No  300  300  1
```

2. Change the configuration server to be startable by FWKERN with the server listening on port 1014 and using the SSL file /home/sufwl2/ssl2/key.kdb.

```
   User:  fwdaemon cmd=change daemon=cfgsrv started=yes \
          daemonopts="-p 1014 -f /home/sufwl2/ssl2/key.kdb"
System:
```

3. Show the configuration server configuration

```
   User:  fwdaemon cmd=list daemon=cfgsrv
System:
                 daemon = CFGSRV
                started = Yes
                timeout = 300
                restart = 300
               maxconns = 1
                runopts =
             daemonopts = -p 1014 -f /home/sufwl2/ssl2/key.kdb
             outputfile = DD:CFGSOUT
```

4. Start the configuration server

```
   User:  fwdaemon cmd=start daemon=cfgsrv
System:
```

5. Query all configured daemons

```
   User:  fwdaemon cmd=query
System:   ICAM1001i Firewall daemon SOCKD status is DOWN and process id is 0.
          ICAM1001i Firewall daemon PFTPD status is DOWN and process id is 0.
          ICAM1001i Firewall daemon CFGSRV status is READY and process id is 218103831.
          ICAM1001i Firewall daemon FWSTACKD status is READY and process id is 134217754.
          ICAM1001i Firewall daemon ISAKMPD status is DOWN and process id is 0.
```

6. Stop all running daemons

```
   User:    fwdaemon cmd=stop
 System:
   User:    fwdaemon cmd=query
 System:    ICAM1001i Firewall daemon SOCKD status is DOWN and process id is 0.
            ICAM1001i Firewall daemon PFTPD status is DOWN and process id is 0.
            ICAM1001i Firewall daemon CFGSRV status is DOWN and process id is 0.
            ICAM1001i Firewall daemon FWSTACKD status is DOWN and process id is 0.
            ICAM1001i Firewall daemon ISAKMPD status is DOWN and process id is 0.
```

7. Change the Configuration Server to execute in the Japanese code page (Ja_JP.IBM-939):

```
   User:    fwdaemon cmd=change daemon=cfgsrv
                       runopts='ENVAR("LANG-Ja_JP.IBM-939")'
 System:
   User:    fwdaemon cmd=list daemon=cfgsrv
                        daemon=cfgsrv
                       started=yes
                       timeout=300
                       restart=300
                      maxconns=1
                       runopts=ENVAR("LANG=Ja_JP.IBM-939")
                     daemonopts=-p 1014 -f/home/SUFWL2/SS12/key.kdb
                     outputfile=DD : CFGSOUT
 System:
```

fwdatapol

Purpose

Configures data policies.

Format

```
fwdatapol cmd=create
          name=name
         [desc=description]
         [pfs={none|GROUP1|GROUP2}]
          dataproplist=data_proposal_id_or_name[,...]

          defaults: pfs=none

fwdatapol cmd=change
          datapol=data_policy_id_or_name
         [desc=[description]]
         [pfs={none|GROUP1|GROUP2}]
         [dataproplist=data_proposal_id_or_name[,...]]

fwdatapol cmd=delete
          datapol=data_policy_id_or_name

fwdatapol cmd=list
         [datapol=data_policy_id_or_name]
         [format={short|long|wide|raw}]

fwdatapol cmd=add
          datapol=data_policy_id_or_name
          dataproplist=data_proposal_id_or_name[,...]

fwdatapol cmd=remove
          datapol=data_policy_id_or_name
          dataproplist=data_proposal_id_or_name[,...]

fwdatapol cmd=move
          datapol=data_policy_id_or_name
          dataprop=data_proposal_id_or_name
          after=[data_proposal_id_or_name]
```

Keywords and Parameters

cmd=create
creates a data policy.

name=*name*
the name of the data policy. Maximum length is 100 characters.

desc=*description*
the description of the data policy. Maximum length is 100 characters. This parameter is optional.

pfs={none|GROUP1|GROUP2}
specifies whether perfect forward secrecy is used when negotiating a security association, and if so, what Diffie-Hellman group is used. The default is **none**.

None	do not use perfect forward secrecy
Group1	modular exponentiation group with a 768-bit modulus
Group2	modular exponentiation group with a 1024-bit modulus

dataproplist=*data_proposal_id_or_name***[,...]**
IDs or names of the data proposals to apply to this data policy. If more than one is specified, separate them with commas.

The data proposals must be specified in preferred order, with the most preferred data proposal specified first.

cmd=change
changes a data policy. The parameters are the same as for **create**, except that **datapol** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a data policy.

datapol=*data_policy_id_or_name*
the ID or name of the data policy to delete.

cmd=list
lists one or all data policies.

datapol=*data_policy_id_or_name*
the ID or name of the data policy to list. If **datapol** is not specified, all data policies are listed. This parameter is optional.

format={short|long|wide|raw}
the format of the listing. If **datapol** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`dataproplist = dataproposalid1,...,dataproposalidn`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|pfs|dataproplist`
	For example:
	`501|dpolicy1|data policy desc.|none|501,503,502`

cmd=add
adds a list of data proposals to a data policy. The specified data proposals are added to the end of the data policy's current list.

datapol=*data_policy_id_or_name*
the ID or name of the data policy.

dataproplist=*data_proposal_id_or_name***[,...]**
IDs or names of the data proposals to add to the data policy. If more than one is specified, separate them with commas.

The data proposals must be specified in preferred order, with the most preferred data proposal specified first.

cmd=remove
removes a list of data proposals from a data policy.

datapol=*data_policy_id_or_name*
the ID or name of the data policy.

dataproplist=*data_proposal_id_or_name***[,...]**
IDs or names of the data proposals to remove from the data policy. If more than one is specified, separate them with commas.

Note: A data policy must reference at least one data proposal. If the list of data proposals contains all remaining data proposals, an error message is issued and nothing is removed.

cmd=move
moves a data proposal to a different place in a data policy's list of data proposals.

datapol=*data_policy_id_or_name*
the ID or name of the data policy.

dataprop=*data_proposal_id_or_name*
ID or name of the data proposal to move.

after=*data_proposal_id_or_name*
ID or name of the data proposal after which to place the above specified data proposal. If a blank value is specified, the data proposal is moved to the top of the list. To see the current order of the data proposals, issue a **cmd=list** with long or raw format.

Usage

The data policy is an ordered list of data proposals used when negotiating a security association for data transmission. This ordering is important when acting as an initiator of a dynamic connection. In this case, the policies are sent to the remote key server in initiator's order of preference as defined in the data policy definition. When acting as responder, the initiator's ordering takes precedence. Before using the **fwdatapol** command, you must define a data proposal using the **fwdataprop** command.

After a data policy is defined, associate it with a dynamic tunnel using the **fwdyntun** command.

Examples

The following data proposals will be used in examples:

```
501   Data Proposal 1
502   Data Proposal 2
503   Data Proposal 3
504   Data Proposal 4
```

1. Create two data policies.

```
User:   fwdatapol cmd=create name="Data Policy A" desc="This is data policy A" \
        dataproplist="Data Proposal 1"
System:
User:   fwdatapol cmd=create name="Data Policy B" desc="This is data policy B" \
        dataproplist="Data Proposal 2,Data Proposal 3"
System:
User:   fwdatapol cmd=list format=long
System:
                id = 501
              name = Data Policy A
              desc = This is data policy A
               pfs = none
     dataproplist = 501

                id = 502
              name = Data Policy B
              desc = This is data policy B
               pfs = none
     dataproplist = 502,503
```

2. Change a data policy so that it uses perfect forward secrecy with Diffie-Hellman Group 1.

```
User:   fwdatapol cmd=change datapol="Data Policy A" pfs=GROUP1
System:
User:   fwdatapol cmd=list datapol="Data Policy A"
System:
                id = 501
              name = Data Policy A
              desc = This is data policy A
               pfs = GROUP1
     dataproplist = 501
```

3. Add a list of data proposals to a data policy.

```
User:   fwdatapol cmd=add datapol="Data Policy B" \
        dataproplist="Data Proposal 1,Data Proposal 4"
System:
User:   fwdatapol cmd=list datapol="Data Policy B"
System:
                id = 502
              name = Data Policy B
              desc = This is data policy B
               pfs = none
     dataproplist = 502,503,501,504
```

4. Move a data proposal to the end of a data policy's list.

```
User:   fwdatapol cmd=move datapol="Data Policy B" dataprop="Data Proposal 2" \
        after="Data Proposal 4"
System:
User:   fwdatapol cmd=list datapol="Data Policy B"
System:
                id = 502
              name = Data Policy B
              desc = This is data policy B
               pfs = none
     dataproplist = 503,501,504,502
```

5. Remove a list of data proposals from a data policy.

User: **fwdatapol cmd=remove datapol="Data Policy B" **
dataproplist="Data Proposal 3,Data Proposal 4"

System:

User: **fwdatapol cmd=list datapol="Data Policy B"**

System:

```
                id = 502
              name = Data Policy B
              desc = This is data policy B
               pfs = none
      dataproplist = 501,502
```

fwdataprop

Purpose

Configures data proposals.

Format

```
fwdataprop cmd=create
           name=name
          [desc=description]
          {ahtranlist=ah_transform_id_or_name[,...]  and/or
           esptranlist=esp_transform_id_or_name[,...]}

fwdataprop cmd=change
           dataprop=data_proposal_id_or_name
          [desc=[description]]
          [ahtranlist=[ah_transform_id_or_name[,...]]]
          [esptranlist=[esp_transform_id_or_name[,...]]]

fwdataprop cmd=delete
           dataprop=data_proposal_id_or_name

fwdataprop cmd=list
          [dataprop=data_proposal_id_or_name]
          [format={short|long|wide|raw}]

fwdataprop cmd=addah
           dataprop=data_proposal_id_or_name
           ahtranlist=ah_transform_id_or_name[,...]

fwdataprop cmd=removeah
           dataprop=data_proposal_id_or_name
           ahtranlist=ah_transform_id_or_name[,...]

fwdataprop cmd=moveah
           dataprop=data_proposal_id_or_name
           ahtran=ah_transform_id_or_name
           after=[ah_transform_id_or_name]

fwdataprop cmd=addesp
           dataprop=data_proposal_id_or_name
           esptranlist=esp_transform_id_or_name[,...]

fwdataprop cmd=removeesp
           dataprop=data_proposal_id_or_name
           esptranlist=esp_transform_id_or_name[,...]

fwdataprop cmd=moveesp
           dataprop=data_proposal_id_or_name
           esptran=esp_transform_id_or_name
           after=[esp_transform_id_or_name]
```

Keywords and Parameters

cmd=create
creates a data proposal.

name=*name*
the name of the data proposal. Maximum length is 100 characters.

desc=*description*
the description of the data proposal. Maximum length is 100 characters. This parameter is optional.

ahtranlist=*ah_transform_id_or_name***[,...]**
IDs or names of the AH transforms to apply to this data proposal. If more than one is specified, separate them with commas. Only AH transforms may be specified for **ahtranlist** and only ESP transforms may be specified for **esptranlist**.

The transforms must be specified in preferred order, with the most preferred transform specified first.

esptranlist=*esp_transform_id_or_name***[,...]**
IDs or names of the ESP transforms to apply to this data proposal. If more than one is specified, separate them with commas. Only ESP transforms may be specified for **esptranlist** and only AH transforms may be specified for **ahtranlist**.

The transforms must be specified in preferred order, with the most preferred transform specified first.

cmd=change
changes a data proposal. The parameters are the same as for **create** except that **dataprop** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a data proposal.

dataprop=*data_proposal_id_or_name*
the ID or name of the data proposal to delete.

cmd=list
lists one or all data proposals.

dataprop=*data_proposal_id_or_name*
the ID or name of the data proposal to list. If **dataprop** is not specified, all data proposals are listed. This parameter is optional.

format={short|long|wide|raw}
the format of the listing. If **dataprop** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`
long	formatted as keyword = *value*, one parameter per line.

```
          id = id
        name = name
        desc = description
 ahtranlist = ahtransformid1,...,ahtransformidn
esptranlist = esptransformid1,...,esptransformidn
```

wide 132 columns formatted as:

```
ID name desc
```

raw the values are separated by vertical bars, in the following order:

```
id|name|desc|ahtranlist|esptranlist
```

For example:

```
501|dproposal1|data prop. desc.|505,503|501,502
```

cmd=addah
adds a list of AH transforms to a data proposal. The specified AH transforms are added to the end of the data proposal's current list.

dataprop=*data_proposal_id_or_name*
the ID or name of the data proposal.

ahtranlist=*ah_transform_id_or_name***[,...]**
IDs or names of the AH transforms to add to the data proposal. If more than one is specified, separate them with commas.

The AH transforms must be specified in preferred order, with the most preferred transform specified first. Only AH transforms may be specified.

cmd=removeah
removes a list of transforms from the **ahtranlist** in a data proposal.

dataprop=*data_proposal_id_or_name*
the ID or name of the data proposal.

ahtranlist=*ah_transform_id_or_name***[,...]**
IDs or names of the AH transforms to remove from the data proposal. If more than one is specified, separate them with commas. Only AH transforms may be specified.

Note: A data proposal must reference at least one AH or ESP transform. If the list of transforms contains all remaining transforms, an error message is issued and nothing is removed.

cmd=moveah
moves an AH transform to a different place in a data proposal's list of transforms.

dataprop=*data_proposal_id_or_name*
the ID or name of the data proposal.

ahtran=*ah_transform_id_or_name*
ID or name of the AH transform to move. Only AH transforms may be specified.

after=*ah_transform_id_or_name*
ID or name of the AH transform after which to place the above specified transform. If a blank value is specified, the transform is moved to the top of the list of AH transforms. To see the current order of the AH transforms, issue a **cmd=list** with long or raw format.

cmd=addesp
adds a list of ESP transforms to a data proposal. The specified transforms are added to the end of the data proposal's current list.

dataprop=*data_proposal_id_or_name*
the ID or name of the data proposal.

esptranlist=*esp_transform_id_or_name***[,...]**
IDs or names of the ESP transforms to add to the data proposal. If more than one is specified, separate them with commas.

The ESP transforms must be specified in preferred order, with the most preferred transform specified first. Only ESP transforms may be specified.

cmd=removeesp
removes a list of ESP transforms from a data proposal.

dataprop=*data_proposal_id_or_name*
the ID or name of the data proposal.

esptranlist=*esp_transform_id_or_name***[,...]**
IDs or names of the ESP transforms to remove from the data proposal. If more than one is specified, separate them with commas. Only ESP transforms may be specified.

Note: A data proposal must reference at least one AH or ESP transform. If the list of transforms contains all remaining transforms, an error message is issued and nothing is removed.

cmd=moveesp
moves an ESP transform to a different place in a data proposal's list of transforms.

dataprop=*data_proposal_id_or_name*
the ID or name of the data proposal.

esptran=*esp_transform_id_or_name*
ID or name of the ESP transform to move. Only ESP transforms may be specified.

after=*esp_transform_id_or_name*
ID or name of the ESP transform after which to place the above specified transform. If a blank value is specified, the transform is moved to the top of the list. To see the current order of the ESP transforms, issue a **cmd=list** with long or raw format.

Usage

The data proposal contains ordered lists of AH and ESP transforms used when negotiating a security association for data transmission. This ordering is important when acting as an initiator of a dynamic connection. In this case the ESP an AH transforms are sent to the remote key server in initiator's order of preference as defined in the data proposal definition. When acting as responder, the initiator's ordering takes precedence. Before using the **fwdataprop** command, you must define a transform using the **fwahtran** or **fwesptran** command.

The data proposal configuration must contain at least one transform to be valid. It can be an **ah** transform or an **esp** transform. Both transform lists cannot be empty; but one or the other can.

After a data proposal is defined, associate it with a data policy using the **fwdatapol** command.

Examples

The following AH transforms will be used in examples:

```
501   AH Transform 1
502   AH Transform 2
503   AH Transform 3
```

The following ESP transforms will be used in examples:

```
501   ESP Transform 1
502   ESP Transform 2
503   ESP Transform 3
```

1. Create two data proposals: one with an AH transform list and one with an ESP transform list.

```
   User:  fwdataprop cmd=create name="Data Proposal A" desc="This is data proposal A" \
          ahtranlist="AH Transform 1,AH Transform 2,AH Transform 3"
System:
   User:  fwdataprop cmd=create name="Data Proposal B" desc="This is data proposal B" \
          esptranlist="ESP Transform 1"
System:
   User:  fwdataprop cmd=list format=long
System:

                 id = 501
               name = Data Proposal A
               desc = This is data proposal A
        ahtranlist = 501,502,503
       esptranlist =

                 id = 502
               name = Data Proposal B
               desc = This is data proposal B
        ahtranlist =
       esptranlist = 501
```

2. Change a data proposal so that it includes an ESP transform list as well as an AH transform list.

```
   User:  fwdataprop cmd=change dataprop="Data Proposal A" esptranlist="ESP Transform 1"
System:
   User:  fwdataprop cmd=list dataprop="Data Proposal A"
System:

                 id = 501
               name = Data Proposal A
               desc = This is data proposal A
        ahtranlist = 501,502,503
       esptranlist = 501
```

3. Add ESP transforms to a data proposal's ESP transform list.

```
   User:  fwdataprop cmd=addesp dataprop="Data Proposal A" \
          esptranlist="ESP Transform 2,ESP Transform 3"
System:
   User:  fwdataprop cmd=list dataprop="Data Proposal A"
System:

                 id = 501
               name = Data Proposal A
               desc = This is data proposal A
        ahtranlist = 501,502,503
       esptranlist = 501,502,503
```

4. Move an AH transform to the end of a data proposal's AH transform list.

```
   User:  fwdataprop cmd=moveah dataprop="Data Proposal A" ahtran="AH Transform 1" \
          after="AH Transform 3"
System:
   User:  fwdataprop cmd=list dataprop="Data Proposal A"
System:

                 id = 501
               name = Data Proposal A
               desc = This is data proposal A
        ahtranlist = 502,503,501
       esptranlist = 501,502,503
```

5. Remove ESP transforms from a data proposal's ESP transform list.

User: **fwdataprop cmd=removeesp dataprop="Data Proposal A" **
esptranlist="ESP Transform 1,ESP Transform 2"

System:

User: **fwdataprop cmd=list dataprop="Data Proposal A"**

System:

```
                id = 501
              name = Data Proposal A
              desc = This is data proposal A
       ahtranlist = 502,503,501
      esptranlist = 503
```

fwdns

Purpose

Modifies or lists the DNS configuration for the **named** server.

Format

```
fwdns cmd=list
     [stack=stackname]


fwdns cmd=change
      secdomain=SecureDomainName
      secaddr={x.x.x.x | "x.x.x.x x.x.x.x x.x.x.x"}
      remaddr={x.x.x.x | "x.x.x.x x.x.x.x x.x.x.x"}
     [stack=stackname]
     [resolver=resolverFileName]
     [force={yes|no}]
```

Keywords and Parameters

cmd=list
lists the current Domain Name Server configuration.

stack=*stackname*
the stack name as defined on the **fwstack** command. This parameter is optional if only one stack is defined through the **fwstack** commmand.

This parameter provides a stack name to let DNS know where the DNS configuration files are for each stack. Because a DNS stack does not have to be a firewall stack, the **fwstack cmd=add force=yes** may be required for DNS configuration.

cmd=change
changes the configuration of the firewall's DNS.

secdomain=*SecureDomainName*
is the name of the secure domain.

secaddr=addr_list
remaddr=addr_list
are IP addresses of servers to be defined as secure (**secaddr**) or non-secure (**remaddr**). These two variables may be specified as a single address or a list of addresses, in dotted-decimal format. If a list of server addresses is being specified, the list should be delimited by spaces and contained within double quotes, for example: *"x.x.x.x x.x.x.x x.x.x.x"*.

stack=stackname
the stack name used to select the DNS configuration to modify.

The **stack** parameter is an optional parameter if there is only one firewall stack. If there is more than one firewall stack, then the **stack** parameter is required.

resolver=*ResolverFile*
is the TCP/IP UNIX Services resolver file.

Note: See *OS/390 IBM Communications Server: IP Configuration Reference*, SC31-8726, for more information on resolver configuration files.

In general, the search order for use of a resolver file is:

1. *ResolverFile* specified in this command (It must be an HFS file.)
2. previously specified resolver file
3. **/etc/security/fwstacks/**<*STACK*>**/fwresolv.conf**

If you want to use a PDS data set for the resolver file, you must change it based on the contents of **/etc/security/fwstacks/**<*STACK*>**/fwresolv.conf**; that is, comment out the existing domain and name server statements and add the statements found in **/etc/security/fwstacks/**<*STACK*>**/fwresolv.conf**. The TCPIPJOBNAME statement in **/etc/security/fwstacks/**<*STACK*>**/fwresolv.conf** must match that in the UNIX resolver file used by **named**; which in turn gets the value from **/etc/security/fwstack.cfg**. If the statements do not match, you should verify that the **/etc/security/fwstacks/**<*STACK*>**/fwdns.cfg** file (optional file; if not, it uses a default location) was used with the firewall stack.

This parameter is optional.

force={yes|no}
specifies that the change will be made without a confirmation prompt to the user. The default is **force=no**.

Usage

Use the command **fwdns** comand to configure domain name services. The configuration files created work with the OS/390 eNetwork Communications Server.

Note: If filters are being used, you need to set a security policy to predefined service 20 “Permit DNS Queries” to use the functions in this command. See “fwsecpolicy” on page 296 for more information.

If this is the first time DNS is being configured, **fwdns cmd=change** creates the following new files:

- `/etc/fwnamed.boot`

  Example:

  ```
  cache . /etc/named.ca
  primary 0.0.127.in-arda.arpa /etc/named.loc
  ```

  The domain name that is used in the boot file can be substituted for the @ symbol in the SOA statement.

- `/etc/fwnamed.ca`

  The following version of the cache file is created:

  ```
  externaldns.nnn.nnn.nnn.nnn. 3600000 IN A nnn.nnn.nnn.nnn
  ```

- `/etc/fwnamed.loc`

  Example:

  ```
  0.0.127.in-arda.arpa IN SOA firewall.secure.company.com
          root.firewall.secure.company.com (serialnum 3600 600 36000086400) IN NS firewall.secure.company.com
  1 IN PTR localhost.
  ```

  Note: All information appears on the same line.

- `/etc/security/fwstacks/`<*STACK*> directory (or user-specified UNIX resolver file is used, if supplied)

  Example:

```
domain secure.company.com
nameserver sss.sss.sss.sss
TCPIPJOBNAME stackname
```

The original /etc/security/fwstacks/<*STACK*> director file is modified and the original lines commented out. The next time DNS is configured, **fwdns cmd=change** will back up the files created by the previous **stack=stackname** on the syntax **cmd=change** and a new set of files are created. The files are backed up with the following names:

- /etc/fwnamed.ca is backed up to /etc/security/fwstacks/<*STACK*>/fwnamed.ca.sav
- /etc/fwnamed.loc is backed up to /etc/security/fwstacks/<*STACK*>/fwnamed.loc.sav

Note: Since the default boot file for the **named** server is /etc/named.boot and the **fwdns** command creates a boot file called /etc/fwnamed.boot, the **named** server must be started with the boot file specified. For example:

```
named -b /etc/fwnamed.boot
```

Examples

1. Change the DNS configuration for firewall stack TCPIPAA

```
User:   fwdns cmd=change secdomain=mytown.mycompany.com secaddr=99.100.50.250 \
        remaddr="99.100.5.5 99.100.10.10" stack=TCPIPAA
System:
```

2. List the DNS configuration for firewall stack TCPIPAA

```
User:   fwdns cmd=list stack=TCPIPAA
System: Secure Domain Name:             mytown.mycompany.com
        Secure Nameservers:             99.100.50.250
        Non-secure Nameservers:         99.100.5.5  99.100.10.10
        TCPIPJOBNAME:                   TCPIPAA
        OE resolver:                    /etc/security/fwstacks/TCPIPAA/fwresolv.conf
```

fwdynconns

Purpose

Configures dynamic connections.

Format

```
fwdynconns cmd=add
           name=name
          [desc=description]
          [autoact={yes|no}]
           src=network_id_or_name
           dest=network_id_or_name
           srcport=port_num            (port_num: 0-65535)
           destport=port_num           (port_num: 0-65535)
           prot={all|tcp|udp|icmp|ospf|ipip|esp|ah}
           remkeysrv=key_server_id_or_name
          [keysrvgrp=key_server_group_id_or_name]

           defaults: autoact=no, keysrvgrp=(first containing remkeysrv)

fwdynconns cmd=change
           dynconn=dynamic_connection_id_or_name
          [desc=[description]]
          [autoact={yes|no}]
          [src=network_id_or_name]
          [dest=network_id_or_name]
          [srcport=port_num]            (port_num: 0-65535)
          [destport=port_num]           (port_num: 0-65535)
          [prot={all|tcp|udp|icmp|ospf|ipip|esp|ah}]
          [remkeysrv=key_server_id_or_name]
          [keysrvgrp=[key_server_group_id_or_name]]

fwdynconns cmd=delete
           dynconn=dynamic_connection_id_or_name

fwdynconns cmd=list
          [dynconn=dynamic_connection_id_or_name]
          [format={short|long|wide|raw}]

fwdynconns cmd=activate
          [dynconnlist=dynamic_connection_id_or_name[,...]]

fwdynconns cmd=listactive
          [tunlist=tunnel_name[,...]]
          [stack=stackname]
          [format={short|long|raw}]

fwdynconns cmd=deactivate
           tunlist=tunnel_name[,...]

fwdynconns cmd=refresh
           tunlist=tunnel_name[,...]

fwdynconns cmd=shutdown
```

Keywords and Parameters

cmd=add
adds a dynamic connection

name=*name*
the name of the dynamic connection. Maximum length is 100 characters.

desc=*description*
the description of the dynamic connection. Maximum length is 100 characters. This parameter is optional.

autoact={yes|no}
specifies that this connection is activated when the TCP/IP stack is started provided there is an active anchor filter rule that matches. The default is **no**.

src=*network_id_or_name*
the ID or name of the source network object or network object group for the dynamic connection.

dest=*network_id_or_name*
the ID or name of the destination network object or network object group for the dynamic connection.

srcport=*port_num*
the source port number. Valid values are 0 - 65535. If 0 is specified, then all ports will be included.

destport=*port_num*
the destination port number. Valid values are 0 - 65535. If 0 is specified, then all ports will be included.

protocol=all|tcp|udp|icmp|ospf|ipip|esp|ah
the internet protocol to which this connection applies. Valid values are:

all	Any protocol
tcp	Transmission Control Protocol (TCP)
udp	User Datagram Protocol (UDP)
icmp	Internet Control Message Protocol (ICMP)
ospf	Open Shortest Path First protocol (OSPF)
ipip	IP-in-IP protocol. When **ipip** is specified, the port fields must be specified as 0.
esp	Encapsulating Security Protocol
ah	Authentication Header protocol

You may not use an internet protocol of **all** with port values of zero with non-zero ports. For example, you cannot specify **all** with (0,21). Both ports must be zero with protocol=all.

remkeysrv=*key_server_id_or_name*
the ID or name of the remote key server to use for the dynamic connections.

When the remote key server ID or name is entered, the key server group that it is a member of is determined and stored in the data repository in the **keysrvgrp** field. If a key server group is also defined, it is used instead of the first found.

keysrvgrp=*key_server_group_id_or_name*
the ID or name of the key server group used to set up the dynamic connection.

When looking for a remote key server's information, the first key server group found to contain the remote key server is used. If another key server is preferred, specify the key server group with this parameter.

cmd=change
changes a dynamic connection. The parameters are the same as for **add** except that **dynconn** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a dynamic connection by ID or name.

dynconn=*dynamic_connection_id_or_name*
the ID or name of the dynamic connection to delete.

cmd=list
lists one or all dynamic connections.

dynconn=*dynamic_connection_id_or_name*
the ID or name of the dynamic connection. If **dynconn** is not specified, all dynamic connections are listed. This parameter is optional.

format={short|long|raw|wide}
the format of the listing. If **dynconn** is not specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`autoact = autoactivate`
	`src = sourceid`
	`dest = destinationid`
	`srcport = srcport`
	`destport = destport`
	`prot = protocol`
	`remkeysrv = remotekeyserverid`
	`keysrvgrp = keyservergroupid`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|autoact|src|dest|srcport|destport|prot|remkeysrv|keysrvgrp`
	For example:
	`501|dynconn1|dynamic connection desc.|yes|504|528|21|0|tcp|508|531`

cmd=listactive
lists active dynamic connections.

tunlist=*generated_tunnel_id***[,...]**
the list of generated tunnel IDs as shown on a **fwdynconns cmd=listactive** command. If not specified, all dynamic connections are listed.

stack=stackname
the stack name used to select the active dynamic connections.

format={short|long|raw}
the format of the listing. If **tunlist** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `tunnelID src dest state activator`

long	formatted as keyword = *value*, one parameter per line.
	`TunnelID = generated_tunnel_ID`
	`TunSrc = source`
	`TunDest = destination`
	`State = state`
	`Activator = activator`
	`SrcAddr1 = src_connection_endpoint`
	`SrcAddr2 = src_connection_endpoint`
	`DestAddr1 = dest_connection_endpoint`
	`DestAddr2 = dest_connection_endpoint`
	`Tunnel expires at = time`
	`Bytes till expire = bytes`
	`AHEncapMode = AH_encap_mode`
	`ESPEncapMode = ESP_encap_mode`
	`AHAuthAlg = AH_auth_alg`
	`EncryptAlg = encrypt_alg`
	`ESPAuthAlg = ESP_auth_alg`
	`ConnID = connection_id`
	`SrcObjID = src_object_id`
	`DestobjID = dest_object_id`
	`SvcID = services_id`
	`FruleID = filter_rule_id`
	`DynTunID = dynamic_tunnel_id`

raw	parameters are separated by vertical bars, in the following order:
	`tunnelID|tunsrc|tundest|state|activator|srcaddr1|srcaddr2|destaddr1|destaddr2|`
	`exptime|expsize|ahencapmode|espencapmode|ahauthalg|encrypalg|espauthalg|`
	`connid|srcobjid|destobjid|svcid|fruleid|dyntunid`
	For example:
	`4|99.100.55.55|11.130.10.11|active|user|99.100.55.55|255.255.255.255|11.130.10.11|`
	`255.255.255.255|Wed Apr 21 19:23:35 1999 EDT|0|tunnel|tunnel|HMAC_SHA|DES_CBC_8|`
	`HMAC_SHA|509 501|515|516|506|513|509`

The state will be either **active** or **stopped**. Dynamic connections listed as **stopped** have reached the expiration time or size and are no longer active. They will be removed from the TCP/IP cache imminently and will no longer show on this list.

The activator will be **user**, **remote**, or **ondemand**. Dynamic connections listed as user activated were activated by the **fwdynconns cmd=activate** command on this system. Dynamic connections listed as remotely activated were activated by a remote system. Dynamic connections listed as activated ondemand were activated as needed.

If both **tunlist** and **stack** are specified on a **cmd=listactive**, the results will show the tunnels from **tunlist** that are active on the specified stack.

cmd=activate
activate dynamic connections.

dynconnlist=*dynamic_connection_id_or_name***[,...]**
the name or ID of the dynamic connections that are being activated. If not specified, all dynamic connections are activated.

The connections that can successfully be activated are activated. The connections that cannot be activated will return a message.

cmd=deactivate
deactivates dynamic connections.

tunlist=*generated_tunnel_id***[,...]**
the list of generated tunnel IDs as shown on a **fwdynconns cmd=listactive** command.

cmd=refresh
refresh dynamic connections

tunlist=*generated_tunnel_id***[,...]**
the list of generated tunnel IDs as shown on a **fwdynconns cmd=listactive** command.

The connections that can be refreshed are refreshed. The connections that cannot be refreshed will return a message.

cmd=shutdown
deactivates all dynamic connections.

Usage

The dynamic connection describes the communication link between two specific end points. These end points are defined as source and destination network objects. The source network object either identifies the system you are configuring or a host behind your system. The destination network object identifies the remote system for which this dynamic VPN connection is being established.

The **fwdynconns** command is also used to **activate**, **deactivate** and **refresh** specific connections as well as **shutdown** all activate dynamic connections.

The **fwdynconns** and **fwdyntun** commands are tied to each other through anchor rules on the **fwfrule** command. The *dynconns* parms must be a match or subset of an anchor rule in order for the tunnel to be associated with the *dynconns*. For example, a protocol of *all* on the anchor rule and *tcp* on the *dynconns* is correct. The reverse is not. This association applies to all *dynconns* parms.

Remkeysrv must resolve to a single IP address through the **fwkeysrv** command.

Key and data policy agreed to when a dynamic connection is activated is cached by the ISAKMP server. This cached information will be used when a refresh of a dynamic connection is initiated by the local key server. When a refresh is initiated by a remote key server, the policy database is consulted and the cached information is bypassed.

To correlate a specific tunnel to the dynamic connection parameters that were specified, the user must review the active filter rules to find the entry as specified by the tunnel ID.

Examples

1. Create a dynamic connection from port 60000 at source Network Object 1 to port 60000 at destination Network Group 1. This dynamic connection will not be activated automatically (because it uses the default 'no' for autoact). This connection will apply only to the TCP protocol. It will use Remote Key Server 2, which it will look for in Key Server Group 1.

```
   User:    fwdynconns cmd=add name="Dynamic Connection B" \
            desc="This is dynamic connection B" src="Network Object 1" \
            dest="Network Group 1" srcport=60000 destport=60000 prot=tcp \
            remkeysrv="Remote Key Server 2" keysrvgrp="Key Server Group 1"
System:
   User:    fwdynconns cmd=list dynconn="Dynamic Connection B"
System:
                    id = 508
                  name = Dynamic Connection B
                  desc = This is dynamic connection B
               autoact = no
                   src = 517
                  dest = 520
               srcport = 60000
              destport = 60000
                  prot = tcp
             remkeysrv = 516
             keysrvgrp = 521
```

2. Change a dynamic connection so that it only applies to the IP-in-IP protocol. The port fields must also be changed to 0 so that all ports are included.

```
   User:    fwdynconns cmd=change dynconn="Dynamic Connection A" prot=ipip srcport=0 \
            destport=0
System:
   User:    fwdynconns cmd=list dynconn="Dynamic Connection A"
System:
                    id = 505
                  name = Dynamic Connection A
                  desc = This is dynamic connection A
               autoact = yes
                   src = 517
                  dest = 518
               srcport = 0
              destport = 0
                  prot = ipip
             remkeysrv = 515
             keysrvgrp = 521
```

3. Activate a dynamic connection.

```
   User:    fwdynconns cmd=activate dynconnlist="Dynamic Connection A"
System:     ICAC1756i Processing dynconn: Dynamic Connection A
```

4. List the active dynamic connections for the TCPIPAA stack.

User: **fwdynconns cmd=listactive stack=TCPIPAA format=long**
System:

```
         TunnelID = 1
           TunSrc = 99.100.55.55
          TunDest = 11.130.10.6
            State = active
        Activator = user
         SrcAddr1 = 99.100.55.55
         SrcAddr2 = 255.255.255.255
        DestAddr1 = 11.130.10.6
        DestAddr2 = 255.255.255.255
Tunnel expires at = Fri Apr 30 11:11:28 1999 EDT
Bytes till expire = 0
    AH Encap Mode = tunnel
   ESP Encap Mode = tunnel
      AH Auth Alg = HMAC_MD5
      Encrypt Alg = DES_CBC_8
     ESP Auth Alg =
           ConnID = 502
         SrcObjID = 517
        DestObjID = 518
            SvcID = 510
          FruleID = 517
         DynTunID = 581
```

fwdyntun

Purpose

Configures dynamic tunnels.

Format

```
fwdyntun cmd=add
         name=name
        [desc=description]
         datapol=data_policy_id_or_name
        [init={local|remote|either}]
        [connlifetime=minutes]    (minutes: 0-525600)

         defaults: init=either, connlifetime=1440

fwdyntun cmd=change
         dyntun=dynamic_tunnel_id_or_name
        [desc=[description]]
        [datapol=data_policy_id_or_name]
        [init={local|remote|either}]
        [connlifetime=minutes]    (minutes: 0-525600)

fwdyntun cmd=delete
         dyntun=dynamic_tunnel_id_or_name

fwdyntun cmd=list
        [dyntun=dynamic_tunnel_id_or_name]
        [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds a dynamic tunnel.

name=*name*
the name of the dynamic tunnel. Maximum length is 100 characters.

desc=*description*
the description of the dynamic tunnel. Maximum length is 100 characters. This parameter is optional.

datapol=*data_policy_id_or_name*
the policy to use when negotiating a security association for this dynamic tunnel.

init={local|remote|either}
which system may negotiate security associations for this dynamic tunnel.

local
specifies that this system must initiate the negotiation.

remote
specifies that another system must initiate the negotiation.

either
specifies that this system may either initiate a negotiation or respond to a negotiation initiated by another system.

The default is **either**.

connlifetime=*minutes*
the connection lifetime. Valid values for minutes are 0-525600. Specifying 0 means that the lifetime is infinite. The default is 1440 minutes.

cmd=change
changes a dynamic tunnel. The parameters are the same as for **add** except that **dyntun** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a dynamic tunnel.

dyntun=*dynamic_tunnel_id_or_name*
the ID or name of the dynamic tunnel to delete.

cmd=list
lists one or all dynamic tunnels.

dyntun=*dynamic_tunnel_id_or_name*
the ID or name of the dynamic tunnel to list. If **dyntun** is not specified, all dynamic tunnels are listed. This parameter is optional.

format={short|long|wide|raw}
the format of the listing. If **dyntun** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`datapol = datapolicyid`
	`init = initiation`
	`connlifetime = connlifetime`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|datapol|init|connlifetime`
	For example:
	`501|dyntunnel1|dynamic tunnel desc.|502|either|1440`

Usage

A dynamic tunnel is a description of the data policy and role information used to negotiate dynamic security associations.

After a dynamic tunnel is defined, associate it with an anchor rule(s) using the **fwfrule** command.

The *init* variable setting does not apply to refreshes. If a tunnel is nearing expiration, either side may initiate the key refresh.

Examples

1. Create a dynamic tunnel for which the local system may initiate the security association negotiation using Data Policy 1. The connection lifetime for this tunnel will be 1000 minutes.

```
   User:  fwdyntun cmd=add name="Dynamic Tunnel A" desc="This is dynamic tunnel A" \
          datapol="Data Policy 1" init=local connlifetime=1000
System:
   User:  fwdyntun cmd=list format=long
System:
                    id = 501
                  name = Dynamic Tunnel A
                  desc = This is dynamic tunnel A
               datapol = 501
                  init = local
          connlifetime = 1000
```

2. Create a dynamic tunnel for which the remote system must initiate the negotiation using Data Policy 2. This tunnel will utilize the default connection lifetime of 1440 minutes.

```
   User:  fwdyntun cmd=add name="Dynamic Tunnel B" desc="This is dynamic tunnel B" \
          datapol="Data Policy 2" init=remote
System:
   User:  fwdyntun cmd=list dyntun="Dynamic Tunnel B"
System:
                    id = 502
                  name = Dynamic Tunnel B
                  desc = This is dynamic tunnel B
               datapol = 502
                  init = remote
          connlifetime = 1440
```

3. Change a dynamic tunnel so that either the local or remote system may initiate the security association negotiation.

```
   User:  fwdyntun cmd=change dyntun="Dynamic Tunnel A" init=either
System:
   User:  fwdyntun cmd=list dyntun="Dynamic Tunnel A"
System:
                    id = 501
                  name = Dynamic Tunnel A
                  desc = This is dynamic tunnel A
               datapol = 501
                  init = either
          connlifetime = 1000
```

fwesptran

Purpose

Configures ESP transforms.

Format

```
fwesptran cmd=add
          name=name
         [desc=description]
         [mode={tunnel|transport}]
         [authalg={HMAC_MD5|HMAC_SHA}]
          encralg={DES_CBC_8|3DES_CBC|ESP_NULL}
         [itime=minutes]               (minutes:   1-9999)
         [isize=kb]                    (kilobytes: 0-4194300)
         [rtime=minminutes-maxminutes] (minutes:   1-9999)
         [rsize={minkb-maxkb|0|0-0}]  (kilobytes: 0-4194300)

           defaults: mode=tunnel, itime=60, isize=0, rtime=60-480, rsize=0

fwesptran cmd=change
          esptran=esp_transform_id_or_name
         [desc=[description]]
         [mode={tunnel|transport}]
         [authalg=[{HMAC_MD5|HMAC_SHA}]]
         [encralg={DES_CBC_8|3DES_CBC|ESP_NULL}]
         [itime=minutes]               (minutes:   1-9999)
         [isize=kb]                    (kilobytes: 0-4194300)
         [rtime=minminutes-maxminutes] (minutes:   1-9999)
         [rsize={minkb-maxkb|0|0-0}]   (kilobytes: 0-4194300)

fwesptran cmd=delete
          esptran=esp_transform_id_or_name

fwesptran cmd=list
         [esptran=esp_transform_id_or_name]
         [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds an ESP transform.

name=*name*
the name of the ESP transform. Maximum length is 100 characters.

desc=*description*
the description of the ESP transform. Maximum length is 100 characters. This parameter is optional.

mode={tunnel|transport}
specifies the operational mode of the security association negotiation. The default is **tunnel**.

tunnel
specifies that the tunnel operates in tunnel mode, which protects the entire IP packet. Usually, this mode is used for a secure tunnel between two firewalls or between a firewall and a remote system; for example, when either of the two communicating systems is not an endpoint of the tunnel. The default value is **mode=tunnel**.

transport
specifies that the tunnel operates in transport mode, which only protects the transport-layer packet (for example, TCP or UDP packet) inside an IP packet. This mode is used when the endpoints of the secure tunnel are the two communicating systems.

authalg={HMAC_MD5|HMAC_SHA}
the authentication algorithm to use when exchanging user data. If **encralg=ESP_NULL**, then an **authalg** must be specified. This parameter is optional.

encralg={DES_CBC_8|3DES_CBC|ESP_NULL}
the encryption algorithm to use when exchanging user data.

itime=*minutes*
the security association lifetime to propose when negotiating a security association using this transform. Valid values are 1-9999. The default is 60 minutes.

isize=*kb*
the security association lifesize to propose when negotiating a security association using this transform. Valid values are 0-4194300. A value of 0 indicates no size limit.

rtime=*minminutes-maxminutes*
the security association lifetime range allowed when responding to a negotiation request using this transform. Valid values, for each value are 1-9999. The default is 60-480 minutes.

rsize={*minkb-maxkb|0|0-0***}**
the security association lifesize range allowed when responding to a negotiation request using this transform. Valid values, for each value are 0-4194300. A value of 0 indicates no size limit.

Note: **rsize=0-0** is equivalent to **rsize=0**, and it will be stored as 0, not as 0-0.

cmd=change
changes an ESP transform. The parameters are the same as for **add**, except that **esptran** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes an ESP transform.

esptran=*esp_transform_id_or_name*
the ID or name of the ESP transform to delete.

cmd=list
lists one or all ESP transforms.

esptran=*esp_transform_id_or_name*
the ID or name of the ESP transform to list. If **esptran** is not specified, all ESP transforms are listed. This parameter is optional.

format={short|long|wide|raw}
the format of the listing. If **esptran** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`mode = mode`
	`authalg = authalg`
	`encralg = encralg`
	`itime = minutes`
	`isize = kb`
	`rtime = minminutes-maxminutes`
	`rsize = minkb-maxkb`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|mode|authalg|encralg|itime|isize|rtime|rsize`
	For example:
	`501|esptran1|esp trans|tunnel|HMAC_MD5|DES_CBC_8|60|9999|60-480|9999-99999`

Usage

The ESP transform is a description of a user data authentication and encryption method that is acceptable to the owners of this system. It is used when negotiating a security association for data transmission.

To authenticate using an ESP header, without encryption rather than an AH header, specify **encralg=ESP_NULL** and the preferred authentication algorithm for **authalg**.

Use the **fwesptran cmd=list** command to list the ESP transforms that are supplied with OS/390 Firewall Technologies. Carefully consider the listed ESP transforms, then define all other ESP transforms needed on this system.

After an ESP transform is defined, associate it with a data proposal using the **fwdataprop** command.

Session expiration values must be specified for both initiator and responder roles. The initiator role must have an expiration time. It may also have an expiration size. The responder role must have an expiration time range. It may also have an expiration size range. The initiator role values should be within the range specified for the responder.

Examples

1. Create an ESP transform that uses transport mode and DES_CBC_8 encryption. This ESP transform will use default values for the other parameters.

```
   User:     fwesptran cmd=add name="ESP Transform A" desc="This is ESP transform A" \
             mode=transport encralg=DES_CBC_8
System:
   User:     fwesptran cmd=list format=long
System:
                    id = 501
                  name = ESP Transform A
                  desc = This is ESP transform A
                  mode = transport
               authalg =
               encralg = DES_CBC_8
                 itime = 60
                 isize = 0
                 rtime = 60-480
                 rsize = 0
```

2. Create an ESP transform that uses HMAC_SHA authentication using an ESP header, but no encryption. This ESP transform will have a lifetime of 480 minutes and an unlimited lifesize when used to initiate a negotiation and a a lifetime between 480 and 720 minutes and a maximum lifesize of 1 million kilobytes when used to respond to a negotiation request.

```
   User:     fwesptran cmd=add name="ESP Transform B" desc="This is ESP transform B" \
             authalg=HMAC_SHA encralg=ESP_NULL itime=480 isize=0 rtime=480-720 \
             rsize=1-1000000
System:
   User:     fwesptran cmd=list esptran="ESP Transform B"
System:
                    id = 502
                  name = ESP Transform B
                  desc = This is ESP transform B
                  mode = tunnel
               authalg = HMAC_SHA
               encralg = ESP_NULL
                 itime = 480
                 isize = 0
                 rtime = 480-720
                 rsize = 1-1000000
```

3. Change an ESP transform so that there is no limit on the security association responder lifesize (rsize). Note that if rsize is specified as '0-0' in the command, it shows in the list as '0'.

```
   User:     fwesptran cmd=change esptran="ESP Transform B" rsize=0-0
System:
   User:     fwesptran cmd=list esptran="ESP Transform B"
System:
                    id = 502
                  name = ESP Transform B
                  desc = This is ESP transform B
                  mode = tunnel
               authalg = HMAC_SHA
               encralg = ESP_NULL
                 itime = 480
                 isize = 0
                 rtime = 480-720
                 rsize = 0
```

fwfilter

Purpose

Updates, verifies, logs, lists and shuts down IP filters.

Format

```
fwfilter cmd={update
        [predecap={yes|no}]
        [stack=stackname]

fwfilter cmd={verify|list|listactive|shutdown|startlog|stoplog|listsocks}
        [stack=stackname]
```

Keywords and Parameters

cmd=update
refreshes the filter rules file and activates filter and Socks rules.

predecap={yes|no}
to signal the stack to avoid the AH/ESP filtering that is done prior to decapsulation, you can use the optional subcommand **predecap={yes|no}** with **cmd=update**. The default for this parameter is **yes** (filter enabled).

cmd=verify
verifies the filter rule file for valid syntax and dependencies.

cmd=list
lists the last generated filter rules.

cmd=listactive
lists the currently activated static and dynamic filter rules from the stack. Dynamic filters are placed before anchor filters in an order determined by the stack. Both filter names and dynamic tunnel names are displayed.

cmd=shutdown
deactivates all filter rules.

cmd=startlog
starts logging of filter, tunnel and NAT activity to the logging daemon (SYSLOGD), using the **local4** facility. See Chapter 6, “Managing Firewall Logging Activity” on page 45 for more information about configuring logging.

cmd=stoplog
stops logging of filter, tunnel and NAT activity.

cmd=listsocks
lists the socks rules in the socks configuration file.

stack=*stackname*
an optional parameter to specify the stack name. This parameter can be used by all the **fwfilter** commands. If the stack name is not specified, the **fwfilter** command applies to all stacks. If the stack name is specified, the **fwfilter** command applies to the stack specified by *stackname*.

Note: A stack can be specified for a **listsocks** command, which is not stack specific. If the stack name is used for **listsocks**, then the stack is verified as part of the **fwstack.cfg** configuration, but is not used because there is only one **sockd.conf** file generated. (This applies to all stacks.)

Usage

Administration of IP filtering.

Examples

1. List the active IP filter rules.

```
   User:   fwfilter cmd=listactive stack=TCPIPAA
 System:
         IBM Firewall Filter Rules Dump Facility (Level 2.10).
         Dump Facility run at 09:05:09 on Apr-28-2000 for Firewall TCP/IP stack TCPIPAA
         Source of input for dump was storage.
         Number of secure interfaces currently defined is 1.
           Secure interface address 1 : 99.100.50.250
         Number of rules in the "active" rules array is 1.
         Status of filter support code        : active.
         Status of packet logging             : enabled.
         Status of AH/ESP pre-decap filtering : enabled.
         Last update to rules info made at 09:02:08 on Apr-28-2000.
         Number of updates to rules info since initialization is 0.
         Rule 1:
           Rule Type                  : static/non-ipsec
           Rule action                : deny
           Source Addr Type           : Address/Mask
           Source Field 1             : 0.0.0.0
           Source Field 2             : 0.0.0.0
           Destination Addr Type      : Address/Mask
           Destination Field 1        : 0.0.0.0
           Destination Field 2        : 0.0.0.0
           Protocol                   : all
           Source Port/ICMP/OSPF Type : any 0
           Destination Port/ICMP Code : any 0
           Interface                  : both
           Routing                    : both
           Direction                  : both
           Logging control            : yes
           Fragment control           : yes
           Tunnel name                : not applicable
           Time                       : none
           On-demand Enabled          : not applicable
           Source IP Granularity      : not applicable
           Destination IP Granularity : not applicable
```

2. List the socks configuration rules.

```
   User:   fwfilter cmd=listsocks
 System:   # This file is generated by Firewall configuration.
           # DO NOT EDIT THIS FILE MANUALLY.
           #
           #        For Network Object 1
           permit 99.100.55.55 255.255.255.255   s=m d=m
```

3. Verify the filter rule file for valid syntax and dependencies.

```
   User:   fwfilter cmd=verify
 System:   ICAC0001i Command completed successfully.
```

4. Deactivate all filter and socks rules.

```
   User:   fwfilter cmd=shutdown
System:
```

5. Refresh the filter rules file and activate filter and socks rules.

```
   User:   fwfilter cmd=update
System:   ICAC1577i Processing firewall TCP/IP stack TCPIPAA:

          Filter support initialized at 09:05:13 on Apr-28-2000
```

6. Stop logging of filter, tunnel and NAT activity.

```
   User:   fwfilter cmd=stoplog
System:
```

7. Start logging again.

```
   User:   fwfilter cmd=startlog
System:
```

fwfrule

Purpose

Configures IP filter rules.

Format

```
fwfrule cmd=add
        name=name
       [desc=description]
        type={permit|deny}
        protocol={all|tcp|tcp/ack|udp|icmp|ospf|ipip|esp|ah}
       [srcopcode={any|eq|neq|lt|gt|le|ge} and
        srcport=port_num]                        (port_num=0-65535)
       [destopcode={any|eq|neq|lt|gt|le|ge} and
        destport=port_num]                       (port_num=0-65535)
        interface={both|secure|nonsecure}
        routing={both|local|route}
        direction={both|inbound|outbound}
       [log={yes|no}]
       [tunnel=tunnel_id]                        (tunnel_id=1-999999)

fwfrule cmd=add
        name=name
       [desc=description]
        type={anchor}
        protocol={all|tcp|udp|icmp|ospf|ipip|esp|ah}
       [srcopcode={any|eq|neq|lt|gt|le|ge} and
        srcport=port_num]                        (port_num=0-65535)
       [destopcode={any|eq|neq|lt|gt|le|ge} and
        destport=port_num]                       (port_num=0-65535)
        interface={both|secure|nonsecure}
        routing={both|local|route}
       [log={yes|no|permit|deny}]
        tunnel=dynamic_tunnel_id_or_name

fwfrule cmd=change
        name=name | id=ruleid
       [desc=[description]
       [type={permit|deny|anchor}]
       [protocol={all|tcp|tcp/ack|udp|icmp|ospf|ipip|esp|ah}]
        (tcp/ack not supported on anchor rules)
       [srcopcode=[{any|eq|neq|lt|gt|le|ge}] and
        srcport=[port_num]]                      (port_num=0-65535)
       [destopcode=[{any|eq|neq|lt|gt|le|ge}] and
        destport=[port_num]]                     (port_num=0-65535)
       [interface={both|secure|nonsecure}]
       [routing={both|local|route}]
    Anchor Rules:
       [log=[{yes|no|permit|deny]}}
       [tunnel=dynamic_tunnel_id_or_name]
    Permit/Deny Rules:
       [direction={both|inbound|outbound}]
       [log=[{yes|no}]]
       [tunnel=[tunnel_id]]                      (tunnel_id=1-999999)
```

```
fwfrule cmd=delete
        name=name | id=ruleid
       [force={yes|no}]

fwfrule cmd=list
       [name=name] | [id=ruleid]
       [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds a filter rule.

name=*name* **| id=***ruleid*
name or ID of the filter rule. Maximum length is 100 characters.

desc=*description*
text description of the filter rule. Maximum length is 100 characters. This parameter is optional.

type={permit|deny|anchor}
type of filter rule.

- Use `type=permit` to allow a filter rule.
- Use `type=deny` to deny a filter rule.
- Use `type=anchor` to specify a dynamic filter rule.

protocol={all|tcp|tcp/ack|udp|icmp|ospf|ipip|esp|ah}
internet protocol to which this rule applies. Valid values are:

all	Any protocol
tcp	Transmission Control Protocol (TCP)
tcp/ack	TCP with acknowledgement (not supported for anchor rules)
udp	user datagram protocol (UDP)
icmp	Internet Control Message Protocol (ICMP)
ospf	Open Shortest Path First Protocol (OSPF).
ipip	IP-in-IP protocol (IPIP). When IPIP is specified, the port fields must be specified as **any 0**.
esp	Encapsulating Security Payload protocol
ah	Authentication Header protocol

Note: **tcp/ack** is not valid when **type = anchor**.

srcopcode={any|eq|neq|lt|gt|le|ge}
opcode for the source operation. **Any** may be specified, and if so, **srcport** must = 0. This parameter is optional, and if specified, must be accompanied by a **srcport** value.

See “Using the Operation and Port Number Operands” on page 243 for more information on how to use this and the next 3 operation and port number operands.

srcport=*port_num*
source port number. Valid values are 0 - 65535. This parameter is optional, and if specified, must be accompanied by a **srcopcode** value.

destopcode={any|eq|neq|lt|gt|le|ge}
opcode for the destination operation. **Any** may be specified, and if so, **destport** must = 0. This parameter is optional, and if specified, must be accompanied by **destport**.

destport=*port_num*
destination port. Valid values are 0 - 65535. This parameter is optional, and if specified, must be accompanied by a **destopcode** value.

interface={both|secure|nonsecure}
specifies the interface the rule applies to. Valid values are:

both For packets coming or going on either the secure or the nonsecure interface

secure For packets coming or going on the secure interface

nonsecure
For packets coming or going on the nonsecure interface

routing={both|local|route}
type of traffic to which the rule applies. Valid values are:

both Applies to all traffic.

local Implies that the packet is local to the firewall host. This means that:

- Incoming local packets are packets that are received by the interface and are destined for this firewall host; they will not be routed to another host.
- Outgoing packets are transmitted from the interface, but originate on the firewall host.

route Implies that the packet is routed by the firewall host. This means that:

- Incoming routed packets are packets that are received by the interface and are destined for some other host; they will not remain on the firewall.
- Outgoing packets are transmitted from the interface, and originated on some other host.

direction={both|inbound|outbound}
specifies the packet's direction through the interface. Valid values are:

both For packets going out from or into the selected interface

inbound For packets coming into the selected interface from the network

outbound
For packets going out from the selected interface to the network

Note: This parameter does not apply for **anchor** rules.

log={yes|no|permit|deny}
specifies whether to log the rule.

For **permit|deny** rules: if you specify Yes, every packet that matches that rule gets recorded in the **local4** log. If this parameter is not specified, the default is No for PERMIT rules and Yes for DENY rules.

For **anchor** rules: Permit - log all packets matching a dynamic filter for the given anchor rule. Deny - log all packets that match the anchor rule filter but that do not match a dynamic filter. Yes - logs all packets that match the anchor filter. No - no logging. The default is No.

This parameter is optional.

tunnel=*tunnel_id* **or**

tunnel=*dynamic_tunnel_id_or_name*

For **permit|deny** rules, tunnel must be a manual tunnel. For **anchor** rules, tunnel must be a dynamic tunnel ID or name.

cmd=change
changes a filter rule. Parameters are the same as for **add** above, except that all parameters except **name=** or **id=** are optional. In addition, **desc=**, **log=**, **tunnel=**, and **fragment=** may be specified as blank to delete existing information from the filter. For **anchor** rules, **tunnel=** cannot be blank.

cmd=delete
deletes a filter rule.

name=*name* **| id=***ruleid*
is the name or ID of the rule to be deleted.

force={yes|no}
specifies whether the delete is forced if there are dependencies on this rule. If "yes" is specified, the rule is deleted and any references to it are removed from services and connections. This parameter is optional. The default is no.

cmd=list
lists one or all filter rules.

name=*name* **| id=***ruleid*
is the name or ID of the rule to be listed. If **name=** or **id=** are not specified, all rules are listed.

This parameter is optional.

format={short|long|wide|raw}
is the format of the listing. Following are the formats available:

Format	Description
short	80 columns, formatted as: id(id1), type(permit or deny), name(rule1), desc(desc1).

long	is formatted as: keyword=value, one parameter per line. For example:
	`  id = id`
	`type = type`
	`name = name`
	`desc = desc`
	⋮

wide	similar to **short** format, but 132 characters wide.

raw	the values are separated by vertical bars, in the following order:
	`id|type|name|desc|protocol|srcopcode|srcport|`
	`destopcode|destport|interface|routing|direction|log|tunnel`
	`|fragment`

This parameter is optional. Default is **short** if no **name=** or **id=** is specified; **long** if **name=** or **id=** is specified.

Usage

A **filter rule** identifies a set of filter parameters. Filter rules are grouped together into **services** that perform an operation. The source and destination of a filter rule are **network objects**, which you define using the **fwnwobj** and **fwnwgrp** commands.

You can use the **fwfrule cmd=list** command to list the filter rules that are supplied with OS/390 Firewall Technologies. Carefully consider these filter rules that are supplied by OS/390 Firewall Technologies using this command, then define all special IP filter rules you are going to use with your firewall.

See Chapter 8, “Controlling Traffic Through the Firewall with Filters” on page 51 for more information about how filtering works.

Fragmentation: Although previously documented, the **fragment** parameter is no longer applicable. All packet fragments will be reassembled before any filtering is performed. Objects which currently include the **fragment** parameter do not need to be changed. The **fragment** parameter may be specified when using the **fwfrule** command.

fragment={yes|no|only|headers}

IP fragment control code. This code is interpreted as follows, for the purposes of determining if an IP packet matches the rule:

Yes The rule will match fragment headers, fragments and non-fragments. For fragments, the port information will be ignored and assumed to match.

No Only non-fragments can match. Fragment headers and fragments are excluded by this parameter.

Only Only fragments and fragment headers can match. For fragment headers, port information must match. For fragments, port information will be ignored.

Headers Only non-fragments and fragment headers can match. Fragments are excluded by this parameter.

If this parameter is not specified, the default for both PERMIT and DENY rules is Yes.

Using the Operation and Port Number Operands: The logical operation and port number operands are used together. The source *opcode* and destination *opcode* operands are logical operation that state a relationship between the port number (destination or origin) for the packet and the source *portno* and destination *portno* operands. For example, if the packet destination port is port 20, and the destination *opcode* and destination *portno* operands are “ge 15,” the packet matches because port 20 is greater than or equal to port 15.

If you use a source *opcode* or destination *opcode* with the value **any**, the filter does not look at the port number; **any** means that any port will match. The port number must be zero and cannot be changed unless the *opcode* is changed.

Some of the more important ports to protect are:

Port	Use
20	FTP data
21	FTP control
23	Telnet
25	Mail

53 Domain Name Server

70 Gopher

80 HTTP

111 RPC

1014 Configuration Server

1080 socks

In the rule values, note these constraints:

- The values for the port numbers must be in the range 0 through 65535.
- Your port number will never be less than 1 or greater than 65535.

OSPF Protocol: When OSPF is specified as the protocol, the source port operation and source port value is used for the OSPF record type value. Filtering can also be performed on the OSPF type. A type value of **any** can be specified for the *srcopcode*, and the destination port fields must be specified as **any 0**. Anything else is ignored. (Refer to RFC 1583 for OSPF types and codes.)

ICMP Protocol: For the ICMP protocol, rather than specifying a source port, specify an ICMP type and in place of a destination port, specify an ICMP code. The logical operator specified is applied to the type or code and, as for ports, an operator of any, means that any type or code value will match the rule.

The values for source *opcode*, destination *opcode*, and *ICMP opcode* are:

eq Equal to

neq Not equal to

lt Less than

gt Greater than

le Less than or equal to

ge Greater than or equal to

Here are some of the ICMP types and codes:

Type	Code Description
0	0 - Ping reply
8	0 - Ping request
3	1 - Host unreachable
3	3 - Port unreachable
5	1 - Redirect for host

Examples

1. Add some filter rules.

```
    User:   fwfrule cmd=add name="Example Filter Rule 1" desc="Example 1" type=permit \
            protocol=tcp interface=both routing=both direction=both
  System:
    User:   fwfrule cmd=add name="Example Filter Rule 2" desc="Example 2" type=deny \
            protocol=udp interface=secure routing=local direction=outbound srcopcode=gt \
            srcport=1024
  System:
```

2. Add a filter rule for an anchor.

```
User:   fwfrule cmd=add name="Example Filter Rule 3" desc="Example 3" type=anchor \
        protocol=all srcopcode=any srcport=0 destopcode=any destport=0 interface=both \
        routing=both log=no tunnel=501
System:
```

3. Show these filter rules.

```
User:   fwfrule cmd=list id=517
System:
                id = 517
              type = permit
              name = Example Filter Rule 1
              desc = Example 1
          protocol = tcp
         srcopcode = any
           srcport = 0
        destopcode = any
          destport = 0
         interface = both
           routing = both
         direction = both
               log =
            tunnel =
          fragment =

User:   fwfrule cmd=list id=518
System:
                id = 518
              type = deny
              name = Example Filter Rule 2
              desc = Example 2
          protocol = udp
         srcopcode = gt
           srcport = 1024
        destopcode = any
          destport = 0
         interface = secure
           routing = local
         direction = outbound
               log =
            tunnel =
          fragment =

User:   fwfrule cmd=list id=519
System:
                id = 519
              type = anchor
              name = Example Filter Rule 3
              desc = Example 3
          protocol = all
         srcopcode = any
           srcport = 0
        destopcode = any
          destport = 0
         interface = both
           routing = both
               log = no
            tunnel = 501
```

4. Change a filter rule so that it references a tunnel, then show the updated rule in raw format.

```
User:   fwfrule cmd=change id=518 tunnel=10 log=yes
System:
User:   fwfrule cmd=list id=518 format=raw
System: 518|deny|Example Filter Rule 2|Example 2|udp|gt|1024|any|0|secure|local|outbound|yes|10|
```

5. Delete the filter rules.

User: **fwfrule cmd=delete id=517**
System:
User: **fwfrule cmd=delete id=518**
System:

fwkern

Purpose

Starts, stops and queries first the firewall kernel and then, optionally, the firewall servers.

Note: This command runs only from the OS/390 operator console.

Format

```
modiFy fwkern,[start|stop|query] server
modiFy fwkern,query level
stoP   fwkern
Start  fwkern [, parms='–nofw']
Cancel fwkern
```

Keywords and Parameters

modiFy fwkern,[start|stop|query] *server*
sends a modify request to start, stop, or query the specified server(s). The following may be specified:

all all of the configured firewall servers.

cfgsrv
the configuration server (if configured).

sockd
the Socks server (if configured).

pftpd
the FTP proxy server (if configured).

fwstackd
the **fwstackd** server (if configured) and IPSec initialization (tunnels, filters, NAT, and RealAudio).

isakmpd
the ISAKMP server (if configured).

modiFy fwkern,query level
queries the Firewall level. The version, release, modification level, service level and creation date are displayed.

stoP fwkern
terminates the firewall program and stops any servers that are configured in the server configuration file.

Start fwkern
starts the firewall program and any servers that are configured in the server configuration file.

Start fwkern, parms=*'–nofw'*
starts the firewall program without starting any servers.

Cancel fwkern
Cancels the firewall kernel address space. **Using this parameter is not recommended, as it may leave your system in an indeterminate state**.

Usage

Control the FWKERN started process from the OS/390 operator console using the **start**, **stop**, and **modify** commands.

Note: To activate firewall functions, the firewall kernel should be started after the firewall TCP/IP stacks are started. (The firewall TCP/IP stacks are defined through the command “fwstack” on page 311.) The firewall kernel will wait for *at least one* firewall TCP/IP stack to start before performing server initialization. Once started, the FWSTACK server listens for the firewall TCP/IP stacks to start and stop.

When started, the FWSTACKD reactivates any filters, tunnels and NAT functions that were active when TCP/IP and FWKERN were stopped. This can include any or all of the following tasks:

- reactivating filter logging
- reactivating NAT logging
- reactivating NAT
- reactivating tunnels
- reactivating filters
- reactivating RealAudio

Notes:

1. Tunnels reactivated by FWSTACKD are given the full timeout value specified when the tunnel definition was created.
2. IPSec initialization will not occur if FWKERN fails to start FWSTACKD.

Examples

1. Start the firewall without starting any servers:

   User: **s fwkern,parms='-nofw'**

   System:

```
$HASP373 FWKERN   STARTED
IEF403I FWKERN - STARTED - TIME=09.17.41
ICAM1057i Release 2.10.0 Service Level 0000000 Created
          on Apr 18 2000.
ICAM1003i FWKERN initialization complete.
```

2. Start each firewall server that was configured using the fwdaemon command:

User: **s fwkern**

System:

```
$HASP373 FWKERN   STARTED
IEF403I FWKERN - STARTED - TIME=09.26.54
ICAM1057i Release 2.10.0 Service Level 0000000 Created
          on Apr 18 2000.
$HASP373 ICAPSOCK STARTED
IEF403I ICAPSOCK - STARTED - TIME=09.27.03
ICAM1069i Daemon SOCKD has been started.
$HASP373 ICAPPFTP STARTED
IEF403I ICAPPFTP - STARTED - TIME=09.27.09
ICAM1069i Daemon PFTPD has been started.
$HASP373 ICAPCFGS STARTED
IEF403I ICAPCFGS - STARTED - TIME=09.27.13
ICAM1069i Daemon CFGSRV has been started.
$HASP373 ICAPIKED STARTED
IEF403I ICAPIKED - STARTED - TIME=09.27.33
ICAM1069i Daemon ISAKMPD has been started.
$HASP373 ICAPSTAK STARTED
IEF403I ICAPSTAK - STARTED - TIME=09.27.56
ICAM1069i Daemon FWSTACKD has been started.
ICAM1003i FWKERN initialization complete.
ICAM1081i IP/SEC initialization complete for CS390IP
```

3. Stop the firewall and each server that was started:

User: **p fwkern**

System:

```
ICAM1004i FWKERN has received STOP command.
IEF404I ICAPSTAK - ENDED - TIME=10.10.04
$HASP395 ICAPSTAK ENDED
ICAM1000i Firewall daemon FWSTACKD has stopped.
IEF404I ICAPIKED - ENDED - TIME=10.10.06
$HASP395 ICAPIKED ENDED
ICAM1000i Firewall daemon ISAKMPD has stopped.
IEF404I ICAPCFGS - ENDED - TIME=10.10.08
$HASP395 ICAPCFGS ENDED
ICAM1000i Firewall daemon CFGSRV has stopped.
IEF404I ICAPPFTP - ENDED - TIME=10.10.09
$HASP395 ICAPPFTP ENDED
ICAM1000i Firewall daemon PFTPD has stopped.
IEF404I ICAPSOCK - ENDED - TIME=10.10.12
$HASP395 ICAPSOCK ENDED
ICAM1000i Firewall daemon SOCKD has stopped.
IEF404I FWKERN - ENDED - TIME=10.10.16
$HASP395 FWKERN   ENDED
```

4. Start each of the firewall servers, one at a time:

```
   User:    s fwkern,parms='-nofw'
 System:
                        $HASP373 FWKERN   STARTED
                        IEF403I FWKERN - STARTED - TIME=10.12.56
                        ICAM1057i Release 2.10.0 Service Level 0000000 Created
                                  on Apr 18 2000.
                        ICAM1003i FWKERN initialization complete.
   User:    f fwkern,start sockd
 System:           $HASP373 ICAPSOCK STARTED
                        IEF403I ICAPSOCK - STARTED - TIME=10.14.39
                        ICAM1069i Daemon SOCKD has been started.
   User:    f fwkern,start pftpd
 System:           $HASP373 ICAPPFTP STARTED
                        IEF403I ICAPPFTP - STARTED - TIME=10.15.02
                        ICAM1069i Daemon PFTPD has been started.
   User:    f fwkern,start cfgsrv
 System:           $HASP373 ICAPCFGS STARTED
                        IEF403I ICAPCFGS - STARTED - TIME=10.15.47
                        ICAM1069i Daemon CFGSRV has been started.
   User:    f fwkern,start isakmpd
 System:           $HASP373 ICAPIKED STARTED
                        IEF403I ICAPIKED - STARTED - TIME=10.15.47
                        ICAM1069i Daemon ISAKMPD has been started.
   User:    f fwkern,start fwstackd
 System:           $HASP373 ICAPSTAK STARTED
                        IEF403I ICAPSTAK - STARTED - TIME=10.15.59
                        ICAM1069i Daemon FWSTACKD has been started.
```

5. Query the status of the servers:

```
   User:    f fwkern,query all
 System:
                        ICAM1001i Firewall daemon SOCKD status is READY and
                                  process id is 33554447.
                        ICAM1100i Current child process status for SOCKD:
                                  Subtask    PID       Clients   MAX Clients
                                  ------- ---------- ----------- -----------
                                     1      50331665      0          151
                                     2      83886101      0          150
                        ICAM1001i Firewall daemon PFTPD status is READY and
                                  process id is 33554448.
                        ICAM1100i Current child process status for
                                  Subtask    PID       Clients   MAX Clients
                                  ------- ---------- ----------- -----------
                                     1      50331675      0        150
                                     2      33554452      0        150
                        ICAM1001i Firewall daemon CFGSRV status is READY and
                                  process id is 33554454.
                        ICAM1001i Firewall daemon ISAKMPD status is READY and
                                  process id is 33554456.
                        ICAM1001i Firewall daemon FWSTACKD status is READY and
                                  process id is 33554458.
```

6. Query the Firewall level:

```
   User:    f fwkern,query level
 System:           ICAM1057i Release 2.10.0 Service Level 0000000. Created
                                  on Apr 18 2000.
```

fwkeypol

Purpose

Configures key policies.

Format

```
fwkeypol cmd=add
         name=name
        [desc=description]
        [imode={main|aggressive}]
        [rmode={main|aggressive|any}]
         keyprop=key_proposal_id_or_name

         defaults: imode=main, rmode=main

fwkeypol cmd=change
         keypol=key_policy_id_or_name
        [desc=[description]]
        [imode={main|aggressive}]
        [rmode={main|aggressive|any}]
        [keyprop=key_proposal_id_or_name]

fwkeypol cmd=delete
         keypol=key_policy_id_or_name

fwkeypol cmd=list
        [keypol=key_policy_id_or_name]
        [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds a key policy.

name=*name*
the name of the key policy. Maximum length is 100 characters.

desc=*description*
the description of the key policy. Maximum length is 100 characters. This parameter is optional.

imode={main|aggressive}
initiator key management negotiation mode. **Main** mode indicates that identity protection is used when key negotiations are initiated by this system. **Agressive** mode indicates that identity protection is not used when key negotiations are initiated by this system. The default is **main**.

Note: Refer to Appendix H, “Initializing Security Associations with ISAKMP and IKE” on page 463 for a more detailed description of **Main** and **Aggressive** mode.

rmode={main|aggressive|any}
responder key management negotiation mode. **Main** mode requires remote systems to initiate key negotiations using identity protection. **Aggressive** mode requires remote systems to initiate key

negotiations without using identity protection. **Any** mode allows remote systems to initiate key exchange negotiations with or without identity protection. The default is **main**.

keyprop=*key_proposal_id_or_name*
ID or name of the key proposal to use when this system negotiates keys.

cmd=change
changes a key policy. The parameters are the same as for **add**, except that **keypol** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a key policy.

keypol=*key_policy_id_or_name*
the ID or name of the key policy to delete.

cmd=list
lists one or all key policies.

keypol=*key_policy_id_or_name*
the ID or name of the key policy to list. If **keypol** is not specified, all key policies are listed. This parameter is optional.

format={short|long|wide|raw}
the format of the listing. If **keypol** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`imode = imode`
	`rmode = rmode`
	`keyprop = keyproposalid`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars in the following order:
	`id|name|desc|imode|rmode|keyprop`
	For example:
	`501|kpolicy1|key policy desc.|main|main|502`

Usage

The key policy is an association of initiator and responder modes with a key proposal. Before using the **fwkeypol** command, you must define a key proposal using the **fwkeyprop** command.

The key transform that is used to initiate a key negotiation in aggressive mode is not negotiated, and therefore only one transform can be specified. Therefore, when **imode = aggressive**, the first key transform in the key proposal will be the only transform offered in the negotiation.

Use the **fwkeypol cmd=list** command to list the key policy that is supplied with OS/390 Firewall Technologies. Carefully consider the listed key policy, then define all other key policies needed on this system.

After a key policy is defined, associate it with a key server group using the **fwkeysrvgrp** command.

The key policy that is shipped with the OS/390 Firewall Technologies is used by the ISAKMP daemon as the default key policy until the identity of the requestor can be verified and the correct key policy can be determined. The default key policy as shipped can be changed to reflect the security parameters desired for your installation. The default key policy cannot be deleted.

Examples

1. Create a key policy. This key policy will use the default main mode for both the initiator key management negotiation mode (imode) and the responder key management negotiation mode (rmode). When main mode is specified for imode, it indicates that identity protection will be used when key negotiations are initiated by the local system. When main mode is specified for rmode, it indicates that remote systems will be required to initiate key negotiations using identity protection.

```
   User:  fwkeypol cmd=add name="Key Policy A" desc="This is key policy A" \
          keyprop="Key Proposal 1"
System:
   User:  fwkeypol cmd=list format=long
System:
                     id = 501
                   name = Key Policy A
                   desc = This is key policy A
                  imode = main
                  rmode = main
                keyprop = 501
```

2. Create another key policy. This key policy will use aggressive mode for its initiator key management negotiation mode (imode), which indicates that the local system will not use identity protection when initiating key negotiations. This key policy will use main mode for its responder key management negotiation mode (rmode), which indicates that it will require remote systems to initiate key negotiations using identity protection.

```
   User:  fwkeypol cmd=add name="Key Policy B" desc="This is key policy B" \
          keyprop="Key Proposal 2" imode=aggressive rmode=main
System:
   User:  fwkeypol cmd=list keypol="Key Policy B"
System:
                     id = 502
                   name = Key Policy B
                   desc = This is key policy B
                  imode = aggressive
                  rmode = main
                keyprop = 502
```

3. Change a key policy's responder key management negotiation mode so that it uses any mode, meaning that it will allow remote systems to initiate key exchange negotiations with or without identity protection.

```
   User:  fwkeypol cmd=change keypol="Key Policy A" rmode=any
System:
   User:  fwkeypol cmd=list keypol="Key Policy A"
System:
                     id = 501
                   name = Key Policy A
                   desc = This is key policy A
                  imode = main
                  rmode = any
                keyprop = 501
```

fwkeyprop

Purpose

Configures key proposals.

Format

```
fwkeyprop cmd=create
          name=name
         [desc=description]
          keytranlist=key_transform_id_or_name[,...]

fwkeyprop cmd=change
          keyprop=key_proposal_id_or_name
         [desc=[description]]
         [keytranlist=key_transform_id_or_name[,...]]

fwkeyprop cmd=delete
          keyprop=key_proposal_id_or_name

fwkeyprop cmd=list
         [keyprop=key_proposal_id_or_name]
         [format={short|long|wide|raw}]

fwkeyprop cmd=add
          keyprop=key_proposal_id_or_name
          keytranlist=key_transform_id_or_name[,...]

fwkeyprop cmd=remove
          keyprop=key_proposal_id_or_name
          keytranlist=key_transform_id_or_name[,...]

fwkeyprop cmd=move
          keyprop=key_proposal_id_or_name
          keytran=key_transform_id_or_name
          after=[key_transform_id_or_name]
```

Keywords and Parameters

cmd=create
creates a key proposal.

name=*name*
the name of the key proposal. Maximum length is 100 characters.

desc=*description*
the description of the key proposal. Maximum length is 100 characters. This parameter is optional.

keytranlist=*key_transform_id_or_name***[,...]**
IDs or names of the key transforms to apply to this key proposal. If more than one is specified, separate them with commas.

The key transforms must be specified in preferred order, with the most preferred key transform specified first.

cmd=change
changes a key proposal. The parameters are the same as for **create**, except that **keyprop** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a key proposal.

keyprop=*key_proposal_id_or_name*
the ID or name of the key proposal to delete.

cmd=list
lists one or all key proposals.

keyprop=*key_proposal_id_or_name*
the ID or name of the key proposal to list. If **keyprop** is not specified, all key proposals are listed. This parameter is optional.

format={short|long|wide|raw}
the format of the listing. If **keyprop** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`keytranlist = keytransformid1,...,keytransformidn`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|keytranlist`
	For example:
	`501|keyprop1|proposal desc.|501,502,503`

cmd=add
adds a list of key transforms to a key proposal. The specified key transforms are added to the end of the key proposal's current list.

keyprop=*key_proposal_id_or_name*
the ID or name of the key proposal.

keytranlist=*key_transform_id_or_name***[,...]**
IDs or names of the key transforms to add to the key proposal. If more than one is specified, separate them with commas.

The key transforms must be specified in preferred order, with the most preferred key transform specified first.

cmd=remove
removes a list of key transforms from a key proposal.

keyprop=*key_proposal_id_or_name*
the ID or name of the key proposal.

keytranlist=*key_transform_id_or_name***[,...]**
IDs or names of the key transforms to remove from the key proposal. If more than one is specified, separate them with commas.

Note: A key proposal must reference at least one key transform. If the list of key transforms contains all remaining key transforms, an error message is issued and nothing is removed.

cmd=move
moves a key transform to a different place in a key proposal's list of key transforms.

keyprop=*key_proposal_id_or_name*
the ID or name of the key proposal.

keytran=*key_transform_id_or_name*
ID or name of the key transform to move.

after=*key_transform_id_or_name*
ID or name of the key transform after which to place the above specified key transform. If a blank value is specified, the key transform is moved to the top of the list. To see the current order of the key transforms, issue a **cmd=list** with any format.

Usage

The key proposal is an ordered list of key transforms. This ordering is important when acting as an initiator of a dynamic connection. In this case, the key transforms are sent to the remote key server in initiator's order of preference as defined in the key proposal definition. When acting as responder, the initiator's ordering takes precedence. Before using the **fwkeyprop** command, you must define one or more key transforms using the **fwkeytran** command.

Use the **fwkeyprop cmd=list** command to list the key proposal that is supplied with OS/390 Firewall Technologies. Carefully consider the listed key proposal, then define all other key proposals needed on this system.

After a key proposal is defined, associate it with a key policy using the **fwkeypol** command.

Examples

The following key transforms will be used in examples:

```
501   Key Transform 1
502   Key Transform 2
503   Key Transform 3
```

1. Create two key proposals.

```
User:   fwkeyprop cmd=create name="Key Proposal A" desc="This is key proposal A" \
        keytranlist="Key Transform 1"
System:
User:   fwkeyprop cmd=create name="Key Proposal B" desc="This is key proposal B" \
        keytranlist="Key Transform 2,Key Transform 3"
System:
User:   fwkeyprop cmd=list format=long
System:
                id = 501
              name = Key Proposal A
              desc = This is key proposal A
        keytranlist = 501

                id = 502
              name = Key Proposal B
              desc = This is key proposal B
        keytranlist = 502,503
```

2. Change the key transform list of a key proposal.

```
User:   fwkeyprop cmd=change keyprop="Key Proposal A" keytranlist="Key Transform 2"
System:
User:   fwkeyprop cmd=list keyprop="Key Proposal A"
System:
                id = 501
              name = Key Proposal A
              desc = This is key proposal A
        keytranlist = 502
```

3. Add key transforms to a key proposal's transform list.

```
User:   fwkeyprop cmd=add keyprop="Key Proposal A" \
        keytranlist="Key Transform 1,Key Transform 3"
System:
User:   fwkeyprop cmd=list keyprop="Key Proposal A"
System:
                id = 501
              name = Key Proposal A
              desc = This is key proposal A
        keytranlist = 502,501,503
```

4. Move a key transform to the end of a key proposal's transform list.

```
User:   fwkeyprop cmd=move keyprop="Key Proposal A" keytran="Key Transform 2" \
        after="Key Transform 3"
System:
User:   fwkeyprop cmd=list keyprop="Key Proposal A"
System:
                id = 501
              name = Key Proposal A
              desc = This is key proposal A
        keytranlist = 501,503,502
```

5. Remove key transforms from a key proposal's transform list.

```
User:   fwkeyprop cmd=remove keyprop="Key Proposal A" \
        keytranlist="Key Transform 1,Key Transform 2"
System:
User:   fwkeyprop cmd=list keyprop="Key Proposal A"
System:
                id = 501
              name = Key Proposal A
              desc = This is key proposal A
        keytranlist = 503
```

fwkeyring

Purpose

Configures key rings.

Format

```
fwkeyring cmd=change
          keyring=userid/keyringname

fwkeyring cmd=list
```

Keywords and Parameters

cmd=change
changes the name of the key ring used by the ISAKMP daemon.

keyring=*userid/keyringname*
the new key ring name.

The user ID is the ID which, according to RACF, owns the key ring. The user ID should be specified in uppercase characters. This command will change a lower case user ID to uppercase. The user ID must be separated from the keyring name by a slash, '/'. The *userid/keyringname* value must not contain an asterisk, '*', and may be up to 237 characters long. The keyring name is case sensitive.

cmd=list
lists the current key ring name.

Usage

Use **fwkeyring** to change or list the name of the key ring that the ISAKMP daemon uses when it is started.

The default key ring name supplied with OS/390 Firewall Technologies is **FWKERN/ikekeyring**.

Because only one key ring name is used and a default is provided with the product, only **change** and **list** subcommands are provided on the **fwkeyring** command.

Note: If the **change** command is issued to change the name of the key ring, the ISAKMP server must be stopped and restarted. This is required in order for the new key ring name to be recognized.

Examples

1. Change the key ring used by the ISAKMP daemon.

```
   User:    fwkeyring cmd=change keyring=MYUSER/mykeyringname
System:
   User:    fwkeyring cmd=list
System:
                    keyring = MYUSER/mykeyringname
```

fwkeysrvgrp

Purpose

Configures a key server group.

Format

```
fwkeysrvgrp cmd=create
            name=name
           [desc=description]
            keypol=key_policy_id_or_name
            lockeysrv=key_server_id_or_name
            remkeysrvlist=key_server_id_or_name[,...]

fwkeysrvgrp cmd=change
            keysrvgrp=key_server_group_id_or_name
           [desc=[description]]
           [keypol=key_policy_id_or_name]
           [lockeysrv=key_server_id_or_name]
           [remkeysrvlist=key_server_id_or_name[,...]]

fwkeysrvgrp cmd=delete
            keysrvgrp=key_server_group_id_or_name

fwkeysrvgrp cmd=list
           [keysrvgrp=key_server_group_id_or_name]
           [format={short|long|wide|raw}]

fwkeysrvgrp cmd=move
            keysrvgrp=key_server_group_id_or_name
            after=[key_server_group_id_or_name]

fwkeysrvgrp cmd=addremkeysrv
            keysrvgrp=key_server_group_id_or_name
            remkeysrvlist=key_server_id_or_name[,...]

fwkeysrvgrp cmd=removeremkeysrv
            keysrvgrp=key_server_group_id_or_name
            remkeysrvlist=key_server_id_or_name[,...]
```

Keywords and Parameters

cmd=create
creates a key server group.

name=*name*
the name of the key server group. Maximum length is 100 characters.

desc=*description*
the description of the key server group. Maximum length is 100 characters. This parameter is optional.

keypol=*key_policy_id_or_name*
ID or name of the key policy used between the local and remote key servers.

lockeysrv=*key_server_id_or_name*
specifies the ID or name of the key server that is local to the system being configured. This is the encryption/decryption point - tunnel endpoint.

remkeysrvlist=*key_server_id_or_name***[,...]**
the list of the remote key servers to be associated with this local key server. This is an unordered list of key servers.

cmd=change
changes a key server group. The parameters are the same as for **create**, except that **keysrvgrp** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a key server group.

keysrvgrp=*key_server_group_id_or_name*
the ID or name of the key server group to delete.

cmd=list
lists one or all key server groups.

keysrvgrp=*key_server_group_id_or_name*
the ID or name of the key server group. If **keysrvgrp** is not specified, all key server groups are listed. This parameter is optional.

format={short|long|raw|wide}
the format of the listing. If **keysrvgrp** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`keypol = keypolicyid`
	`lockeysrv = localkeyserverid`
	`remkeysrvlist = remotekeyserverid1,...,remotekeyserveridn`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|keypol|lockeysrv|remkeysrvlist`
	For example,
	`501|ksg1|ksg desc.|501|512|504,505,506,517`

cmd=move
moves a key server group within the list of all key server groups.

keysrvgrp=*key_server_group_id_or_name*
the ID or name of the key server group to be moved.

after=[*key_server_group_id_or_name***]**
ID or name of the key server group after which to place the above specified key server group. If a blank value is specified, the key server group is moved to the top of the list. To see the current order of the key server groups, issue a **cmd=list** with any format.

cmd=addremkeysrv
adds a list of remote key servers to a key server group. The specified remote key servers are added to the end of the key server group's current list.

keysrvgrp=*key_server_group_id_or_name*
the ID or name of the key server group to which the remote key servers are added.

remkeysrvlist=*key_server_id_or_name***[,...]**
the IDs or names of the remote key servers to add to the remote key server list.

cmd=removeremkeysrv
removes a list of remote key servers from a key server group.

keysrvgrp=*key_server_group_id_or_name*
the ID or name of the key server group from which the remote key servers are removed.

remkeysrvlist=*key_server_id_or_name***[,...]**
the IDs or names of the remote key servers to remove from the remote key server list.

Usage

A key server group is an association of local and remote ISAKMP servers with a key policy. It is used when negotiating a security association between the key server and one of the listed remote servers.

Key server groups are ordered among themselves. This ordering is important when locating the key policy to be used when acting in responder mode. When in initiator mode, the ordering is important when accepting the default key server group on a dynamic connection. Searching is always performed by the order they appear in the **list** command. Therefore, the **cmd=move** command allows the user to move the entire key server group relative to all other key server groups.

The remote key servers listed within a key server group are not ordered. This is a list of the remote key servers that are associated with the local key server and which will use the specified key policy. Therefore, the user may add or remove key servers from the list, but there is no need to order them.

After a key server group is defined, associate it with a dynamic connection using the **fwdynconns** command.

When defining a key server group, the local key server must have an associated IP address to be able to initiate a connection.

Examples

The following key policies will be used in examples:

```
501   Key Policy 1
502   Key Policy 2
503   Key Policy 3
```

The following key servers will be used in examples:

```
501   Key Server 1
502   Key Server 2
503   Key Server 3
504   Key Server 4
505   Key Server 5
```

1. Create two key server groups.

```
   User:   fwkeysrvgrp cmd=create name="Key Server Group A" \
           desc="This is key server group A" keypol="Key Policy 1" \
           lockeysrv="Key Server 1" remkeysrvlist="Key Server 2,Key Server 3"
System:
   User:   fwkeysrvgrp cmd=create name="Key Server Group B" \
           desc="This is key server group B" keypol="Key Policy 2" \
           lockeysrv="Key Server 4" remkeysrvlist="Key Server 5"
System:
   User:   fwkeysrvgrp cmd=list format=long
System:
                 id = 502
               name = Key Server Group B
               desc = This is key server group B
            keypol = 502
         lockeysrv = 504
      remkeysrvlist = 505

                 id = 501
               name = Key Server Group A
               desc = This is key server group A
            keypol = 501
         lockeysrv = 501
      remkeysrvlist = 502,503
```

2. Change the key policy used by a key server group.

```
   User:   fwkeysrvgrp cmd=change keysrvgrp="Key Server Group A" keypol="Key Policy 3"
System:
   User:   fwkeysrvgrp cmd=list keysrvgrp="Key Server Group A"
System:
                 id = 501
               name = Key Server Group A
               desc = This is key server group A
            keypol = 503
         lockeysrv = 501
      remkeysrvlist = 502,503
```

3. Move a key server group to the end of the list of key server groups.

```
   User:   fwkeysrvgrp cmd=move keysrvgrp="Key Server Group B" after="Key Server Group A"
System:
   User:   fwkeysrvgrp cmd=list
System:    501   Key Server Group A         This is key server group A
           502   Key Server Group B         This is key server group B
```

4. Add remote key servers to a key server group's key server list.

```
   User:   fwkeysrvgrp cmd=addremkeysrv keysrvgrp="Key Server Group B" \
           remkeysrvlist="Key Server 2,Key Server 3"
System:
   User:   fwkeysrvgrp cmd=list keysrvgrp="Key Server Group B"
System:
                 id = 502
               name = Key Server Group B
               desc = This is key server group B
            keypol = 502
         lockeysrv = 504
      remkeysrvlist = 505,502,503
```

5. Remove a remote key server from a key server group's key server list.

User: **fwkeysrvgrp cmd=removeremkeysrv keysrvgrp="Key Server Group B" **
remkeysrvlist="Key Server 5"

System:

User: **fwkeysrvgrp cmd=list keysrvgrp="Key Server Group B"**

System:

```
                id = 502
              name = Key Server Group B
              desc = This is key server group B
            keypol = 502
         lockeysrv = 504
     remkeysrvlist = 502,503
```

fwkeysrv

Purpose

Configures key servers.

Format

```
fwkeysrv cmd=add
         name=name
        [desc=description]
         idtype={IPV4|FQDN|USER@FQDN|X500_DN}
         authid=string
        [{ipaddr=x.x.x.x or
         hostname=string}]

fwkeysrv cmd=change
         keysrv=key_server_id_or_name
        [desc=[description]]
        [idtype={IPV4|FQDN|USER@FQDN|X500_DN}]
        [authid={string}]
        [ipaddr=[x.x.x.x]]
        [hostname=[string]]

fwkeysrv cmd=delete
         keysrv=key_server_id_or_name

fwkeysrv cmd=list
        [keysrv=key_server_id_or_name]
        [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds a key server.

name=*name*
the name of the key server. Maximum length is 100 characters.

desc=*description*
the description of the key server. Maximum length is 100 characters. This parameter is optional.

idtype={IPV4|FQDN|USER@FQDN|X500_DN}
the type of the ID specified in **authid**.

IPV4 an IP address in dotted-decimal format. For example: `1.2.3.4`

FQDN a fully qualified domain name. For example: `vnet.ibm.com`

USER@FQDN
a user at a fully qualified domain name. The user name cannot contain a blank. For example: `ibm@vnet.ibm.com`

X500_DN an X.500 distinguished name (DN). The individual relative distinguished names (RDNs) must be specified as: **keyword=value**. Use commas to separate RDNs within a DN.

Use double quotes to surround an RDN's value in the following cases:

- when the value contains any of the following characters: ",", "+", "=", """, "\n", "\r", "<", ">", "#", or ";".
- when the value contains leading or trailing blanks.
- when the value contains consecutive blanks.

When using double quotes around a value within a double quoted DN, use the escape sequence (\") so that the double quote character is kept as part of the value. Otherwise it indicates the end of the DN string.

The RDNs accepted are:

Figure 75.

Keyword	Maximum value length	Description
CN	64 characters	Common Name
T	64 characters	Title
OU	64 characters	Organizational Unit name
O	64 characters	Organization name
L	128 characters	Locality
ST	128 characters	State or Province
C	3 characters	Country

Note: To be compatible with RACF, SP is also acceptable as the keyword for State or Province. If SP is specified, it is stored as ST and ST is shown on subsequent list requests.

All keywords do not need to be used. Keywords specified must be in the above listed order. The organizational unit name (OU) keyword may be repeated as many times as needed. If multiple OU keywords are specified, they must be specified consecutively.

The maximum length accepted for a DN is 1024 characters. including the keywords "=" characters, values, and separating commas.

```
authid="CN=John Smith, OU=\"   Raw Materials  \",OU=Purchasing Dept.,
        O=Acme Manufacturing,ST=NY,C=usa"
```

authid=*string*
the string identifier of the key server. This can be represented as an IPV4, FQDN, USER@fQDN, or X500_DN name. This field is case sensitive.

ipaddr=*address*
the real IP address of the key server. This must be entered in **IPV4** format. If an IP address is specified, then a **hostname** may not be specified.

hostname=*string*
the fully qualified domain name of the key server. This must be entered as an **FQDN** string (for example, keyserver.ibm.com). The maximum length accepted for a **hostname** is 512 characters. If a **hostname** is specified, then an IP address may not be specified. This field is case sensitive.

cmd=change
changes a key server. The parameters are the same as for **add**, except that **keysrv** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a key server.

keysrv=*key_server_id_or_name*
the ID or name of the key server to delete.

cmd=list
lists one or all key servers.

keysrv=*key_server_id_or_name*
the ID or name of the key server. If **keysrv** is not specified, all key servers are listed. This parameter is optional.

format={short|long|raw|wide}
the format of the listing. If **keysrv** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	formatted as keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`idtype = idtype`
	`authid = string`
	`ipaddr = address`
	`hostname = string`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|idtype|authid|ipaddr|hostname`
	For example,
	`501|KeySrv1|server desc.|fqdn|keyserver.ibm.com||keyserver.endicott.ibm`
	or
	`502|KeySrv2|server desc.|fqdn|keyserver.ibm.com|9.130.47.4`

Usage

A key server specifies the address and identification information used to authenticate a key server.

An **authid/idtype** is required for all key server objects. If the key server object is defining an ISAKMP server from which connections may be initiated, then **authid/idtype** and **ipaddr** or **hostname** are required. If the key server object is defining an ISAKMP server that only acts in responder mode, then only the **authid/idtype** is required.

After a key server is defined, it may be used wherever a key server is required (**fwkeysrvgrp**, **fwauthinfo**, **fwdynconns** and **fwondemand** commands).

Examples

1. Create an ISAKMP key server definition for a remote key server for which the local key server will act as the initiator using an IP address in dotted-decimal format (IPV4) for authentication and a host name to locate the remote key server.

```
   User:  fwkeysrv cmd=add name="Key Server A" desc="This is key server A" idtype=IPV4 \
          authid=1.2.3.4 hostname=keyserver.ibm.com
System:
   User:  fwkeysrv cmd=list format=long
System:
                    id = 501
                  name = Key Server A
                  desc = This is key server A
                idtype = IPV4
                authid = 1.2.3.4
                ipaddr =
              hostname = keyserver.ibm.com
```

2. Create a local ISAKMP key server that is authenticated using a fully qualified domain name (FQDN).

```
   User:  fwkeysrv cmd=add name="Key Server B" desc="This is key server B" idtype=FQDN \
          authid=mytown.mycompany.com ipaddr=99.100.50.250
System:
   User:  fwkeysrv cmd=list keysrv="Key Server B"
System:
                    id = 502
                  name = Key Server B
                  desc = This is key server B
                idtype = FQDN
                authid = mytown.mycompany.com
                ipaddr = 99.100.50.250
              hostname =
```

3. Change an ISAKMP key server so that it is authenticated using an X.500 distinguished name (X500_DN). Also remove the IP address from the definition. Without an ipaddr or hostname specified, this key server can only be used for initiating negotiations.

```
   User:  fwkeysrv cmd=change keysrv="Key Server B" idtype=X500_DN \
          authid="CN=My Name,T=My Title,O=My Organization,L=My Locality,ST=My State" \
          ipaddr=
System:
   User:  fwkeysrv cmd=list keysrv="Key Server B"
System:
                    id = 502
                  name = Key Server B
                  desc = This is key server B
                idtype = X500_DN
                authid = CN=My Name,T=My Title,O=My Organization,L=My Locality,ST=My State
                ipaddr =
              hostname =
```

fwkeytran

Purpose

Configures key transforms.

Format

```
fwkeytran cmd=add
          name=name
         [desc=description]
         [prot={IKE}]
          authmeth={SHAREDKEY|RSASIG}
          hashalg={MD5|SHA}
          encralg={DES_CBC_8|3DES_CBC}
         [dhgrp={GROUP1|GROUP2}]
         [itime=minutes]               (minutes:   1-9999)
         [isize=kb]                    (kilobytes: 0-4194300)
         [rtime=minminutes-maxminutes](minutes:   1-9999)
         [rsize={minkb-maxkb|0|0-0}]  (kilobytes: 0-4194300)

          defaults: prot=IKE, dhgrp=GROUP1, itime=480, isize=0, rtime=60-1440, rsize=0

fwkeytran cmd=change
          keytran=key_transform_id_or_name
         [desc=[description]]
         [prot={IKE}]
         [authmeth={SHAREDKEY|RSASIG}]
         [hashalg={MD5|SHA}]
         [encralg={DES_CBC_8|3DES_CBC}]
         [dhgrp={GROUP1|GROUP2}]
         [itime=minutes]               (minutes:   1-9999)
         [isize=kb]                    (kilobytes: 0-4194300)
         [rtime=minminutes-maxminutes](minutes:   1-9999)
         [rsize={minkb-maxkb|0|0-0}]  (kilobytes: 0-4194300)

fwkeytran cmd=delete
          keytran=key_transform_id_or_name

fwkeytran cmd=list
         [keytran=key_transform_id_or_name]
         [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds a key transform.

name=*name*
the name of the key transform. Maximum length is 100 characters.

desc=*description*
the description of the key transform. Maximum length is 100 characters. This parameter is optional.

prot={IKE}
the name of the key transform protocol. **IKE** is the only supported protocol and is the default.

authmeth={SHAREDKEY|RSASIG}
specifies whether a pre-shared key or RSA signatures are used to authenticate key exchanges.

hashalg={MD5|SHA}
specifies the hash algorithm used to authenticate key exchanges.

encralg={DES_CBC_8|3DES_CBC}
specifies the encryption algorithm used to protect key exchanges.

dhgrp={GROUP1|GROUP2}
specifies the Diffie-Hellman group used when negotiating a security association. The default is **GROUP1**.

itime=*minutes*
the security association lifetime to propose when initiating a connection using this transform. Valid values are 1-9999. The default is 480 minutes.

isize=*kb*
the security association lifesize to propose when initiating a connection using this transform. Valid values are 0-4194300. A value of 0 indicates no size limit. The default value is 0.

rtime=*minminutes-maxminutes*
the security association lifetime range allowed when responding to a connection request using this transform. Valid values, for each value are 1-9999. The range must be specified with the smaller value first, or both values may be equal. The default is 60-1440 minutes.

rsize={*minkb-maxkb|0|0-0***}**
the security association lifesize range allowed when responding to a connection request using this transform. Valid values, for each value are 0-4194300. The range must be specified with the smaller value first, or both values may be equal. The default value is 0.

Note: **rsize=0-0** is equivalent to **rsize=0**, and it will be stored as 0, not as 0-0.

cmd=change
changes a key transform. The parameters are the same as for **add**, except that **keytran** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes a key transform.

keytran=*key_transform_id_or_name*
the ID or name of the key transform to delete.

cmd=list
lists one or all key transforms.

keytran=*key_transform_id_or_name*
the ID or name of the key transform to list. If **keytran** is not specified, all key transforms are listed. This parameter is optional.

format={short|long|wide|raw}
the format of the listing. If **keytran** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

Format	Description
short	80 columns formatted as: `ID name desc`

long	80 columns, formatted as: keyword = *value*, one parameter per line.
	`id = id`
	`name = name`
	`desc = description`
	`prot = protocol`
	`authmeth = authmethod`
	`hashalg = hashalg`
	`encralg = encralg`
	`dhgrp = dhgroup`
	`itime = minutes`
	`isize = kb`
	`rtime = minminutes-maxminutes`
	`rsize = minkb-maxkb`

wide	132 columns formatted as: `ID name desc`

raw	the values are separated by vertical bars, in the following order:
	`id|name|desc|prot|authmeth|hashalg|encralg|dhgrp|itime|isize|rtime|rsize`
	For example:
	`501|ktransf1|transform desc.|IKE|SHAREDKEY|MD5|DES_CBC_8|`
	`GROUP1|480|0|60-1440|9999-999999`

Usage

A key transform is a set of information used during key protection negotiation.

Use the **fwkeytran cmd=list** command to list the key transforms that are supplied with OS/390 Firewall Technologies. Carefully consider the listed key transforms, then define all other key transforms needed on this system.

After a key transform is defined, associate it with a key proposal using the **fwkeyprop** command.

Security association life values must be specified for both initiator and responder roles. The initiator role must have an expiration time. It may also have an expiration size. The responder role must have an expiration time range. It may also have an expiration size range. The initiator role values should be within the range specified for the responder.

Examples

1. Create a key transform that uses a pre-shared key to authenticate key exchanges. This key transform will use default values for the other parameters.

```
User:   fwkeytran cmd=add name="Key Transform A" desc="This is key transform A" \
        authmeth=SHAREDKEY hashalg=MD5 encralg=DES_CBC_8
System:
User:   fwkeytran cmd=list format=long
System:
                id = 501
              name = Key Transform A
              desc = This is key transform A
              prot = IKE
          authmeth = SHAREDKEY
           hashalg = MD5
           encralg = DES_CBC_8
             dhgrp = GROUP1
             itime = 480
             isize = 0
             rtime = 60-1440
             rsize = 0
```

2. Create a key transform that uses an RSA signature certificate to authenticate key exchanges. This key transform will have a lifetime of 480 minutes and an unlimited lifesize when used to initiate a negotiation and a lifetime between 480 and 720 minutes and a maximum lifesize of 1 million kilobytes when used to respond to a negotiation request.

```
User:   fwkeytran cmd=add name="Key Transform B" desc="This is key transform B" \
        authmeth=RSASIG hashalg=SHA encralg=3DES_CBC itime=480 isize=0 rtime=480-720 \
        rsize=1-1000000
System:
User:   fwkeytran cmd=list keytran="Key Transform B"
System:
                id = 502
              name = Key Transform B
              desc = This is key transform B
              prot = IKE
          authmeth = RSASIG
           hashalg = SHA
           encralg = 3DES_CBC
             dhgrp = GROUP1
             itime = 480
             isize = 0
             rtime = 480-720
             rsize = 1-1000000
```

3. Change a key transform so that there is no limit on the security association responder lifesize (rsize). Note that if rsize is specified as '0-0' in the command, it shows in the list as '0'.

```
User:   fwkeytran cmd=change keytran="Key Transform B" rsize=0-0
System:
User:   fwkeytran cmd=list keytran="Key Transform B"
System:
                id = 502
              name = Key Transform B
              desc = This is key transform B
              prot = IKE
          authmeth = RSASIG
           hashalg = SHA
           encralg = 3DES_CBC
             dhgrp = GROUP1
             itime = 480
             isize = 0
             rtime = 480-720
             rsize = 0
```

fwlogmgmt

Purpose

Activates log file archiving.

Note: This command can only be run from a **superuser**, not a **fwgrp** user.

Format

```
fwlogmgmt  {-a|-l}
```

Keywords and Parameters

-a purges the archives.

-l activates log file archiving.

Usage

The **fwarchive** command provides configuration definitions for record retention. It allows you to define the number of days a log record will remain in the log file before it is archived, the archive filename and the number of days before it is purged from the archive. The **fwlogmgmt** command maintains the archive files using the definitions defined through the **fwarchive** command.

Archiving Logs

To start a log management program to archive accumulated logs, you have two options:

1. run the **fwlogmgmt -l** command from the command line.
2. Set up the **fwlogmgmt -l** command in the `crontab`. See the *OS/390 UNIX System Services Command Reference*, SC28-1892, for information on the `crontab` function.

Log Management Outputs

The log management facility does some preliminary integrity checks before proceeding with any log management activities. If any problems are found, diagnostics are output to the OS/390 operator console when you run the **fwlogmgmt** command from the command line.

The **fwlogmgmt -a** command purges the archives.

Once records are archived, the records stay in the archive for a defined period of time before being purged. This time is defined through the **arctime** parameter on the **fwarchive** command.

To read or work with records that were placed in the archive, use the **ar** OS/390 UNIX command to extract the file from the archive. Then uncompress the file using the **uncompress** OS/390 UNIX command. For additional information on OS/390 UNIX commands, see the *OS/390 UNIX System Services Command Reference*, SC28-1892.

Examples

The most efficient or convenient means of running the log management process is to set it up as a `cron` job. This periodically runs the log archiving process at a predetermined frequency. Root must set up the `crontab` file to determine the frequency of running log management.

1. For example, if you want to set up the log management archiving process to run at 3:00 am every day of the year, type:

   crontab -e

   and add the following line:

   ```
   0 3 * * * fwlogmgmt -l
   ```

2. If you want to purge the archives at 3:00 am every day of the year, type `crontab -e` and add the following line:

   ```
   0 3 * * * fwlogmgmt -a
   ```

fwlogtxt

Purpose

fwlogtxt takes a standard **local4** log file produced by **syslogd** and outputs a readable file of message text.

Note: This command may only be issued by a **superuser**.

Format

```
fwlogtxt logfilename outputfilename
```

Keywords and Parameters

logfilename
is a standard system log file produced by **syslogd**.

outputfilename
is a readable text file of messages from the firewall daemons recording their activity (as monitored by **syslogd**).

Usage

It is recommended that you run **fwlogtxt** against log files containing `local4` facility messages.

Examples

1. Produce readable file `logtxt.out` from input system log file `fw961031.log`:

   User:

   ```
   fwlogtxt  fw961031.log logtxt.out
   ```

fwmigrate

Purpose

Handles generic migration and defines an initial TCP/IP firewall stack.

Format

```
fwmigrate fromrel={V2R6|V2R7|V2R8}
```

Keywords and Parameters

fromrel={V2R6|V2R7|V2R8}
the version/release of OS/390 Firewall Technologies which is currently in use.

V2R6
select this if you are running a V2R7 or earlier version. The **fwmigrate** command will migrate the system directly to V2R8.

V2R7
select this if you are running OS/390 Firewall Technologies V2R7. Selecting V2R7 when you are running OS/390 Firewall Technologies V2R6 or earlier will yield unpredictable results.

V2R8
select this if you are running OS/390 Firewall Technologies V2R8. You can use V2R8 when migrating from V2R9 as well, because there were no changes to OS/390 Firewall Technologies from Release 8 to Release 9. Selecting V2R8 when you are running OS/390 Firewall Technologies V2R6 or earlier will yield unpredictable results.

Usage

fwmigrate handles automatic migration for individual configuration files and defines an initial TCP/IP firewall stack. **fwmigrate** converts old configuration file formats to the new formats.

This command is designed to be executed multiple times, as errors are found, reported, and fixed, skipping completed steps.

Note: **fwmigrate** backup files (files that existed in the previous release) are stored in `/etc/security/fwbackup`. If restoring the previous release, these files can be copied into their original location.

If **fwmigrate** had previously been used to migrate to V2R7 or V2R8, the files in `/etc/security/fwbackup` relate to that migration. These files will be overwritten with the files as they currently exist.

Examples

1. Run a migration from Release 6 to Release 10.

```
User:     fwmigrate fromrel=V2R6
System:   ICAC1581i FIREWALL MIGRATION STARTED.

          ICAC1583i Initial firewall stack migration started.
          ICAC1584i Initial firewall stack migration completed.

          ICAC1600i Removing firewall .active status files started.
          ICAC1601i Removing firewall .active status files completed.

          ICAC1592i Filter migration started.
          ICAC1593i Filter migration completed.

          ICAC1585i Tunnel migration started.
          ICAC1586i Tunnel migration complete.

          ICAC1587i NAT migration started.
          ICAC1588i NAT migration completed.

          ICAC1589i DNS migration started.
          ICAC1591i DNS migration completed.

          ICAC1600i Removing firewall .active status files started.
          ICAC1601i Removing firewall .active status files completed.

          ICAC1719i Name migration started.
          ICAC1721i Name migration complete.

          ICAC1722i Objects migration started.
          ICAC1724i Objects migration complete.

          ICAC1741e Manual processing required for daemon configuration migration.
          ICAC1732i Predefined services migration started.
          ICAC1734i Predefined services migration completed.

          ICAC1735i Predefined rules migration started.
          ICAC1737i Predefined rules migration completed.

          ICAC1600i  Removing firewall .active status files started.
          ICAC1601i  Removing firewall .active status files completed.

          ICAC1770i  Connections migration started.
          ICAC1771i  Connections migration completed.

          ICAC1738i  Daemon configuration migration started.
          ICAC1740i  Daemon configuration migration completed.

          ICAC1782i  Syslog configuration migration started.
          ICAC1784i  Syslog configuration migration completed.

          ICAC1785i  Log file archive configuration migration started.
          ICAC1787i  Log file archive configuration migration completed.

          ICAC1732i  Predefined services migration started.
          ICAC1734i  Predefined services migration completed.

          ICAC1735i  Predefined rules migration started.
          ICAC1737i  Predefined rules migration completed.

          ICAC1582i FIREWALL MIGRATION COMPLETED SUCCESSFULLY.
```

2. Run a migration from Release 7 to Release 10.

```
User:     fwmigrate fromrel=V2R7
System:   ICAC1581i FIREWALL MIGRATION STARTED.

          ICAC1600i Removing firewall .active status files started.
          ICAC1601i Removing firewall .active status files completed.

          ICAC1719i Name migration started.
          ICAC1721i Name migration complete.

          ICAC1722i Objects migration started.
          ICAC1724i Objects migration complete.

          ICAC1738i Daemon configuration migration started.
          ICAC1740i Daemon configuration migration completed.

          ICAC1732i Predefined services migration started.
          ICAC1734i Predefined services migration completed.

          ICAC1735i Predefined rules migration started.
          ICAC1737i Predefined rules migration completed.

|         ICAC1600i  Removing firewall .active status files started.
|         ICAC1601i  Removing firewall .active status files completed.

|         ICAC1770i  Connections migration started.
|         ICAC1771i  Connections migration completed.

|         ICAC1738i  Daemon configuration migration started.
|         ICAC1740i  Daemon configuration migration completed.

|         ICAC1782i  Syslog configuration migration started.
|         ICAC1784i  Syslog configuration migration completed.

|         ICAC1785i  Log file archive configuration migration started.
|         ICAC1787i  Log file archive configuration migration completed.

|         ICAC1732i  Predefined services migration started.
|         ICAC1734i  Predefined services migration completed.

|         ICAC1735i  Predefined rules migration started.
|         ICAC1737i  Predefined rules migration completed.

          ICAC1582i FIREWALL MIGRATION COMPLETED SUCCESSFULLY.
```

| 3. Run a migration from Release 8 to Release 10.

```
      User:  fwmigrate fromrel=V2R8
    System:  ICAC1581i FIREWALL MIGRATION STARTED.

             ICAC1600i  Removing firewall .active status files started.
             ICAC1601i  Removing firewall .active status files completed.

             ICAC1770i  Connections migration started.
             ICAC1771i  Connections migration completed.

             ICAC1738i  Daemon configuration migration started.
             ICAC1740i  Daemon configuration migration completed.

             ICAC1782i  Syslog configuration migration started.
             ICAC1784i  Syslog configuration migration completed.

             ICAC1785i  Log file archive configuration migration started.
             ICAC1787i  Log file archive configuration migration completed.

             ICAC1732i  Predefined services migration started.
             ICAC1734i  Predefined services migration completed.

             ICAC1735i  Predefined rules migration started.
             ICAC1737i  Predefined rules migration completed.

             ICAC1582i FIREWALL MIGRATION COMPLETED SUCCESSFULLY.
```

fwnat

Purpose

Configures network address translation.

Format

```
fwnat cmd=add
      type=reserve
      addr=x.x.x.x
      mask=x.x.x.x
     [timeout=minutes]          (minutes=15-240)
     [stack=stackname]

      defaults: timeout=15

fwnat cmd=add
      type=map
      secaddr=x.x.x.x
      remaddr=x.x.x.x
     [stack=stackname]

      defaults: timeout=15

fwnat cmd=add
      type={translate|exclude}
      addr=x.x.x.x
      mask=x.x.x.x
     [stack=stackname]

fwnat cmd=change
      index=number
     [addr=x.x.x.x]
     [mask=x.x.x.x]
     [secaddr=x.x.x.x]
     [remaddr=x.x.x.x]
     [timeout=[minutes]]          (minutes=15-240)
     [force={yes|no}]
     [stack=stackname]

fwnat cmd=delete
      index=number
     [force={yes|no}]
     [stack=stackname]

fwnat cmd={list|update|verify|shutdown|startlog|stoplog|listlog}
     [stack=stackname]
```

Keywords and Parameters

cmd=add
adds an entry to the NAT configuration. Depending on the type of entry added, different parameters must be specified.

type=reserve
adds the addresses specified to the NAT registered address pool.

addr=*x.x.x.x*
IP address in dotted-decimal form that identifies a range of registered IP addresses to be added to the registered address pool.

mask=*x.x.x.x*
a mask that specifies the bits in the registered IP address used to add a range of IP addresses to the registered address pool. Bits in these masks that are set to 0 indicate that bit positions that have 0 or 1 are reserved registered IP addresses. So, specifying 255.255.255.255 in the mask indicates that only one registered address is added to the registered address pool, whereas a mask of 255.255.0.0 causes 65536 IP addresses to be added to the registered address pool.

timeout=[*minutes***]**
the number of minutes an address translation can remain idle before NAT can free the registered IP address. This timeout value only applies to the address translation that uses a registered IP address within the range of IP addresses specified by this entry.

The default is 15 minutes and is the minimum value allowed. The maximum value allowed is 240 minutes. Both this parameter and its value are optional.

stack=[*stackname***]**
the stack name for the TCP/IP stack created using the **fwstack** command.

type=map
defines a one-to-one secured-to-registered IP address static translation.

secaddr=*x.x.x.x*
IP address in dotted-decimal format that should be translated into a specified registered IP address.

remaddr=*x.x.x.x*
IP address in dotted-decimal format into which a specified secure address should be translated.

stack=[*stackname***]**
the stack name for the TCP/IP stack created using the **fwstack** command.

type=translate
specifies a range of secure IP addresses that require network address translation.

addr=*x.x.x.x*
IP address in dotted-decimal form that identifies a range of secure IP addresses that should be translated.

mask=*x.x.x.x*
a mask (like a subnet mask) that specifies the bits in the secure IP address used to specify a range of IP addresses. Bits in these masks that are set to 0 indicate that bit positions that have 0 or 1 are included in the range of addresses. So, specifying 255.255.255.255 in the mask indicates that only one secure address is included in this translation entry.

stack=[*stackname***]**
the stack name for the TCP/IP stack created using the **fwstack** command.

type=exclude
specifies a range of secure IP addresses that should be excluded from network address translation.

addr=*x.x.x.x*
IP address in dotted-decimal form that identifies a range of secure IP addresses that should be excluded from translation.

mask=*x.x.x.x*
a mask (like a subnet mask) that specifies the bits in the secure IP address used to specify a range of IP addresses. Bits in these masks that are set to 0 indicate that bit positions that have 0 or 1 are included in the range of addresses. So, specifying 255.255.255.255 in the mask indicates that only one secure address is specified in this entry, whereas a mask of 255.255.255.0 indicates 256 IP addresses are excluded from address translation.

stack=[*stackname***]**
the stack name for the TCP/IP stack created using the **fwstack** command.

cmd=change
changes a NAT entry in the configuration file.

index=*number*
is the index number of the entry to be changed.

Specify the remaining parameters **addr**, **mask**, **secaddr**, **remaddr**, **timeout**, and **force** (all optional) according to how you want to change the entry. **timeout** can be set to a blank value to reset it to the default value of 15 minutes.

stack=[*stackname***]**
the stack name for the TCP/IP stack created using the **fwstack** command.

cmd=delete
deletes an entry specified by index number from the NAT configuration file.

index=*number*
index number of the entry to be deleted.

force={yes|no}
specifies whether the delete is forced and the confirmation message suppressed. If "yes" is specified, the entry is deleted with no confirmation. This parameter is optional.

cmd=list
lists the entries in the NAT configuration file by index number.

cmd=update
enables network address translation.

cmd=verify
validates the NAT entries in the NAT configuration file.

cmd=shutdown
disables network address translation.

cmd=startlog
enables the logging produced by **cmd=listlog**. Logging takes place at the initial mapping phase and is not dynamic. It does not show each actual mapping in the kernel upon request. You must enable logging before activation. If not, you could lose log entries. If you refresh the log or erase it, you will also lose log entries.

cmd=stoplog
stops NAT logging controlled by **cmd=startlog**.

Note: This is not **local4** logging.

cmd=listlog
lists the contents of the last 1024 NAT logs. The format is as follows, one line per log entry:

```
Time(%1$s %2$s)  Secured: %3$s  Registered: %4$s  Static(%5$d)
```

where:

%1$s date of log entry

%2$s time of log entry

%3$s secure address

%4$s registered address

%5$d 0 or 1 value specifies whether the entry is a static mapping. 1 means the entry is static. (MAP rules are static; TRANSLATE/RESERVE rules are dynamic. EXCLUDE rules don't have log entries.)

Usage

fwnat manages the NAT configuration, which controls the translation of IP addresses in a secure address space to IP addresses in a nonsecure address space.

For more information about NAT, see Chapter 15, “Network Address Translation” on page 103.

Note: NAT configuration must not conflict between firewall stacks. NAT requires a stack parameter for multiple stacks. An error message appears if the NAT configuration conflicts between stacks.

Examples

1. Add some NAT information

```
   User:  fwnat cmd=add type=reserve addr=1.2.3.4 mask=255.255.255.255 stack=TCPIPAA
System:
   User:  fwnat cmd=add type=reserve addr=5.6.7.8 mask=255.255.255.255 stack=TCPIPAA
System:
   User:  fwnat cmd=add type=map secaddr=99.100.50.250 remaddr=99.100.5.5 stack=TCPIPAA
System:
   User:  fwnat cmd=add type=translate addr=10.20.30.40 mask=255.255.255.0 stack=TCPIPAA
System:
   User:  fwnat cmd=add type=exclude addr=4.30.200.0 mask=255.255.0.0 stack=TCPIPAA
System:
```

2. Show the NAT configuration

```
   User:  fwnat cmd=list stack=TCPIPAA
System:   1. RESERVE  1.2.3.4  255.255.255.255  15
          2. RESERVE  5.6.7.8  255.255.255.255  15
          3. MAP  99.100.50.250  99.100.5.5
          4. TRANSLATE  10.20.30.40  255.255.255.0
          5. EXCLUDE  4.30.200.0  255.255.0.0
```

3. Change a NAT entry

```
   User:    fwnat cmd=change index=1 addr=4.3.2.1 force=yes stack=TCPIPAA
System:
   User:    fwnat cmd=list stack=TCPIPAA
System:     1. RESERVE  4.3.2.1  255.255.255.255  15
            2. RESERVE  5.6.7.8  255.255.255.255  15
            3. MAP  99.100.50.250  99.100.5.5
            4. TRANSLATE  10.20.30.40  255.255.255.0
            5. EXCLUDE  4.30.200.0  255.255.0.0
```

4. Delete a NAT entry

```
   User:    fwnat cmd=delete index=1 force=yes stack=TCPIPAA
System:
   User:    fwnat cmd=list stack=TCPIPAA
System:     1. RESERVE  5.6.7.8  255.255.255.255  15
            2. MAP  99.100.50.250  99.100.5.5
            3. TRANSLATE  10.20.30.40  255.255.255.0
            4. EXCLUDE  4.30.200.0  255.255.0.0
```

fwnwgrp

Purpose

Configures group objects.

Format

```
fwnwgrp cmd=create
        name=name
       [desc=description]
       [idlist=id1,id2,...,idn]
       [namelist="name1|name2|...|name n"]

fwnwgrp cmd=change
       {name=name | id=id}
       [desc=[description]]
       [idlist=[id1,id2,...,idn]]
       [namelist=["name1|name2|...|name n"]]

fwnwgrp cmd=delete
       {name=name | id=id}
       [force={yes|no}]

fwnwgrp cmd=list
       {name=name | id=id}
       [format={short|long|wide|raw|all}]

fwnwgrp cmd=add
       {name=name | id=id}
       {idlist=id1,id2,...,idn and/or
        namelist="name1|name2|...|name n"}

fwnwgrp cmd=remove
       {name=name | id=id}
       {idlist=id1,id2,...,idn and/or
        namelist="name1|name2|...|name n"}
```

Keywords and Parameters

cmd=create
creates a network object group.

name=*name*
is the name of the group object to create. Maximum length is 100 characters.

desc=*description*
is the description of the group object. Maximum length is 100 characters. This parameter is optional.

idlist=*id1,id2, ...,idn*
is list of IDs of single objects that will be in the new group object. This parameter is optional.

namelist=*"name 1*|*name2*| **...**|*name n"*
is a list of names of single objects names separated by vertical bars, that will be in the new group object. This parameter is optional.

cmd=change
changes an existing network object group.

{name=*name* | **id=***id***}**
the name or ID of the group object to be changed.

desc=*description*
is the description of the new object. This parameter is optional, and may be specified as blank to delete an existing description.

idlist=*id1,id2, ...,idn*
is a list of IDs of single objects that will be in the new group object. This parameter is optional.

namelist=*"name1*|*name2*| **...**|*name n"*
is a list of single object names, separated by vertical bars, that will be in the new group object. This parameter is optional and may be specified as blank to delete existing values.

You can also use **cmd=change** with blank values for **idlist=** or **namelist=** to remove all objects from the group, thereby changing it to an empty group.

cmd=delete
deletes a network object group.

{name=*name* | **id=***id***}**
is the name or ID of the group object to be deleted.

force={yes|no}
forces the delete even if dependent connections are found associated with the group you are deleting. This parameter is optional.

cmd=list
lists a specific network group by name or ID.

{name=*name* | **id=***id***}**
is the group object, by name or ID, to be listed.

format={short|long|wide|raw|all}
is the format of the list output. This parameter is optional. The listing can be in any of the following formats:

Format	Example
short	80,columns, formatted as: `ID name desc`
long	is formatted as: keyword = *value*, one parameter per line.
wide	132 columns formatted as:
raw	the values are separated by vertical bars.
all	like long, but each member is also shown similar to format=short with 18 characters truncated from the description and aligned to fit an 80-column screen. (See 287).

```
  id = id
name = name
desc = desc

⋮
```

The default if no name or ID is specified is short; otherwise, it is long.

cmd=add
adds objects to an existing network object group. If the object or name is already present in the group, it does not get added again.

{name=*name* **| id=***id***}**
is the name or ID of the object group to which objects are to be added.

{idlist=*id1,id2, ...,idn* **and/or**
namelist=*"name 1***|***name2***| ...|***name n"***}**
is a list of names (separated by vertical bars) and/or IDs (separated by commas) of single objects that will be in the new group object.

cmd=remove
removes a object from a network object group.

{name=*name* **| id=***id***}**
name of the object to be removed.

{idlist=*id1,id2, ...,idn* **and/or**
namelist=*"name 1***|***name2***| ...|***name n"***}**
is a list of names (separated by vertical bars) and/or IDs (separated by commas) of single objects that will be removed from the group object.

Usage

A **network object group** is itself a network object that represents a collection of single network objects. They are used as a convenience in setting up connections and can eliminate repetitive work; for example, you could group several addresses together into a network object group to represent a department. This network object group can be used as either the source or destination for a connection.

Before using the **fwnwgrp** command, you must use the **fwnwobj** command to define the objects you want to group together to reference as one entity.

For more information on network objects and how they are used in filtering, see Chapter 8, “Controlling Traffic Through the Firewall with Filters” on page 51.

Note: Objects that point to a network group cannot point to an empty network group.

Examples

1. Create an empty network object group

```
   User:  fwnwgrp cmd=create name="Group A" desc="This is group A"
System:
   User:  fwnwgrp cmd=list format=long
System:
                    id = 506
                  name = Group A
                  desc = This is group A
                idlist =
```

2. Add two objects to the network object group

```
   User:     fwnwgrp cmd=add name="Group A" namelist="Network Object 1|Network Object 5"
System:
   User:     fwnwgrp cmd=list format=long
System:

                     id = 506
                   name = Group A
                   desc = This is group A
                 idlist = 501,505
```

3. Create another network object group which contains three objects

```
   User:     fwnwgrp cmd=create name="Group B" idlist=502,503,504
System:
   User:     fwnwgrp cmd=list
System:    506   Group A          This is group A
           507   Group B
```

4. Create a network object group which contains four objects: two specified by id and two specified by name

```
   User:     fwnwgrp cmd=create name="Group C" idlist=502,504 \
             namelist="Network Object 5|Network Object 1"
System:
   User:     fwnwgrp cmd=list name="Group C" format=all
System:

                     id = 508
                   name = Group C
                   desc =
                 idlist = 505   Host         Network Object 5 Network object host 5
                          501   Host         Network Object 1 Network object host 1
                          502   Host         Network Object 2 Network object host 2
                          504   Host         Network Object 4 Network object host 4
```

5. Remove three objects from a network object group

```
   User:     fwnwgrp cmd=remove id=508 idlist=504,502 namelist="Network Object 1"
System:
   User:     fwnwgrp cmd=list id=508 format=all
System:

                     id = 508
                   name = Group C
                   desc =
                 idlist = 505   Host         Network Object 5 Network object host 5
```

6. Delete a network object group. Select the force option, so that not only will the group itself be deleted, but all instances of the group will be removed from the connection definitions.

```
   User:     fwnwgrp cmd=delete name="Group C" force=yes
System:
   User:     fwnwgrp cmd=list
System:    506   Group A          This is group A
           507   Group B
```

fwnwobj

Purpose

Configures single network objects.

Format

```
fwnwobj cmd=add
        name=name
        [desc=description]
        type={Host|Network|Firewall|
             Router|Interface|VPN}
        {addr=x.x.x.x      mask=x.x.x.x or
        startaddr=x.x.x.x endaddr=x.x.x.x}

fwnwobj cmd=change
        {name=name | id=id}
        [desc=[description]]
        [type={Host|Network|Firewall|
               Router|Interface|VPN}]
        [addr=[x.x.x.x]]
        [mask=[x.x.x.x]]
        [startaddr=[x.x.x.x]]
        [endaddr=[x.x.x.x]]

fwnwobj cmd=delete
        {name=name | id=id}
        [force={yes|no}]

fwnwobj cmd=list
        [name=name] | [id=id]
        [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds an object.

name=*name*
is the name of the object. Maximum length is 100 characters.

desc=*description*
is the text description of the object. Maximum length is 100 characters. This parameter is optional.

type={Host|Network|Firewall|Router|Interface|VPN}
is the type of object. Valid types of network objects are:

Host a particular node on your network. It generally will have a mask of 255.255.255.255.

Network a collective range of network addresses that is characterized by an address and subnet mask.

Firewall a single machine with a Firewall installed on it.

Router a host which routes traffic between two or more networks.

Interface a network adapter on a machine. It does not have to be an adapter on the Firewall.

VPN Virtual Private Network (VPN) is the network on the other side of a tunnel.

addr=*x.x.x.x*
is the dotted-decimal IP address of the object.

mask=*x.x.x.x*
is the subnet mask that specifies the bits in the address to compare to the address of the IP packet.

startaddr=*x.x.x.x.*
the starting address of an IP address range. **startaddr** and **endaddr** are used together and are mutually exclusive with **addr** and **mask**.

endaddr=*x.x.x.x.*
the ending address of an IP address range. **startaddr** and **endaddr** are used together and are mutually exclusive with **addr** and **mask**.

cmd=change
changes the definition of an object. The parameters are the same as for **add** above, except that the only required parameter is **name** or **id**. In addition, **desc=** may be specified as blank to delete an existing description.

cmd=delete
deletes an object by name or ID.

{name=*name* **| id=***id***}**
is the name or ID of the object.

force={yes|no}
force=yes forces the delete even if dependent objects are found associated with the object you are deleting.

cmd=list
lists a network object.

name=*name* **| id=***id*
is the name or ID of the object. This parameter is optional. If no value is specified, all single objects will be listed.

format={short|long|wide|raw}
is the format of the list. **format** defaults to **short** if **name** or **id** is not specified, and **long** if **name** or **id** is specified. Following are the formats available:

Format	Description
short	80 columns, formatted as: `ID type name desc`
long	is formatted as: keyword = *value*, one parameter per line.
wide	132 columns formatted as:
raw	the values are separated by vertical bars.

```
  id = id
type = type
name = name

⋮
```

The default if no name or ID is specified is **short**; otherwise, it is **long**.

Usage

Network objects are representations of entities on your network such as hosts, networks, routers, users, or virtual private networks. You use network objects in IP filtering to designate source and destination addresses of services when you create your connections. (See Chapter 8, “Controlling Traffic Through the Firewall with Filters” on page 51 for more information about how filtering works.)

- Objects are used to create a connection.
- Objects and groups are tied together.
- A group is a collection of objects.

You must use the **fwnwobj** command to define all hosts, networks, firewalls, routes, and interfaces, to create a virtual private networks.

Examples

1. Show the predefined network object, in long format.

```
   User:  fwnwobj cmd=list format=long
System:
                     id = 1
                   type = Network
                   name = The World
                   desc =
                   addr = 0.0.0.0
                   mask = 0.0.0.0
              startaddr =
                endaddr =
```

2. Add some network objects.

```
   User:  fwnwobj cmd=add name="Department ABC Host" \
          desc="Host ip address for Department ABC" type=host addr=10.20.30.40 \
          mask=255.255.255.255
System:
   User:  fwnwobj cmd=add name="Firewall Machine 1" desc="This is firewall machine 1" \
          type=firewall addr=11.12.13.14 mask=255.255.0.0
System:
   User:  fwnwobj cmd=add name="Virtual Private Network 1" type=VPN addr=11.22.33.44 \
          mask=255.255.255.255
System:
   User:  fwnwobj cmd=add name="IP Range 1" desc="Range of IP addresses" type=Network \
          startaddr=1.2.3.1 endaddr=1.2.5.10
System:
```

3. Show the network objects, in short format.

```
   User:  fwnwobj cmd=list
System:   501   Host         Department ABC H Host ip address for Department ABC
          502   Firewall     Firewall Machine This is firewall machine 1
          504   Network      IP Range 1       Range of IP addresses
          1     Network      The World
          503   VPN          Virtual Private
```

4. Show the network objects, in raw format.

```
   User:  fwnwobj cmd=list format=raw
System:   501|Host|Department ABC Host|Host ip address for Department ABC|10.20.30.40|255.255.255.255||
          502|Firewall|Firewall Machine 1|This is firewall machine 1|11.12.13.14|255.255.0.0||
          504|Network|IP Range 1|Range of IP addresses|||1.2.3.1|1.2.5.10
          1|Network|The World||0.0.0.0|0.0.0.0||
          503|VPN|Virtual Private Network 1||11.22.33.44|255.255.255.255||
```

5. Change a network object.

```
   User:     fwnwobj cmd=change name="Department ABC Host" mask=255.255.240.240
System:
   User:     fwnwobj cmd=list id=501
System:
                     id = 501
                   type = Host
                   name = Department ABC Host
                   desc = Host ip address for Department ABC
                   addr = 10.20.30.40
                   mask = 255.255.240.240
              startaddr =
                endaddr =
```

6. Delete a network object.

```
   User:     fwnwobj cmd=delete name="Virtual Private Network 1"
System:
```

fwondemand

Purpose

Configures an on-demand object to define the dynamic connection granularity and key server information.

Format

```
fwondemand cmd=add
           name=name
          [desc=description]
          [srcipgran={packet|anchor}]
          [destipgran={packet|anchor}]
          [gatekeysrv=key_server_id_or_name]
           keysrvgrp=key_server_group_id_or_name

fwondemand cmd=change
           ondemand=ondemand_id_or_name
          [desc=description]
          [srcipgran=[{packet|anchor}]]
          [destipgran=[{packet|anchor}]]
          [gatekeysrv=[key_server_id_or_name]]
          [keysrvgrp=key_server_group_id_or_name]

fwondemand cmd=delete
           ondemand=ondemand_id_or_name

fwondemand cmd=list
          [ondemand=ondemand_id_or_name]
          [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
creates an on-demand object

name=*name*
the name of the on-demand object. Maximum length is 100 characters.

desc=*description*
the description of the on-demand object. Maximum length is 100 characters. This parameter is optional.

srcipgran={packet|anchor}
the granularity that will be applied to the source IP address in the dynamic connection being established. The default source granularity is **packet**.

anchor
the source address in the matching anchor rule will be used as the source IP address in the dynamic connection being negotiated. If a mask or range is specified on the anchor rule, anchor granularity will cause one dynamic connection to be created for all addresses included in the mask/range.

packet
the source address in the outbound IP packet will be used as the source IP address in the dynamic connection being negotiated. Packet granularity will cause one dynamic connection to be created for each unique IP address.

destipgran={packet|anchor}
the granularity that will be applied to the destination IP address in the dynamic connection being established. The default destination granularity is **packet**.

anchor
the destination address in the matching anchor rule will be used as the destination IP address in the dynamic connection being negotiated. If a mask or range is specified on the anchor rule, anchor granularity will cause one dynamic connection to be created for all addresses included in the mask/range.

packet
the destination address in the outbound IP packet will be used as the destination IP address in the dynamic connection being negotiated. Packet granularity will cause one dynamic connection to be created for each unique IP address.

gatekeysrv=*key_server_id_or_name*
specifies the ID or name of the key server that is remote to the system being configured. This is the encryption/decryption point, or tunnel endpoint. **gatekeysrv** should be specified when the destination connection IP address is not the tunnel endpoint, in other words, when the tunnel endpoint is a gateway. The specified key server must be a member of the key server group specified in the parameter **keysrvgrp**.

If this parameter is not specified, the destination IP address specification in the dynamic connection being established will be used to locate the remote tunnel endpoint's key server entry. The destination IP address specification of the dynamic connection will be determined by the **destipgran** parameter.

If the destination IP address specification is a single IP address, that IP address will be used to locate the remote tunnel endpoint's key server entry. If the destination IP address specification is an address/mask specification, only the address portion of address/mask specification will be used to locate the remote tunnel endpoint's key server entry. If the destination IP address specification is a range specification, only the low order address of the range specification will be used to locate the remote tunnel endpoint's key server entry.

When locating the remote endpoint's key server entry, the first key server entry to meet on of the following conditions will be used:

- The key server entry's *ipaddr* matches the IP address being used to locate the key server entry.
- The key server entry's *hostname* resolves to the IP address being used to locate the key server entry.

The matching key server must be a member of the key server group specified in the parameter **keysrvgrp**.

If no matching key server entry is found, or if the matching key server entry is not a member of the key server group specified in the parameter **keysrvgrp**, the on-demand activation will fail.

keysrvgrp=*key_server_group_id_or_name*
the ID or name of the key server group used for the connection.

cmd=change
changes an on-demand object. The parameters are the same as **add**, except that **ondemand** and at least one parameter must be specified. Note that optional parameters may be specified as blank to delete an existing value.

cmd=delete
deletes an on-demand object. An on-demand object that is referenced by a connection cannot be deleted until either the connection is deleted or the on-demand object is removed from the connection.

ondemand=*ondemand_id_or_name*
the ID or name of the ondemand object to delete.

cmd=list
lists one or all on-demand objects.

format={short|long|wide|raw}
the format of the listing. If **ondemand** is specified, the default format is **long**, otherwise the default format is **short**. The formats available are:

short
80 columns formatted as: `ID name desc`.

long
formatted as keyword=value, one parameter per line.

```
        id=id
      name=name
      desc=description
 srcipgran=sourceIPgranularity
destipgran=destinationIPgranularity
gatekeysrv=remotekeyserverid
 keysrvgrp=keyservergroupid
```

wide
132 columns formatted as:

```
ID name desc
```

raw
the values are separated by vertical bars, in the following order:

```
id|name|desc|srcipgran|destipgran|gatekeysrv|keysrvgrp
```

For example,

```
501|OnDemand_object1|Ondemand Object|packet|anchor|502|503
```

Usage

The on-demand object describes the attributes applied to dynamic on-demand connection. These connections are not established prior to being used but are established on an as-needed basis. The on-demand object describes what level of granularity to apply to the connection being established.

Examples

1. Create an on-demand object using source and destination granularity of packet, no gateway key server and remote key server group Test1 (id = 503).

```
User:   fwondemand cmd=add name="On-demand Example 1" desc="On-demand - packet granularity"
        scripgran=packet destipgran=packet keysrvgrp=Test1
System:
User:   fwondemand cmd=list ondemand="On-demand Example 1"
System:
                 id = 504
               name = On-demand Example 1
               desc = On-demand - packet granularity
          srcipgran = packet
         destipgran = packet
         gatekeysrv =
          keysrvgrp = 503
```

2. Change the on-demand object to use destination granularity of anchor and gateway key server GateKey1 (id=502).

```
User:   fwondemand cmd=change name="On-demand Example1" destipgran=anchor gatekeysrv=GateKey1
        desc="On-demand with source packet granularity and destination anchor granularity"
System:
          id = 504
        name = On-demand Example 1
        desc = On-demand with source packet granularity and destination anchor
   srcipgran = packet
  destipgran = anchor
  gatekeysrv = 502
   keysrvgrp = 503
```

3. List the on-demand object in short format

```
User:   fwondemand cmd=list format=short
System:
                   504    On-demand Example 1  on-demand with source packet granularity and d
```

4. Delete the on-demand object.

```
User:   fwondemand cmd=delete ondemand="On-demand Example 1"
System:
```

fwsecpolicy

Purpose

Specifies the security policy for the firewall.

Format

```
fwsecpolicy cmd=change
            [servicelist=[id1,id2,...,idn]]

           where:  Security Policy for this firewall
            DNS
             id=20 Permit DNS queries
             id=21 Permit DNS zone transfers
            Logging
             id=23 Deny broadcast message to non-secure interface
            Socks
             id=34 Deny Socks to non-secure interface
            Temporary Only
             id=26 Shutdown service interface (panic)
             id=25 Test IP Routing (debug only)

fwsecpolicy cmd=list
```

Keywords and Parameters

cmd=change
changes the security policy.

servicelist=*id1, id2, ..., idn*
is the list of predefined service IDs that make up the security policy for the firewall. *id* variables represent two-digit numbers. Any number of the IDs may be specified. The valid choices are:

- **DNS**

  id=20 Permit DNS queries

  id=21 Permit DNS zone transfers

- **Logging**

  id=23 Deny broadcast message to non-secure interface

- **Socks**

  id=34 Deny socks to non-secure interface

- **Temporary only**

  id=25 Test IP Routing (debug only)

  id=26 Shutdown service interface (panic)

  All traffic over any secure interface is denied.

  Note: 25 and 26 are mutually exclusive.

If you don't specify the **servicelist=** parameter, you will be prompted to answer yes or no for each of the possible service IDs.

cmd=list
lists the service IDs of the security policy in effect.

Usage

The **security policy** of the firewall is the service or services (for example, filter rule sets) that are in effect for the entire installation. The services that may be specified for this purpose are predefined.

Examples

1. Show the current security policy rules

```
   User:  fwsecpolicy cmd=list
System:   Security Policy for this firewall
          DNS
           id=20 No  Permit DNS queries
           id=21 No  Permit DNS zone transfers
          Logging
           id=23 No  Deny broadcast message to non-secure interface
          Socks
           id=34 No  Deny Socks to non-secure interface
          Temporary only
           id=26 No  Shutdown service interface (panic)
           id=25 No  Test IP Routing (debug only)
```

2. Change the security policy rules

```
   User:  fwsecpolicy cmd=change servicelist=20,21,34
System:
   User:  fwsecpolicy cmd=list
System:   Security Policy for this firewall
          DNS
           id=20 Yes Permit DNS queries
           id=21 Yes Permit DNS zone transfers
          Logging
           id=23 No  Deny broadcast message to non-secure interface
          Socks
           id=34 Yes Deny Socks to non-secure interface
          Temporary only
           id=26 No  Shutdown service interface (panic)
           id=25 No  Test IP Routing (debug only)
```

3. Restore the defaults (no security policy rules)

```
   User:  fwsecpolicy cmd=change servicelist=
System:
   User:  fwsecpolicy cmd=list
System:   Security Policy for this firewall
          DNS
           id=20 No  Permit DNS queries
           id=21 No  Permit DNS zone transfers
          Logging
           id=23 No  Deny broadcast message to non-secure interface
          Socks
           id=34 No  Deny Socks to non-secure interface
          Temporary only
           id=26 No  Shutdown service interface (panic)
           id=25 No  Test IP Routing (debug only)
```

fwservice

Purpose

Configures IP filter services. A *service* is a set of filter rules which instruct the firewall to permit or deny a particular type of communication (for example, FTP).

Format

```
fwservice cmd=create
          name=name
         [desc=description]
         [rulelist=id1/{f|b},id2/{f|b},...,idn/&lbr c.f|b}]
         [log={yes|no}]
         [tunnel=tunnel_id]
         [time=hhmm-hhmm]       (hh=00-23, mm= 00-59)
         [month=mmm[-mmm]]      (mmm= Jan,Feb,...,Dec)
         [day=[nn][-nn]]        (nn= 1,2,...,29 or 30 or 31)
         [weekday=ddd[-ddd]]    (ddd= Sun,Mon,...,Sat)
         [timefilter={activate|deactivate}]

fwservice cmd=change
         {name=name | id=id}
         [desc=[description]]
         [rulelist=[id1/{f|b},id2/{f|b},...,idn/{f|b}]]
         [log=[{yes|no}]]
         [tunnel=[tunnel_id]]
         [time=hhmm-hhmm]       (hh= 00-23, mm= 00-59)
         [month=mmm[-mmm]]      (mmm= Jan,Feb,...,Dec)
         [day=[nn][-nn]]        (nn= 1,2,...,29 or 30 or 31)
         [weekday=ddd[-ddd]]    (ddd= Sun,Mon,...,Sat)
         [timefilter=[{activate|deactivate}]]

fwservice cmd=delete
         {name=name | id=id}
         [force={yes|no}]

fwservice cmd=list
         [name=name | id=id]
         [format={short|long|wide|raw}]

fwservice cmd=add
          rulelist=id1/{f|b},id2/{f|b},...,idn/{f|b}
          service=service_id_or_name
```

```
fwservice cmd=remove
          rulelist=id1/{f|b},id2/{f|b},...,idn/{f|b}
          service=service_id_or_name

fwservice cmd=move
          rule=id/{f|b}
          after=[id/{f|b}]
          service=service_id_or_name
```

Keywords and Parameters

cmd=create
creates a single service.

name=*name*
is the name of the service. Maximum length is 100 characters.

desc=*description*
is a text description of this service. Maximum length is 100 characters. This parameter is optional.

rulelist=*id1***/{f|b},***id2***/{f|b},...,***idn***/{f|b}**
is the comma-delimited list of rules, specified by ID number and direction, to add to the service you are creating. This parameter is optional; you can define a service without any rules and use **fwservice cmd=add** to add them instead. **rulelist=** works as follows:

id is the ID of a filter rule to add The ID specified can be of any type (permit|deny|anchor).

f|b is the direction (**f**orward or **b**ack) of the rule relative to the source and destination it acts upon. **F**orward flow means that the filter rule source is the packet's source and the filter rule destination is the packet's destination. **B**ackward flow means that the filter rule destination is the packet's source and the filter rule source is the packet's destination. (May be specified in upper or lower case.)

Example:

```
rulelist=501/f,502/b,510/f,508/f,512/b
```

The **log** and **tunnel** parameters that follow are overrides for the filter rules in the rule list.

log={yes|no}
specifies whether to log this service. There is no default value.

Log records written for matched filter rules contain the actual values from the IP packet, and can therefore provide valuable information about the content IP packets (for example, protocol and port numbers) seen by the firewall.

This parameter is optional.

tunnel=*tunnel_id*
specifies the ID of the tunnel over which this service should communicate. This parameter is used to specify manual tunnels only and applies to the command **fwfrules type=permit|deny**. This parameter is optional.

time=*hhmm-hhmm* **(***hh***=00-23,** *mm***= 00-59)**
is the timespan in the day the service is activated/deactivated, specified in 24-hour format. This parameter is optional. If no daily timespan is specified, the service is active all day.

Note: You cannot speicfy a time that spans 2 days (wraps concurrent days). To do this, you must set one to 23:59 and start another at 00:00.

month=*mmm***[***-mmm***]** **(***mm***= Jan, Feb, ..., Dec)**
is the month or range of months the service is activated/deactivated. This parameter is optional, and cannot be specified with **weekday=**. If no value is specified, the service is activated/deactivated all year.

day=[*nn***] [***-nn***]** **(***nn***= 1,2,...,29 or 30 or 31)**
is the day or span of days in the month the service is activated/deactivated. This parameter is only applicable with **month=**. It is optional, but will default if not specified. Use **cmd=list** to see the default values, as they will depend on how **month** is specified. The **month** and **day** parameters must be specified together. The simplest specification you can make is a single month and a single day, *mmdd*. If you specify either a **month** or a **day** range (or both), the combined specification must then be thought of as two related ranges:

```
month=start_month[–end_month]
day=start_day[–end_day]
```

where *start_month* is mapped to *start_day* and where *end_month* is mapped to *end_day* such that the two ranges resolve to a single range of the form *mmdd-mmdd*, where *mm* starts at 0. For example:

FWSERVICE Command Input	Results in these Two Ranges	Which Resolve to the Single Range
month=Jan	month=Jan-Jan day=1-31	0001-0031
month=Feb-Aug	month=Feb-Aug day=1-31	0101-0731
month=Aug-Feb	month=Aug-Feb day=1-29	0701-0129
month=Mar-May day=-15	month=Mar-May day=1-15	0201-0415
month=Dec-Feb day=31	month=Dec-Feb day=31-29	1131-0129
month=Dec-Feb day=-15	month=Dec-Feb day=1-15	1101-0115
month=Sep-Nov day=2	month=Sep-Nov day=2-30	0802-1030
month=Sep-Nov day=-2	month=Sep-Nov day=1-2	0801-1002
month=Sep day=3-15	month=Sep-Sep day=3-15	0803-0815
month=Feb day=-20	month=Feb-Feb day=1-20	0101-0120

weekday=*ddd***[***-ddd***]** **(***ddd***= Sun, Mon, ..., Sat)**
is the day or span of days in each week the service is activated/deactivated. This parameter is optional, and cannot be specified with **month=**.

timefilter={activate|deactivate}
specifies whether to activate or deactivate the service during the time periods specified with the other time-related parameters.

Filter rules with `type=anchor` do not support time filtering. It can be entered, but is not reflected on anchor rules.

This parameter is optional.

cmd=change
changes a service. Parameters are the same as for **create** above, but only **name** or **id** is required. All other parameters are optional. In addition, **desc=** may be specified as blank to delete an existing description.

Changing Time Filtering

timefilter= may be specified as blank (with no other time parameters) to remove all time filtering.

Specifying Time Parameters

To change any of the time-related parameters (**time, month, day, weekday**, or **timefilter**), you must specify all of them again that apply for the time you are specifying. For example:

- **time** is required if (and only if) **weekday** or **month** is not specified
- **weekday** or **month** are mutually exclusive.
- **day** is only applicable with **month**
- day ranges are only applicable with month ranges

In general, the variety of possible specifications can be described as one of the following:

- :*hhmm-hhmm*
- *d*[*-d*][*:hhmm-hhmm*]
- *mmdd*[*-mmdd*][*:hhmm-hhmm*]

where

Value	Equals
hhmm	0000-2359 (**time**)
d	0-6 (Sun-Sat) (**weekday**)
mm	00-11 (Jan-Dec) (**month**)
dd	01-29 (or 30, or 31) (**day**)

cmd=delete
deletes a service.

name=*name* **| id=***id*
name or ID of the service.

force={yes|no}
specifies whether to force the delete if dependent objects are found associated with the service you are deleting. If you do specify **force=yes**, the service is deleted and all references to it are removed. This parameter is optional.

cmd=list
lists a specific service or all services.

name=*name* **| id=***id*
name or ID of the service to be listed. This parameter is optional. If **name** or **id** is not specified, all services are listed.

format={short|long|wide|raw}
is the format of the output. The listing can be in any of the following formats:

Format	Example
short	`id(id1), name(service1), desc(desc1)`
long	is formatted as: keyword = *value*, one parameter per line.

```
  id = id
name = name
desc = desc

⋮
```

wide	same as **short** above, but 132 characters wide.

raw the values are separated by vertical bars, in the following order:

```
id|name|desc|rulelist|log|fragment|tunnel|time|month|day|weekday|timefilter
```

This parameter is optional. The default if no name or ID is specified is **short**; otherwise, it is **long**.

cmd=add
adds a rule or rules to a service.

rulelist=*id1***/{flb},***id2***/{flb},...,***idn***/{flb}**
is a comma-delimited list, by ID, of the rules to add.

service=*service_id_or_name*
is the ID of the new service.

cmd=remove
removes a rule or rules from a service.

rulelist=*id1***/{flb},***id2***/{flb},...,***idn***/{flb}**
is the ID, or list of IDs, of rules to be removed from the service.

service=*service_id_or_name*
ID of the service from which the rules or rules should be removed.

cmd=move
moves a rule within a service (that is, re-orders the rules within a service). The relative order of this service compared to other services remains unchanged.

rule=*id***/{flb}**
is the rule to be moved.

after=[*id***/{flb}]**
is the rule after which the rule specified by **rule=** should be moved. If blank, it means move the specified rule to the top of the list.

service=*service_id_or_name*
ID of the service in which the rules are to be re-ordered.

Usage

A **service** is a set of filter rules that work together to perform a task. For example, six rules work together to allow all of the requests and acknowledgements that make up an FTP transfer; these six rules could be members of a service called "Permit FTP Transfer".

In order to define a new service, you must define the rules that make up the service. The rules may already exist on the firewall or you may add them using the **fwfrule** command.

Just as network groups are made up of objects, the firewall service is made up of filter rules. Before using the **fwservice** command, you must define the filter rules that will be used to build the service. Use the **fwservice cmd=list** command for a list of the rich set of predefined services provided with the OS/390 Firewall Technologies.

See Chapter 8, “Controlling Traffic Through the Firewall with Filters” on page 51 for more information about how filtering works.

Fragmentation: Although previously documented, the **fragment** parameter is no longer applicable. All packet fragments will be reassembled before any filtering is performed. Objects which currently include the **fragment** parameter do not need to be changed. The **fragment** parameter may be specified when using the **fwservice** command.

fragment={yes|no|only|headers}
specifies a fragment-control override value. This is a means of overriding the fragmentation control setting in the rule templates that have been selected for this service. For example, if you include a set of rule templates that ordinarily have fragmentation control set to no, you can override this setting to be yes for the purposes of this service. The override setting will act on all of the rules in this service. Valid values are:

yes match any IP packet, for example, non-fragments, fragment headers and fragments without headers.

no match only non-fragment packets, do not match the fragment headers or fragments without headers.

only match only fragment headers and fragments without a header, do not match non-fragments.

headers match only non-fragments and fragment headers, do not match fragments without headers.

Examples

1. Create two services

```
User:   fwservice cmd=create name="Example Service 1" \
        desc="Contains Filter Rules 1 and 2" rulelist=501/f,501/b,502/b
System:
User:   fwservice cmd=create name="Example Service 2" \
        desc="Contains Filter Rules 1 and 2" rulelist=501/b,502/f timefilter=activate \
        month=Jun-Aug day=15-31
System:
```

2. Show the services just created

```
User:   fwservice cmd=list id=501
System:
                id = 501
              name = Example Service 1
              desc = Contains Filter Rules 1 and 2
          rulelist = 501/f,501/b,502/b
               log =
          fragment =
            tunnel =
              time =
             month =
               day =
           weekday =
        timefilter =
User:   fwservice cmd=list id=502
System:
                id = 502
              name = Example Service 2
              desc = Contains Filter Rules 1 and 2
          rulelist = 501/b,502/f
               log =
          fragment =
            tunnel =
              time =
             month = Jun-Aug
               day = 15-31
           weekday =
        timefilter = activate
```

3. Change a service so that it references a tunnel and logging is specified

```
   User:  fwservice cmd=change name="Example Service 1" tunnel=25 log=yes
System:
   User:  fwservice cmd=list id=501 format=raw
System:   501|Example Service 1|Contains Filter Rules 1 and 2|501/f,501/b,502/b|yes||25|||||
```

4. Move a rule within the rulelist of a service

```
   User:  fwservice cmd=move rule=501/f after=501/b service="Example Service 1"
System:
   User:  fwservice cmd=list name="Example Service 1" format=raw
System:   501|Example Service 1|Contains Filter Rules 1 and 2|501/b,501/f,502/b|yes||25|||||
```

5. Remove rules from a service

```
   User:  fwservice cmd=remove rulelist=501/f,501/b service=501
System:
   User:  fwservice cmd=list name="Example Service 1" format=raw
System:   501|Example Service 1|Contains Filter Rules 1 and 2|502/b|yes||25|||||
```

6. Add rules to a service

```
   User:  fwservice cmd=add rulelist=501/b,501/f service=501
System:
   User:  fwservice cmd=list id=501 format=raw
System:   501|Example Service 1|Contains Filter Rules 1 and 2|502/b,501/b,501/f|yes||25|||||
```

7. Delete a service. Select the force option, so that not only will the service itself be deleted, but all instances of the service will be removed from the connection definitions.

```
   User:  fwservice cmd=delete id=501 force=yes
System:
```

fwsrule

Purpose

Configures Socks filter rules.

Format

```
fwsrule cmd=add
        name=name
       [desc=description]
        type={permit|deny}
       [identd={Default|Always|"If Configured"|"Do Not Use"}]
       [userid=userid1,userid2,...,useridn]
       [operator={eq|neq|lt|gt|le|ge}]
       [port=port_num]        (port_num=0-65535)
       [rulecommand="rule command"]

fwsrule cmd=change
       {name=name | id=id}
       [desc=[description]]
       [type={permit|deny}]
       [identd=[{Default|Always|"If Configured"|"Do Not Use"}]]
       [userid=[userid1,userid2,...,useridn]]
       [operator=[{eq|neq|lt|gt|le|ge}] and
        port=port_num]]           (port_num=0-65535)
       [rulecommand=["rule command"]]

fwsrule cmd=delete
       {name=name | id=id}
       [force={yes|no}]

fwsrule cmd=list
       [{name=name | id=id}]
       [format={short|long|wide|raw}]
```

Keywords and Parameters

cmd=add
adds a Socks filter rule.

name=*name*
is the name of the Socks rule. It must be unique and should not contain a vertical bar (|), a single quote (or apostrophe), or a double quote character. Maximum length is 100 characters.

desc=*description*
is a text description of the rule. Maximum length is 100 characters.

This parameter is optional.

type={permit|deny}
specifies whether this rule is a permit or deny rule.

identd={Default|Always|"If Configured"|"Do Not Use"}]
specifies whether **identd** verification should be used for this rule. The **identd** server provides identification of users in your network. If you are using an **identd** server, use this field to specify how you want the results of the identification used. The User List can also be used by adding IDs that will be matched.

See "fwdaemon" on page 201 for more information about displaying or changing server configuration.

Default Use the identification option specified in the configuration for **sockd**, if any is specified.

Always The Identd must be used to verify the user's identity. Access is denied if a connection to the client's `identd` fails or if the result does not match the user ID reported by the client program. This overrides the setting specified in the configuration for **sockd**.

If Configured
Also specifies the use of `identd`, but denies access only if the client's `identd` reports a user ID different from what the client program claims. This overrides the setting specified in the configuration for **sockd**.

Do Not Use
Do not use the `identd` program. This overrides the setting specified in the configuration for **sockd**.

This parameter is optional.

Note: The **fwsrule** command also supports the following *identd* parameters from previous releases: identd=None, ?=I, ?=i, and ?=n

userid=*userid1***,***userid2***,...,***useridn*
specifies the user ID or IDs to which this rule applies. You can enter a user ID, a list of user IDs, a file name, or a list of file names. If you enter a list, separate the entries with commas. File names must be fully qualified (including the leading "/"). Do not use spaces, tabs, a vertical bar (|) or double quotes(") in the user list.

- The user list is limited to 396 characters.
- User IDs must be IDs of users on the requesting host, not those on the destination host or Socks server host.
- A user ID can consist of 1 to 8 characters, including:
  - a through z
  - A through Z
  - 0 through 9
- A user ID should not contain a vertical bar (|), single quote, or double quote characters(").
- If file names are used, they must be fully qualified (with the leading "/" to prevent their being interpreted as user IDs). Each file can contain a list of user IDs, with one or more per line, separated by commas, and optionally including a comment that is delimited with the # character. Full comment lines - those that begin with the # character are also supported. Each line in the file can be up to 1023 characters long and must be terminated by a "newline" character.

  This file must be created in code page IBM-1047.

This parameter is optional.

operator={eq|neq|lt|gt|le|ge}
this is the logical operator code that represents the logical operation to be performed on the port number:

eq Equal to

neq Not equal to

lt Less than

gt Greater than

le Less than or equal to

ge Greater than or equal to

When used with Port Number, the operator establishes a relationship that must be met. For example, if you enter the Operation gt and Port Number 23, then the port number must be greater than 23 for the rule to be invoked.

This parameter is optional.

port=*port_num* **(port_num=0-65535)**
this is the number of a port that is used with the **operator=** parameter to establish a relationship that must be met for the rule to be invoked. For example, if you enter the operator gt and port number 23, then the port number must be greater than 23 for the rule to be invoked. If this pair is omitted, the rule will apply to all destination port numbers. This parameter is optional.

rulecommand=*rule_command*
the command string to be executed when the conditions in this rule are met. The following substitutions occur before the string is presented to the command shell for execution:

%A replaced by the client host's domain name if known, by its IP address otherwise

%a replaced by the client host's IP address

%c replaced by connect or bind, the command **sockd** is asked to execute

%p replaced by the process ID of **sockd**

%S replaced by the service name (for example, FTP) if known, by the destination port number otherwise

%s replaced by the destination port number

%U replaced by the user ID reported by identd

%u replaced by the user ID reported by the client program

%Z replaced by the destination host's domain name if known, by its IP address otherwise

%z replaced by the destination host's IP address

%% replaced by a single %

You can string together several shell commands in a line with a | or ; symbol.

This parameter is optional.

cmd=change
changes a Socks rule. Parameters are the same as for **add** above, except all but **name=** (or **id**) are optional. In addition, all parameters except **type** may be specified as blank to delete existing values.

cmd=delete
deletes a Socks rule.

name=*name***| id=***id*
is the name or ID of the rule to be deleted.

[force={yes|no}]
specifies whether the delete is forced if there are dependencies on this rule. If "yes" is specified, the rule is deleted and any references to it are removed from services and connections.

cmd=list
lists one or all Socks rules.

name=*name* **| id=***id*
is the name or ID of the rule to be listed. If **name=** or **id=** is not specified, all rules are listed. This parameter is optional.

format={short|long|wide|raw}
is the format of the listing. Following are the formats available:

Format	Description
short	80 columns, formatted as: id(id1), name(rule1), desc(desc1).

long	is formatted as: keyword = *value*, one parameter per line. For example:
	`  id = id`
	`name = name`
	`desc = desc`
	`type = type`
	⋮

wide	132 columns, formatted similar to **short** above.
raw	the values are separated with vertical bars, in the following order: `id|name|desc|type|identd|userid|operator|port|rulecommand` **Note:** *rulecommand* itself may contain vertical bars (shell pipe symbol).

This parameter is optional. The default format is **short** if no name or ID is specified; **long** otherwise.

Usage

fwsrule adds, changes, deletes and lists Socks rules. For more information about the firewall Socks proxy server, see Chapter 9, “Configuring and Using the Socks Server” on page 61 and “Configuring Socks Rules” on page 58.

Examples

1. Add some socks rules

```
User:   fwsrule cmd=add name="Socks Rule 1" desc="Example for Socks Rule 1" type=permit \
        identd=Always userid=smith rulecommand="echo Domain name = %A sockd pid = %p"
System:
User:   fwsrule cmd=add name="Socks Rule 2" desc="Example for Socks Rule 2" type=deny \
        operator=lt port=24
System:
```

2. Show all the socks rules

```
   User:  fwsrule cmd=list format=long
System:
                 id = 502
               name = FTP
               desc = permit socksified ftp
               type = permit
             identd = Default
             userid =
           operator = eq
               port = 21
        rulecommand =

                 id = 501
               name = HTTP
               desc = permit socksified http
               type = permit
             identd = Default
             userid =
           operator = eq
               port = 80
        rulecommand =

                 id = 504
               name = Socks Rule 1
               desc = Example for Socks Rule 1
               type = permit
             identd = Always
             userid = smith
           operator =
               port =
        rulecommand = echo Domain name = %A sockd pid = %p

                 id = 505
               name = Socks Rule 2
               desc = Example for Socks Rule 2
               type = deny
             identd = Default
             userid =
           operator = lt
               port = 24
        rulecommand =

                 id = 503
               name = telnet
               desc = permit socksified telnet
               type = permit
             identd = Default
             userid =
           operator = eq
               port = 23
        rulecommand =
```

3. Change a socks rule

```
   User:  fwsrule cmd=change id=505 type=permit desc="Changed Socks Rule"
System:
   User:  fwsrule cmd=list format=raw
System:  502|FTP|permit socksified ftp|permit|Default||eq|21|
         501|HTTP|permit socksified http|permit|Default||eq|80|
         504|Socks Rule 1|Example for Socks Rule 1|permit|Always|smith|||echo Domain name = %A sockd pid = %p
         505|Socks Rule 2|Changed Socks Rule|permit|Default||lt|24|
         503|telnet|permit socksified telnet|permit|Default||eq|23|
```

4. Delete a socks rule. Select the force option, so that not only will the rule itself be deleted, but all instances of the rule will be removed from the connection definitions.

User: `fwsrule cmd=delete id=504 force=yes`
System:

fwstack

Purpose

Manages TCP/IP firewall stacks.

Format

```
fwstack cmd=add
        stack=stackname
       [force={yes|no}]

fwstack cmd=delete
        stack=stackname
       [force={yes|no}]

fwstack cmd=rename
        oldstack=stackname
        newstack=stackname
       [force={yes|no}]

fwstack cmd=list
       [format={short|raw}]

fwstack cmd=query
```

Keywords and Parameters

cmd=add
adds a new firewall stack name to be managed. The user should have already defined this stack to TCP/IP and added the IPCONFIG FIREWALL statement to the TCP/IP PROFILE.

stack=*stackname*
specifies the name of the stack to be added.

force={yes|no}
specifies whether the add is forced. If "yes" is specified, the stack is added.

cmd=delete
deletes a firewall stack name.

stack=*stackname*
specifies the name of the stack to be added.

force={yes|no}
specifies whether the delete is forced. If "yes" is specified, the stack is deleted.

cmd=rename
renames the firewall stack.

oldstack=*stackname*
specifies the previous stack name.

newstack=*stackname*
specifies the new stack name.

force={yes|no}
specifies whether the rename is forced. If "yes" is specified, the stack is renamed.

cmd=list
lists the firewall stacks.

Format	Description
short	formatted as: stackname1, stackname2, stackname3...stackname8
raw	the values are separated with vertical bars, in the following order: `name1|name2|name3|...name8`

cmd=query
queries the status of the firewall stack.

Usage

You cannot use duplicate stack names. You cannot **delete** a firewall stack that is currently active unless the *force=yes* is used (*no* is the default).

If the old stack name is currently active, you must use the *force=yes* parameter on the **rename** command (*no* is the default). When adding a new stack and the stack is not active, you must use *force=yes*.

When a new stack is created, **fwkern** must be restarted to have the stack recognized.

To determine what to specify for the TCP/IP stackname:

- If INET is used, then the value should be what is specified on the FILESYSTYPE statement (usually INET).
- If CINET is used (FILESYSTYPE ENTRYPOINT(BPXTCINT), then the name is what is specified on the corresponding SUBFILESYSTYPE NAME() statement, assuming that you have set up the TCPIP.DATA and PROCs correctly.

Examples

1. Add some stacks

```
   User:  fwstack cmd=add stack=stack1 force=yes
System:
   User:  fwstack cmd=add stack=stack2 force=yes
System:
   User:  fwstack cmd=add stack=stack3 force=yes
System:
```

2. List the stacks

```
   User:  fwstack cmd=list
System:   TCPIPAA
          STACK1
          STACK2
          STACK3
   User:  fwstack cmd=list format=raw
System:   TCPIPAA|STACK1|STACK2|STACK3
```

3. Delete a stack

```
   User:  fwstack cmd=delete stack=stack1 force=yes
System:
```

4. Rename a stack

```
   User: fwstack cmd=rename oldstack=stack2 newstack=stack4 force=yes
System:
```

5. Query the stacks

```
   User: fwstack cmd=query
System:
                  stack = TCPIPAA
                 status = Up
                   type = Firewall
         active filters = No
         filter logging = Yes
             active NAT = No
            Nat logging = No


                  stack = STACK3
                 status = Down
                   type = Unknown
         active filters = No
         filter logging = No
             active NAT = No
            Nat logging = No


                  stack = STACK4
                 status = Down
                   type = Unknown
         active filters = No
         filter logging = No
             active NAT = No
            Nat logging = No
```

fwtrace

Purpose

Traces firewall server activity.

Note: This command is intended for diagnostic purposes only. For routine logging, use the logging facility. This command must be run from a **superuser**, not a **fwgrp** user.

Format

```
fwtrace -l level -c component_list [-f filename] [-m mode]

fwtrace -q

fwtrace -help
```

Keywords and Parameters

-l signifies the "level" of trace. One of:

on trace is on.

off trace is off.

-c indicates component to be traced. More than one of the following components can be specified:

all all components

admin
administration commands/actions

cfgsrv
configuration server

config
configuration code

ftp ftp server activities

fwkern
controls starting and stopping of firewall daemons

isakmp
ISAKMP server

platform
OS/390 platform-specific code

socks
socks server

-f *filename*
specifies the name of the file to be used as the trace output file. If no file is specified and there is no record of a trace output file in the trace configuration file, the file name defaults to **/var/fw/fwdata/fwtrace.data**.

-m specifies the open intent for the trace output file. One of:

append
write trace output at the end of the trace output file. This is the default option.

replace
replace the current contents of the trace output file.

-q queries trace settings.

-help
returns usage information about the command.

Usage

Trace entries for components that were not explicitly chosen may also appear in the log. These trace entries are created because the explicitly chosen component called another component. For example, if tracing is on only for FTP, when the FTP component calls platform functions, those platform functions will be traced.

If the configuration server (CFGSRV) is started when trace is off for the daemon, each connected GUI user will need to log off and back on for the trace to take effect.

Examples

1. Query status of tracing:

```
fwtrace -q

ICAM0010i Status of IBM Firewall for OS/390 trace facility:
          Trace output file = /var/fw/fwdata/fwtrace.data
          Component = ADMIN     Status = OFF
          Component = CFGSRV    Status = OFF
          Component = CONFIG    Status = ON
          Component = FTP       Status = OFF
          Component = FWKERN    Status = OFF
          Component = ISAKMP    Status = OFF
          Component = PLATFORM  Status = OFF
          Component = SOCKS     Status = OFF
```

fwtunnl

Purpose

Configures IP tunnels.

Format

```
fwtunnl cmd=add
        tunnel=tunnel_id                    (tunnel_id=1-999999)
        type={manual}
        addr=x.x.x.x
        remaddr=x.x.x.x
        policy={ae|be|auth|encr}
       {encrypthow={CDMF|DES_CBC_4|DES_CBC_8|3DES_CBC|ESP_NULL} and/or
            algorithm={KEYED_MD5|HMAC_MD5|HMAC_SHA}}
       [destpolicy={ae|be|auth|encr}
         {destencrypthow={CDMF|DES_CBC_4|DES_CBC_8|3DES_CBC|ESP_NULL} and/or
         destalgorithm={KEYED_MD5|HMAC_MD5|HMAC_SHA}}]

       {spi=spi or destESPspi=spi}
       [srcESPspi=spi]
       [destAHspi=spi]
       [srcAHspi=spi]

       [srcAHauthkey=authenticationkey]
       [destAHauthkey=authenticationkey]
       [srcESPencrkey=encryptionkey]
       [destESPencrkey=encryptionkey]

       [srcESPauth={HMAC_MD5|HMAC_SHA}
       [srcESPauthkey=authenticationkey]]
       [destESPauth={HMAC_MD5|HMAC_SHA}
       [destESPauthkey=authenticationkey]]

       [replay={yes|no}]
       [newheader={yes|no}]
       [timeout=minutes]                    (minutes = 0-44640)
       [mode={transport|tunnel}]
       [autoact={yes|no}]
```

```
fwtunnl cmd=change
        tunnel=tunnel_id                    (tunnel_id=1-999999)
       [addr=x.x.x.x]
       [remaddr=x.x.x.x]
       [policy={ae|be|auth|encr}]
       [encrypthow=[{CDMF|DES_CBC_4|DES_CBC_8|3DES_CBC|ESP_NULL}]]
       [algorithm=[{KEYED_MD5|HMAC_MD5|HMAC_SHA}]]
       [destpolicy={ae|be|auth|encr}]
       [destencrypthow=[{CDMF|DES_CBC_4|DES_CBC_8|3DES_CBC|ESP_NULL}]]

       [destalgorithm=[{KEYED_MD5|HMAC_MD5|HMAC_SHA}]]
       [spi=spi]
       [destESPspi=spi]
       [srcESPspi=spi
       [destAHspi=spi]
       [srcAHspi=spi]

       [srcAHauthkey=[authenticationkey]]
       [destAHauthkey=[authenticationkey]]
       [srcESPencrkey=[encryptionkey]]
       [destESPencrkey=[encryptionkey]]

       [srcESPauth=[{HMAC_MD5|HMAC_SHA}]]
       [destESPauth=[{HMAC_MD5|HMAC_SHA}]]
       [srcESPauthkey=[authenticationkey]]
       [destESPauthkey=[authenticationkey]]

       [replay={yes|no}]
       [newheader={yes|no}]
       [timeout=[minutes]]                  (minutes=0-44640)
       [mode={transport|tunnel}]
       [autoact={yes|no}]

fwtunnl cmd=activate
       [tunnel={tunnel_id1,tunnel_id2,...,tunnel_idn}]

fwtunnl cmd=deactivate
       [tunnel={tunnel_id1,tunnel_id2,...,tunnel_idn}]

fwtunnl cmd=delete
        tunnel={tunnel_id1,tunnel_id2,...,tunnel_idn}

fwtunnl cmd=list
       [directory=directory]
       [tunnel={tunnel_id1,tunnel_id2,...,tunnel_idn}]
       [format={short|long|raw}]
```

```
fwtunnl cmd=import
        directory=directory
        tunnel={tunnel_id1,tunnel_id2,...,tunnel_idn}

fwtunnl cmd=export
        directory=directory
        tunnel={tunnel_id1,tunnel_id2,...,tunnel_idn}
       [version={V2R6|new}]

fwtunnl cmd=shutdown

fwtunnl cmd=listactive
       [tunnel=tunnel_id]
       [stack=stackname]
       [format={short|long|raw}]
```

Keywords and Parameters

cmd=add
adds an IP tunnel.

tunnel=*tunnel_id*
is the numeric tunnel ID. Valid values are numbers 1-999999.

type={manual}
specifies a manual tunnel. (OS/390 Firewall Technologies supports only manual tunnels.)

addr=*x.x.x.x*
IP address of the source firewall machine (which is the tunnel owner).

remaddr=*x.x.x.x*
IP address of the tunnel destination (tunnel partner) machine.

policy={ae|be|auth|encr}
specifies the source tunnel encryption and authentication classification. Values are:

ae authenticate **a**fter **e**ncrypting

be authenticate **b**efore **e**ncrypting

auth authenticate

encr encrypt

Note: If **policy=ae** or **policy=be**, both **encrypthow** and **algorithm** are required.

encrypthow={CDMF|DES_CBC_4|DES_CBC_8|3DES_CBC|ESP_NULL}
specifies the source encryption algorithm used to provide data confidentiality.

Note: The algorithms are ordered from least to most secure.

ESP_NULL
no encryption is used. This encryption type is only valid in combination with a source ESP authentication algorithm. Use of ESP_NULL requires that the new header format parameter be specified as Yes.

CDMF Commercial Data Masking Facility is a limited version of Data Encryption Standard (DES) that uses a 40 bit key during encryption.

DES_CBC_4
DES encryption is used with a 56 bit key and a 32 bit initialization vector.

DES_CBC_8
DES encryption is used with a 56 bit key and a 64 bit initialization vector.

3DES_CBC
Triple DES executes the DES encryption algorithm three times and uses a 24 byte key. Use of 3DES_CBC requires that the new header format parameter be specified as Yes.

algorithm={KEYED_MD5|HMAC_MD5|HMAC_SHA}
specifies the source authentication algorithm used to produce a checksum used for IP packet authentication and integrity verification.

This parameter is optional if **policy=encr** is specified.

Note: The algorithms are ordered from least to most secure. **KEYED_MD5** requires that the new header format parameter be specified as No. **HMAC_MD5** and **HMAC_SHA** requires that the new header format parameter be specified as Yes.

KEYED_MD5
computes the authentication checksum by combining a 128 bit key with the Message Digest-5 (MD5) hash algorithm.

HMAC_MD5
computes the authentication checksum by combining a 128 bit key, the Hash-based Message Authentication Code (HMAC) authentication algorithm and the MD5 hash algorithm.

HMAC_SHA
computes the authentication checksum by combining a 160 bit key, the HMAC authentication algorithm and the Secure Hash Algorithm (SHA) hash algorithm.

destpolicy={ae|be|auth|encr}
specifies the destination tunnel encryption and authentication classification. This parameter is optional. If not specified, the value specified in **policy** is used.

Note: If specified, **destencrypthow** and/or **destalgorithm** are required depending on the value of **destpolicy**. That is, the values of encrythow and algorithm will not be propogated if **destpolicy** is specified.

destencrypthow={CDMF|DES_CBC_4|DES_CBC_8|3DES_CBC|ESP_NULL}
specifies the destination encryption algorithm used to provide data confidentiality. This parameter is optional when **destpolicy** is not specified or when **destpolicy=auth**. If not specified, the value in the **encrypthow** parameter is used. For more information on the encryption algorithms, see the **encrypthow** parameter.

destalgorithm={KEYED_MD5|HMAC_MD5|HMAC_SHA}
specifies the destination authentication algorithm used to produce a checksum used for IP packet authentication and integrity verification. This parameter is optional when **destpolicy** is not specified or when **destpolicy=encr**. If not specified, the value in the **algorithm** parameter is used. For more information on the authentication algorithms, see the **algorithm** parameter.

spi=*spi or* **destESPspi=***spi* **Security Parameter Index of the tunnel partner**
specifies the destination ESP Security Parameter Index. Valid values are 1 - 4294967294.

srcAHspi=*spi*
specifies the source AH Security Parameter Index. Valid values are 1 - 4294967294. This parameter is optional. If not specified, the value in **srcESPspi** is used.

srcESPspi=*spi*
specifies the source ESP Security Parameter Index. Valid values are 1 - 4294967294. This parameter is optional. If not specified, a system generated value is assigned.

destAHspi=*spi*
specifies the destination AH Security Parameter Index. Valid values are 1 - 4294967294. This parameter is optional. If not specified, the value specified in **spi** or **destESPspi** is used.

srcAHauthkey=*authenticationkey*
specifies the source AH authentication key. The key must be specified in hexadecimal prefixed with '0x'. Each byte of the key represents a value 00-ff. The length of the key is determined by the source authentication algorithm specified in the **algorithm** field. The key length (in bytes) for each algorithm type are: KEYED_MD5 and HMAC_MD5 (16), HMAC_SHA (20).

This parameter is optional. If a key is not specified, a system generated key is assigned.

destAHauthkey=*authenticationkey*
specifies the destination AH authentication key. The key must be specified in hexadecimal prefixed with '0x'. Each byte of the key represents a value 00-ff. The length of the key is determined by the destination AH authentication algorithm specified in the **destalgorithm** field. The key length (in bytes) for each algorithm type are: KEYED_MD5 and HMAC_MD5 (16), HMAC_SHA (20).

This parameter is optional. If a key is not specified, a system generated key is assigned.

srcESPencrkey=*encryptionkey*
specifies the source ESP encryption key. The key must be specified in hexadecimal prefixed with '0x'. Each byte of the key represents a value 00-ff. The length of the key is determined by the source encryption algorithm specified in the **encrypthow** field. The key length (in bytes) for each algorithm type are: ESP_NULL (no key), CDMF/DES_CBC_4/DES_CBC_8 (8) and 3DES_CBC (24).

This parameter is optional. If a key is not specified, a system generated key is assigned.

destESPencrkey=*encryptionkey*
specifies the destination ESP encryption key. The key must be specified in hexadecimal prefixed with '0x'. Each byte of the key represents a value 00-ff. The length of the key is determined by the destination encryption algorithm specified in the **destencrypthow** field. The key length (in bytes) for each algorithm type are: ESP_NULL (no key), CDMF/DES_CBC_4/DES_CBC_8 (8) and 3DES_CBC (24).

This parameter is optional. If a key is not specified, a system generated key is assigned.

srcESPauth={HMAC_MD5|HMAC_SHA}
specifies the source ESP authentication algorithm. When specified, ESP authentication also requires that the new header format parameter be specified as Yes.

Note: The algorithms are ordered from least to most secure.

HMAC_MD5
computes the authentication checksum by combining a 128 bit key, the Hash-based Message Authentication Code (HMAC) authentication algorithm and the MD5 hash algorithm.

HMAC_SHA
computes the authentication checksum by combining a 160 bit key, the HMAC authentication algorithm and the Secure Hash Algorithm (SHA) hash algorithm.

This parameter is optional.

srcESPauthkey=*authenticationkey*
specifies the source ESP authentication key. The key must be specified in hexadecimal prefixed with '0x'. Each byte of the key represents a value 00-ff. The length of the key is determined by the source ESP authentication algorithm specified in the **srcESPauth** field. The key length (in bytes) for each algorithm type are: HMAC_MD5 (16), HMAC_SHA (20).

This parameter is optional. If a key is not specified a system generated key is assigned.

destESPauth={HMAC_MD5|HMAC_SHA}
specifies the destination ESP authentication algorithm. When specified, ESP authentication also requires that the new header format parameter be specified as Yes. For more information on the authentication algorithms, see the **srcESPauth** parameter.

This parameter is optional.

destESPauthkey=*authenticationkey*
specifies the destination ESP authentication key. The key must be specified in hexadecimal prefixed with '0x'. Each byte of the key represents a value 00-ff. The length of the key is determined by the destination ESP authentication algorithm specified in the **destESPauth** field. The key length (in bytes) for each algorithm type are: HMAC_MD5 (16), HMAC_SHA (20).

This parameter is optional. If a key is not specified a system generated key is assigned.

replay={yes|no}
Replay prevention monitors incoming IP packets to ensure that the system does not receive identical or old packets. Replay prevention is only valid when new header format is specified. The valid values are Yes and No. The default value is No.

Replay should only be enabled for test purposes.

newheader={yes|no}
New header format specifies whether the AH and ESP headers are built according to RFCs 1825-1829 (**no**) or RFCs 2401-2406 and 2410 (**yes**). Default value depends on the tunnel algorithms specified.

Notes:

1. **newheader** must be set to No when the source and destination AH authentication algorithms are KEYED_MD5.
2. **newheader** must be set to Yes when one or more of the following is true:
   - source or destination AH authentication algorithm is not KEYED_MD5
   - source or destination encryption algorithm is 3DES_CBC or ESP_NULL
   - replay is enabled
   - source or destination ESP authentication is specified

timeout
specifies the lifespan of the tunnel in minutes. This parameter is optional. The default value is 480 minutes (8 hours), and the maximum time allowed is 44640 minutes (31 days). A value of 0 means the tunnel will never expire.

mode={transport|tunnel}
specifies the operational mode of the tunnel.

transport Specifies that the tunnel operates in transport mode, which only protects the transport-layer packet (for example, TCP or UDP packet) inside an IP packet. This mode is used when the endpoints of the secure tunnel are the two communicating systems.

tunnel Specifies that the tunnel operates in tunnel mode, which protects the entire IP packet. Usually, this mode is used for a secure tunnel between two firewalls or between a firewall and a remote system; for example, when either of the two communicating systems is not an endpoint of the tunnel. The default value is **mode=tunnel**.

autoact={yes|no}
defines manual tunnels that should be activated each time its associated Firewall TCP/IP stack is started. Valid values are **yes** or **no**. If not specified, the default value is **no**.

cmd=change
changes a tunnel specification. The parameters used are the same as for **add** above, except that all parameters are optional but

Note: When changing a valid of a particular field, additional related fields may also need to be changed. For example, when changing an encryption algorithm, its key may also need to be changed if the new algorithm key length is different than the length of the old algorithm key length. *tunnel_id*.

cmd=activate
activates a list of tunnels or all tunnels.

tunnel={*tunnel_id1, tunnel_id2, ... ,tunnel_idn***}**
is the numeric tunnel ID or IDs. This parameter is optional; if not specified, all tunnels will be activated.

cmd=deactivate
deactivates a list of tunnels or all tunnels.

tunnel={*tunnel_id1, tunnel_id2, ... ,tunnel_idn***}**
is the numeric tunnel ID or IDs. This parameter is optional; if not specified, all tunnels will be deactivated.

cmd=delete
deletes a list of tunnels.

tunnel={*tunnel_id1, tunnel_id2, ... ,tunnel_idn***}**
is the numeric tunnel ID or IDs.

cmd=list
lists the tunnels in the import or export directory when the **directory** parameter is specified. If the directory is not specified, **cmd=list** lists all defined and imported tunnel definitions known to the firewall.

directory=*directory*
is the fully-qualified name of the directory where the tunnel information is temporarily placed for import/export. This parameter is optional.

tunnel={*tunnel_id1, tunnel_id2, ... ,tunnel_idn***}**
is the ID or IDs of the tunnel(s) whose definition(s) are to be listed. This parameter is optional.

format={short|long|raw}
is the format of the list output. This parameter is optional. The listing can be in any of the following formats:

Format	Example
short	80 columns, formatted as: `tunnel, addr, remaddr, state`
long	is formatted as: keyword = *value*, one parameter per line.

```
tunnel = 5
 state = Inactive
  type = manual
  addr = x.x.x.x

⋮
```

raw the values are separated by vertical bars, in the following order:

```
tunnel|state|type|addr|remaddr|newheader|replay|
autoact|timeout|mode|policy|
algorithm|encrypthow|destpolicy|destalgorithm|destencrypthow|
destESPspi|srcESPspi|destAHspi|srcAHspi|srcAHauthkey|
destAHauthkey|srcESPencrkey|destESPencrkey|srcESPauth|
destESPauth|srcESPauthkey|destESPauthkey
```

cmd=import
imports tunnel definition files from a tunnel owner.

directory=*directory*
is a fully-qualified name of a directory where the tunnel information is temporarily placed for import.

tunnel={*tunnel_id1, tunnel_id2, ... ,tunnel_idn***}**
is the ID or IDs of the tunnel(s) whose definition(s) are to be imported.

cmd=export
exports tunnel definition files to a tunnel partner.

directory=*directory*
is a fully-qualified name of a directory where the tunnel information is temporarily placed for export.

tunnel={*tunnel_id1, tunnel_id2, ... ,tunnel_idn***}**
is the ID or IDs of the tunnel(s) whose definition(s) are to be exported.

version={V2R6|new}
is the format of the tunnel export files.

Enter 'V2R6' to create export files in the format required for tunnel partners running OS/390 Version 2 Release 6 and earlier. Enter 'new' to create the format needed for OS/390 Version 2 Release 7 partners and later. The default is 'new'.

cmd=shutdown
stops all tunnel activity.

cmd=listactive
displays a list of active tunnel definitions and when the tunnel is going to expire.

tunnel={*tunnel_id1, tunnel_id2, ... ,tunnel_idn***}**
is the ID or IDs of the tunnel(s) whose definition(s) are to be listed. This parameter is optional.

stack=*stackname*
an optional parameter to specify the stack name. If the stack name is not specified, the **fwtunnl** command applies to all stacks. If the stack name is specified, the **fwtunnl** command applies to the stack specified by *stackname*.

format={short|long|raw}
is the format of the list output. This parameter is optional. The listing can be in any of the following formats:

Format	Example
short	80 columns, formatted as: `tunnel, addr, remaddr, expiration time`
long	is formatted as: keyword = *value*, one parameter per line.

```
 tunnel = 5
   addr = x.x.x.x
remaddr = x.x.x.x
ExpTime = time
```

raw the values are separated by vertical bars, in the following order:

```
tunnel|addr|remaddr|ExpTime
```

Usage

fwtunnl defines **IP tunnels**. IP tunnels are secure logical connections between two communicating systems. Tunnels provide security in the form of authentication and encryption for the data passing through them. This requires that the systems at each end of the tunnel share a secret key.

Destination parameters are those that are used for outbound traffic while source parameters are those that apply to inbound traffic.

Note: You cannot change or delete the configuration for a tunnel that is currently active. A combination of **Source Address**, **Destination Address** and **Destination ESP SPI** must be unique for active tunnels.

All Security Parameter Index (SPI) values 1-255 are reserved for the Internet Assigned Number Authority.

Examples

1. Add two tunnels and show them, in long format

```
User:   fwtunnl cmd=add tunnel=12 addr=99.100.50.250 remaddr=99.100.5.5 \
        algorithm=KEYED_MD5 type=manual policy=auth encrypthow=CDMF timeout=901 \
        spi=1001
System:
User:   fwtunnl cmd=add tunnel=26 addr=99.100.50.250 remaddr=99.100.10.10 \
        algorithm=KEYED_MD5 type=manual policy=ae encrypthow=DES_CBC_8 timeout=902 \
        spi=1002 mode=transport
System:
User:   fwtunnl cmd=list tunnel=12,26 format=long
System:
           tunnel = 12
            state = Inactive
             type = manual
             addr = 99.100.50.250
          remaddr = 99.100.5.5
        newheader = No
           replay = No
          autoact = No
          timeout = 901
             mode = tunnel
           policy = auth
        algorithm = KEYED_MD5
       encrypthow = CDMF
       destpolicy = auth
    destalgorithm = KEYED_MD5
   destencrypthow = CDMF
       destESPspi = 1001
        srcESPspi = 1069507952
        destAHspi = 1001
         srcAHspi = 1069507952
     srcAHauthkey = 0xa791679d9e37bf042c427e356f151a29
    destAHauthkey = 0xab46c194dc68455e64f1a1210d2206a9
    srcESPencrkey = 0x8368913dc67b2a2b
   destESPencrkey = 0x013dfbf77403658a
       srcESPauth =
      destESPauth =
    srcESPauthkey =
   destESPauthkey =

           tunnel = 26
            state = Inactive
             type = manual
             addr = 99.100.50.250
          remaddr = 99.100.10.10
        newheader = No
           replay = No
          autoact = No
          timeout = 902
             mode = transport
           policy = ae
        algorithm = KEYED_MD5
       encrypthow = DES_CBC_8
       destpolicy = ae
    destalgorithm = KEYED_MD5
   destencrypthow = DES_CBC_8
       destESPspi = 1002
        srcESPspi = 1069709636
        destAHspi = 1002
         srcAHspi = 1069709636
     srcAHauthkey = 0x2abc7a19c79b1cda22b071411c12fccc
    destAHauthkey = 0x324691158cbad0f7a1b890f2475007e7
    srcESPencrkey = 0x3e949432518227cf
   destESPencrkey = 0x3451d3bc68932abf
       srcESPauth =
      destESPauth =
    srcESPauthkey =
   destESPauthkey =
```

2. Change a tunnel

```
User:   fwtunnl cmd=change tunnel=26 policy=be timeout=209
System:
User:   fwtunnl cmd=list tunnel=26
System:
             tunnel = 26
              state = Inactive
               type = manual
               addr = 99.100.50.250
            remaddr = 99.100.10.10
          newheader = No
             replay = No
            autoact = No
            timeout = 209
               mode = transport
             policy = be
          algorithm = KEYED_MD5
         encrypthow = DES_CBC_8
         destpolicy = ae
      destalgorithm = KEYED_MD5
     destencrypthow = DES_CBC_8
         destESPspi = 1002
          srcESPspi = 1069709636
          destAHspi = 1002
           srcAHspi = 1069709636
       srcAHauthkey = 0x2abc7a19c79b1cda22b071411c12fccc
      destAHauthkey = 0x324691158cbad0f7a1b890f2475007e7
      srcESPencrkey = 0x3e949432518227cf
     destESPencrkey = 0x3451d3bc68932abf
         srcESPauth =
        destESPauth =
      srcESPauthkey =
     destESPauthkey =
```

3. Add a tunnel with mode=tunnel (which is the default)

```
User:   fwtunnl cmd=add tunnel=53 addr=99.100.50.250 remaddr=99.100.10.10 type=manual \
        policy=encr encrypthow=DES_CBC_8 spi=1003 mode=tunnel
System:
User:   fwtunnl cmd=list format=raw
System: 1|Inactive|manual|99.100.55.55|11.130.10.6|No|No|No|10|transport|encr|KEYED_MD5|
          CDMF|encr|KEYED_MD5|CDMF|10001|489652161|10001|489652161|
          0x9498526283e0f2520d9dc14d242ee0b5|0xe61f7a40ae0e738043f3c44d4bcac917|
          0xa42070078d1f1bb0|0xe3fb79bf8115fc71||||
        2|Inactive|manual|99.100.55.55|11.130.10.6|No|No|No|480|tunnel|encr|KEYED_MD5|
          DES_CBC_4|encr|KEYED_MD5|DES_CBC_4|10002|496425381|10002|496425381|
          0x0e521973705dcdda96e474a0329f8377|0x25adc80e1a0ef4dae10f4d7f64178fb3|
          0xe9ab137c9f690c7d|0xaedc319200df4605||||
        123|Inactive|manual|99.100.77.77|11.130.10.6|Yes|No|No|480|tunnel|encr|HMAC_MD5|
          3DES_CBC|ae|HMAC_SHA|CDMF|123456|34567|123456|34567|0xfb016dd9d920f1983f4513a99f0426ad|
          0xb0c2ba130e3840d94cf8eb0bedd158b8e34cc4cd|0x23cec81fd679431a231a3e3d6eb380ae432033aef6d4344e|
          0x08a2a79d26105002|HMAC_SHA||0xa4d9a7941c194307d068ef0f20f0d7834298459c|44444|Inactive|manual|
          99.100.77.77|9.130.25.2|Yes|No|No|44640|tunnel|ae|HMAC_SHA|DES_CBC_4|encr|HMAC_SHA|ESP_NULL|
          4294967294|481610715|4294967294|481610715|0xb3d3134af14c1f6ebcaed2bd2a094aaa1a961822|
          0x4cd55783c2b631856de33862987f6fbac83333cb|0xdf6ba151e7e9716e|0x||HMAC_MD5||
          0xdfeaeaf7da7610ab3dcb647ffaa165fe
        12|Inactive|manual|99.100.50.250|99.100.5.5|No|No|No|901|tunnel|auth|KEYED_MD5|CDMF|
          auth|KEYED_MD5|CDMF|1001|1069507952|1001|1069507952|0xa791679d9e37bf042c427e356f151a29|
          0xab46c194dc68455e64f1a1210d2206a9|0x8368913dc67b2a2b|0x013dfbf77403658a||||
        26|Inactive|manual|99.100.50.250|99.100.10.10|No|No|No|209|transport|be|KEYED_MD5|
          DES_CBC_8|ae|KEYED_MD5|DES_CBC_8|1002|1069709636|1002|1069709636|
          0x2abc7a19c79b1cda22b071411c12fccc|0x324691158cbad0f7a1b890f2475007e7|0x3e949432518227cf|
          0x3451d3bc68932abf||||
        53|Inactive|manual|99.100.50.250|99.100.10.10|Yes|No|No|480|tunnel|encr|HMAC_MD5|DES_CBC_8|
          encr|HMAC_MD5|DES_CBC_8|1003|1070163425|1003|1070163425|0x91dca7eca27c347a8d25541ca14bfabf|
          0xa786e0b55db00e516a776c8aca390ab1|0x25f2dadc0a516281|0x2cbf79374ed5a677||||
```

4. Add a Version 2.7.0 enhanced tunnel (newheader=yes)

User: **fwtunnl cmd=add tunnel=270 type=manual addr=99.100.50.250 remaddr=99.100.5.5 **
**policy=encr encrypthow=3DES_CBC destpolicy=encr destencrypthow=3DES_CBC **
**srcESPspi=2700 destESPspi=2701 srcAHspi=2702 destAHspi=2703 **
destESPauth=HMAC_MD5 replay=yes newheader=yes

System:

User: **fwtunnl cmd=list tunnel=270**

System:

```
          tunnel = 270
           state = Inactive
            type = manual
            addr = 99.100.50.250
         remaddr = 99.100.5.5
       newheader = Yes
          replay = No
         autoact = No
         timeout = 480
            mode = tunnel
          policy = encr
       algorithm = HMAC_MD5
      encrypthow = 3DES_CBC
      destpolicy = encr
   destalgorithm = HMAC_MD5
  destencrypthow = 3DES_CBC
      destESPspi = 2701
       srcESPspi = 2700
       destAHspi = 2703
        srcAHspi = 2702
    srcAHauthkey = 0x1623c764e5ce7fdfd28693da6c46747c
   destAHauthkey = 0x3786e389d98cf8a42518fc3b2b330cc2
   srcESPencrkey = 0x0b737f529b8cc19d3b6249a447a00a3a2989cd93c92aa971
  destESPencrkey = 0x9434efb69232e3ad0e1cb0013b1533f7a960eae7ff146d03
      srcESPauth =
     destESPauth = HMAC_MD5
   srcESPauthkey =
  destESPauthkey = 0xf40ec46ed5f81a9b4e6adef117f3468d
```

5. Delete a tunnel

User: **fwtunnl cmd=delete tunnel=12**

System:

Chapter 19. Messages

This section gives you the following information about the OS/390 Firewall Technologies messages:

- How the messages are formatted
- The messages' severity levels
- The messages and their explanations

Message Format

OS/390 Firewall Technologies messages have the following format:

ICA*xxxxa*
-or-
ICAC*xxxxa*

where:

ICA
Product identifier for OS/390 Firewall Technologies (fixed).

C Component identifier.

xxxx
A number in the range 0000–9999.

a An indicator of severity. Messages are classified by severity level. The last byte in the message tag indicates the message severity level.

i—info
w—warning
e—error
r—reply
s—severe

Messages *without* a component identifier (format **ICA***xxxxa*) are logging messages that OS/390 Firewall Technologies issues. Messages *with* a component identifier (format **ICAC***xxxa*) are general messages from firewall configuration and servers.

In logging messages, the number represented by *xxxx* is classified into the following categories depending on its origin:

0000–0999	Log Monitor
1000–1999	Filters
2000–2999	Proxy (with authentication)
3000–3999	Socks
5000–5999	SSLRC
6000–6999	Available
9000–9999	General/Others

© Copyright IBM Corp. 1997, 2000

Messages and Explanations

ICA1005e **Suppressed logging of** *filter_rule_no* **packet message(s) due to buffer overflow**

Explanation: The fwlogd server filter log buffer has overflowed. A packet for the specified filter rule cannot be logged.

User Response: Check the log. Your firewall may be under a denial-of-service attack or you may be logging messages which are not required. For example, broadcast messages should have a deny rule with log control set to no (l=n) to prevent filling up the log.

ICA1032i **Filter rules updated at** *time* **on** *date*

Explanation: IP packet filtering rules have been updated.

ICA1033i **Filter support initialized at** *time* **on** *date*

Explanation: Firewall filter support is initialized.

ICA1034i **Filter support deactivated at** *time* **on** *date*

Explanation: IP packet filtering now using default filter rules rather than those defined in /etc/security/fwfilters.cfg file.

ICA1035i **Status of packet logging set to** *enabled/disabled* **at** *time* **on** *date*

Explanation: Status of packet logging has changed. Message indicates current state with time stamp.

ICA1051i **The following tunnel specification(s) is defined:**

Explanation: Tunnel specifications are being defined, as listed on the following log records.

ICA1052i **>tunnel_ID:***number***, src_addr:***IP_address***, dst_addr:***IP_address***, src_enc:***algorithm* **rem_enc:***algorithm* **src_mac:***algorithm* **rem_mac:***algorithm* **src_enc_mac:***algorithm* **rem_enc_mac:***algorithm* **src_pol:***policy* **rem_pol:***policy* **mode:***transport_mode*

Explanation: Message lists specific attributes of activated tunnel.

ICA1073i *stackname* **R***:.rule_type* **direction***:.interface* **s***:.src_addr* **d***:.dst_addr* **p***:.protocol* **tag***:.scr_port/icmp_type* **tag***:.dst_port/icmp_code* **r***:.routed/local* **a***:.secure/non_secure* **f***:.yes/no* **T***:.tunnel_name* **AH***:.AH SPI* **ESP***:.ESP SPI* **l***:.packet_length*

Explanation: Log record indicating a processed IP packet and the corresponding filter rule it matched. For this record to be written, the matched filter rule must have log control set to *yes*. If the IP packet which matched this rule is a fragment, the ports/icmp type/code information appears for the header packet but is shown as zero for packets other than the header packet.

ICA1075i **Invalid ipsec package: s:***Source IP Address* **d:***Destination IP Address* **protocol:***Protocol* **tunnel name:***Tunnel Name* **AH SPI:***AH Security Parameters Index* **ESP SPI:***ESP Security Parameters Index*

Explanation: The ipsec package cannot be decapsulated by the receiving firewall.

User Response: Ensure that the tunnel is defined and activated correctly on the sending and receiving systems.

ICA1076i **Specification deleted for tunnel:***Tunnel Name* **AH SPI:***AH Security Parameters Index* **ESP SPI:***ESP Security Parameters Index*

Explanation: The specified tunnel definition is deleted.

ICA1077i **Tunnel:***Tunnel Name* **AH SPI:***AH Security Parameters Index* **ESP SPI:***ESP Security Parameters Index* **has expired**

Explanation: The specified tunnel definition has expired. The tunnel has been marked inactive, but has not been deleted.

ICA1078i *stackname* **#***:.rule_no action src_addr src_mask dst_addr dst_mask protocol src_op src_value dst_op dst_value interface_type routing direction* **l=***log_control* **f=***fragment_control <p*

Explanation: When filter rules are updated, the activated rules are written to the log. This log message describes one of the activated rules.

ICA2000i New FTP session to *IP_address* **from** *IP_address* **(non-secure site).**

Explanation: Starting a new ftp session from non-secure site.

ICA2001e Authentication failed for user *name* **(unknown) from** *net ftp*:*IP_address*.

Explanation: A user, without an account, attempted to use ftp proxy from the network.

User Response: See your firewall administrator to setup a proxy account.

ICA2002e Authentication failed for user *name* **with** *authentication method* **from** *network*:*host name*.

Explanation: Firewall is unable to authenticate the indicated user name using the specified authentication method.

User Response: See your Firewall administrator.

ICA2012i New FTP session to *IP_address* **from** *IP_address* **(secure site).**

Explanation: Starting a new ftp session.

ICA2016i Remote user-id: *name*.

Explanation: ftp connection request for indicated user.

ICA2021i Trying to connect to remote host *name* **with user-id** *name*.

Explanation: Trying to establish a new ftp connection.

ICA2022i Trying to connect to remote host *name*.

Explanation: Trying to establish a new ftp connection.

ICA2024i User *name* **successfully authenticated using** *method* **authentication from** *network*:*host name*.

Explanation: FW authenticated the indicated user name using the specified authentication method.

ICA2025i User *name* **logged in using** *method* **authentication from** *network* :*host name*.

Explanation: ftp user logged in.

ICA2026i User *name* **timed out after** *n* **seconds at** *current time*.

Explanation: Connection attempt timed out for specified user. Potential network routing problem or remote host is not available.

ICA2027i Connection from *remote host* **at** *time*.

Explanation: Net ftp connection established to Firewall.

ICA2028e FTP connection attempt to *IP_address* **from** *IP_address* **refused. This machine does not support FTP from non-secure site.**

Explanation: Generally indicates an attempt to establish an ftp connection to Firewall across the non-secure interface.

System Action: Reject the connection.

ICA2041i FTP session *session id* **started for** *user id* **(***source IP addr*:*dest IP addr*).

Explanation: Message generated at the start of each FTP session. A session begins when userid, source ip and destination ip are all known to the firewall. The session id is a unique identifier generated by the firewall.

ICA2044i Incorrect time (*value***) specified for -t.**

Explanation: The time value shown contains characters outside the numeric range of 0..9 or exceeds the maximum allowed value.

ICA2049i Unknown flag -*value* **ignored.**

Explanation: Indicated flag was specified and is not recognized.

ICA2050i Unknown parm *value*.

Explanation: Indicated value, specified as an option, is not recognized.

ICA2051e adapt_addr conversion error on address.

Explanation: IP address shown is not valid.

User Response: Possible corruption of the file /etc/security/fwsecadpt.cfg. Remove the file, reconfigure your secure interface(s) and reinitialize the filters.

ICA2053i Could not open secure interface file.

Explanation: A secure interface has not been configured.

User Response: If a secure interface should be defined, use Firewall commands/smit panels to define the secure interface(s).

ICA2055i getpeername() (*invocation name***):** *errno*.

Explanation: System error when ftp server attempted to get socket name.

ICA2056i getsockname() (*invocation name*): *errno*.

Explanation: System error when ftp server attempted to get port name.

ICA2059i ioctl(): *errno*

Explanation: System error on ioctl() call for SIOCSPGRP.

ICA2066i signal(): *errno*.

Explanation: System error when ftp server attempted to establish signal handler.

ICA2067e Fatal pftpd initialization error - bind(): *errno*

Explanation: pftpd server initialization failed, server terminated. The most likely cause of this error is another ftp server already listening on the standard ftp port (21).

User Response: Correct the indicated system problem and restart pftpd.

ICA2068e Fatal pftpd initialization error - listen(): *errno*

Explanation: pftpd server initialization failed, server terminated.

User Response: Correct the indicated system problem and restart pftpd.

ICA2070e Fatal pftpd initialization error - socket(): *errno*

Explanation: pftpd server initialization failed, server terminated.

User Response: Correct the indicated system problem and restart pftpd.

ICA2072i ftp configuration file (*filename*) is not available.

Explanation: ftp server attempted to open the specified ftp configuration file but it either does not exist or cannot be opened.

System Action: ftp server processing uses the default configuration

User Response: None, unless the file should exist, in which case it should be created or moved to the location specified in the message.

ICA2073e Unable to obtain storage for ftp language table.

Explanation: Storage required to represent a REPLYLANGUAGE statement in the ftp configuration file cannot be obtained.

System Action: Processing continues.

User Response: Increase the region size or reduce the entries in the configuration file.

ICA2075i FTP for *user id* (*source IP addr*:*dest IP addr*), *operation file name*, *numbytes* bytes. sid: *session id*.

Explanation: Message generated for each file transfer on open FTP sessions. The sid is a unique identifier generated by the firewall at session start.

ICA2076i FTP Session *session id* ended for *user id* (*source IP address*:*dest IP addr*), *duration* seconds, *numbytes* bytes.

Explanation: Message generated at the end of each FTP server session. The sid is a unique identifier generated by the firewall at session start.

ICA2080e Syntax error (*reason*) near column *column* in ftp configuration file line *line*: *configuration statement*

Explanation: The ftp configuration statement at the given line is in error. The reason for the error and the location where the error was detected is provided.

System Action: Statement is ignored.

User Response: Correct the statement in the ftp configuration file.

ICA2081e No message catalog given by ftp configuration statements is usable.

Explanation: Attempts to open the message catalogs given by the REPLYLANGUAGE ftp configuration statements failed. No client message catalog can be used.

System Action: Client message catalog is forced to the English language in the C directory.

User Response: Ensure that there are catalog files in each of the directories associated with the language directories in the ftp configuration REPLYLANGUAGE statements. Also check that the NLSPATH environment variable is correctly set to allow substitution of both the sub-directory from the LANG environment variable (%L) and the catalog name (%N).

ICA2082e Unable to set ftp LANG environment variable to *sub-directory*, **reason:** *reason*

Explanation: A system error (given by the reason) occurred when the ftp server was trying to change the setting of the LANG environment variable to the sub-directory specified.

System Action: Processing continues. Recovery may generate other messages.

User Response: Use the reason given to determine if this is a system error or programming error.

ICA2083e Unable to open ftp client message catalog in directory: *sub-directory*, **reason:** *reason*

Explanation: ftp server cannot open the message catalog in the given sub-directory. The reason given is the errno returned from catopen().

System Action: Processing continues. Recovery may generate other messages.

User Response: Ensure that there is a catalog in the directory associated with the language directory provided. Check that the NLSPATH environment variable is correctly set to allow substitution of both the sub-directory (%L) and the catalog name (%N).

ICA2084e Forcing ftp client message catalog to English via the C sub-directory.

Explanation: Due to previously listed errors, the ftp server has forced the client message catalog to the English language using the C sub-directory.

System Action: If the language can be forced to the C message catalog processing continues. If it can not, the program exits.

User Response: Correct the error from the previous messages. If the program also existed, create the message catalog in the C sub-directory and set the NLSPATH environment variable correctly.

ICA2138e Insufficient memory: pftpd: malloc(*bytes***) returned NULL in function** *instance***.**

Explanation: Unable to allocate enough memory - malloc returned NULL.

ICA2141i FTP session to *IP_address* **from** *IP_address* **terminates.**

Explanation: The ftp session to firewall terminates.

ICA2150e Pftpd error - *failing function***: return code =** *function return code*

Explanation: The pftpd server detected an error in the indicated function. The server terminates.

User Response: Correct the indicated system problem and restart pftpd.

ICA2155e Pftpd error - *failing function***:** *system error message*

Explanation: The pftpd server detected an error in the indicated function. The server terminates.

User Response: Correct the indicated system problem and restart pftpd.

ICA2158e Option *value* **was specified incorrectly.**

Explanation: Indicated flag was specified incorrectly.

ICA2159e Timeout value not specified for -t option.

Explanation: A timeout value must be supplied for the -t option.

ICA2160i Password changed for user *user ID* **from** *network* **:***host name***.**

Explanation: An FTP user has successfully changed his password in the password database.

ICA2161i User *user ID* **attempted login using expired password from** *network* **:***host name***.**

Explanation: An FTP user attempted to establish a connection to the Firewall using an expired password.

System Action: The FTP login validation fails and the user is returned to the FTP command shell.

User Response: The user must attempt to validate again through the FTP USER command or by re-establishing the FTP connection and passing the password string of the form "old_password/new_password/new_password".

ICA2162e Password change failure for user *user ID* **from** *network* **:***host name***.**

Explanation: An FTP user attempted to change his password and the password validation routine failed. The possible reasons for the failure include: (1) Incorrect "old" password was specified, (2) Only one occurrence of the "new" password was specified, (3) Two occurrences of "new" password do not match, or (4) Delimiter used to separate passwords was not "/".

System Action: FTP password validation fails and the user is returned to the FTP command shell.

User Response: Attempt to re-validate with the FTP server ensureing the passwords are being entered correctly. If the problem persists, contact the service representative.

ICA2167 *user id* **authentication failed for** *service* **using** *authentication scheme* **from** *client address* **on** *network type*

Explanation: The specified user failed to be authenticated for the specified service using the specified authentication method. The user was requesting the service from the indicated address and network type. System Action : The user authentication fails.

User Response: Contact system administrator.

ICA2188e Incorrect time (*value***) specified for -I.**

Explanation: The time value shown contains characters outside the numeric range of 0..9 or exceeds the maximum allowed value.

ICA2189e Timeout value not specified for -I option.

Explanation: A timeout value must be supplied for the -I option.

ICA3001e *Alert*: real user is *ident user name***, not** *socks connect user name*

Explanation: Possible security breach attempt, user name not authenticated.

ICA3010i connected -- Bind from *user*(*real_user*)@*src_addr* **for** *dst_addr* (*destination port*)

Explanation: Connection established.

ICA3011i connected -- Connect from *user*(*real_user*)@*src_addr* **to** *dst_addr* (*application*)

Explanation: Successful socket connection to outside world.

ICA3012w refused -- Connect from *user*(*real_user*)@*src_addr* **to** *dst_addr* (*application*)

Explanation: Remote host refused connection.

ICA3013w select() *errno*

Explanation: System error.

ICA3014i terminated -- Bind from *user*(*real_user*)@*src_addr* **for** *dst_addr* (*destination port*).(*count* **bytes from** *client*, *count* **bytes from** *server*)

Explanation: Connection terminated.

ICA3015i terminated -- Connect from *user*(*real_user*)@*src_addr* **to** *dst_addr* (*destination host*).(*count* **bytes from** *client*, *count* **bytes from** *server*)

Explanation: Connection to server terminated.

ICA3016w ***Cannot find appropriate interface to communicate with *destination host*

Explanation: File /etc/sockd.route does not contain routing information for the specified destination host.

ICA3018w refused -- Bind from *user*(*real_user*)@*src_addr* **for** *dst_addr*

Explanation: Remote host refused connection.

ICA3019e Error in GetDst() from host *socks_src_name*: *errno*

Explanation: Error in resolving destination address for requested connection.

ICA3030w Unable to open config file (*/etc/sockd.conf*)

Explanation: Open request against indicated file failed.

ICA3032w Unable to open userfile (*user name file***):** *errno*

Explanation: The filename specified for *=userlist on a permit rule cannot be found.

ICA3033w Unexpected result from Validate()

Explanation: Identd verification of the user name was specified, Identd responded with unexpected result.

ICA3035w Cannot connect to identd on *client host*

Explanation: Identd verification of the user name was specified, Identd does not respond.

ICA3039e Error -- shell command "*cmd***" contains no alphanumeric characters.**

Explanation: Invalid shell command, see log message.

ICA3040e Error -- shell_cmd fork() *errno*

Explanation: Sockd server unable to switch to child process via 'fork()'

ICA3041e Error -- unable to get client address.

Explanation: Error return from 'getpeername()' call.

User Response: Check routing and DNS configuration.

ICA3042e Error -- undefined command (*hex-command-received***) from host** *client address*

Explanation: Invalid command received from client application.

User Response: Possible client configuration problem, or mismatch on client and Firewall support level.

ICA3043e Error -- wrong version (*hex-version-number***) from host** *client address***.**

Explanation: Firewall supports socks version 4.2.

User Response: Possible client configuration problem, or mismatch on client and Firewall support level.

ICA3044e Failed -- Connect from *user***(***real_user***)@***src_addr* **to** *dst_addr* **(***application***). Error code:** *command causing failure errno***.**

Explanation: Connection request failed.

ICA3045e Failed -- Bind from *user***(***real_user***)@***src_addr* **for** *dst_addr***. Error: connected to wrong host** *dst_name* **(***dst_port (application)***).**

Explanation: Bind request failed.

ICA3046e Failed -- Bind from *user***(***real_user***)@***src_addr* **for** *dst_addr***. Error code:** *command causing failure errno***.**

Explanation: Bind request failed.

ICA3047w Timed-out -- Bind from *user***(***real_user***)@***src_addr* **for** *dst_addr*

Explanation: Connection timed out.

ICA3048w Shell command too long: *command***...**

Explanation: The command to be executed, from the /etc/sockd.conf file, is too long.

ICA3049w Timed-out -- Connect from *user***(***real_user***)@***src_addr* **to** *dst_addr* **(***application***)**

Explanation: Connection timed out.

ICA3050i *matched sockd.conf filter rule*

Explanation: Filter rule from the /etc/sockd.conf file which matched the socks connection.

ICA3101w Sockd error sending data - select(): *system error message*

Explanation: (SOCKS422) Error while sending data.

ICA3102w Sockd error sending data - write(): *system error message*

Explanation: (SOCKS422) Error while sending data.

ICA3103w Sockd error receiving data - select(): *system error message*

Explanation: (SOCKS422) Error while receiving data.

ICA3104w Sockd error receiving data - read(): *system error message*

Explanation: (SOCKS422) Error while receiving data.

ICA3107w Set inbound socket SO_LINGER option failed: *system error message*

Explanation: (SOCKS422) not critical

ICA3108w Set outbound socket SO_LINGER option failed: *system error message*

Explanation: (SOCKS422) not critical

ICA3109w Invalid entry at line *line number* **in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3111w Illegal destination IP at line *line number* **in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3112w Illegal destination mask at line *line number* **in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3113i Parsed *number of lines* **lines in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3115w Invalid 'permit/deny' field at line *line number* **in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3116w Invalid '?=' field at line *line number* **in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3117w Illegal source IP at line *line number* **in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3118w Illegal source mask at line *line number* **in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3119w Invalid comparison at line *line number* **in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3120w Invalid port number at line *line number* **in file** *filename***.**

Explanation: (SOCKS422) Incorrect configuration entry syntax.

ICA3123i Sockd server starting.

Explanation: (SOCKS422) Sockd has successfully initialized and is awaiting connections.

ICA3124e Fatal sockd initialization error - bind(): *system error message*

Explanation: (SOCKS422) Sockd server initialization failed, server terminated.

User Response: Correct the indicated system problem and restart sockd.

ICA3125e Fatal sockd initialization error - listen(): *system error message*

Explanation: (SOCKS422) Sockd server initialization failed, server terminated.

User Response: Correct the indicated system problem and restart sockd.

ICA3128e Fatal sockd initialization error - socket(): *system error message*

Explanation: Sockd server initialization failed, server terminated.

User Response: Correct the indicated system problem and restart sockd.

ICA3129e Fatal sockd initialization error - *failing function***:** *system error message*

Explanation: Sockd server initialization failed in the indicated function, server terminated.

User Response: Correct the indicated system problem and restart sockd.

ICA3130e Sockd error - *failing function***:** *system error message*

Explanation: The sockd server detected an error in the indicated function. The server continues, but connections may be refused or terminated.

User Response: If the problem persists, stop sockd, correct the indicated system problem and restart sockd.

ICA3131w Error reading *file name***. Previously cached data will be used.**

Explanation: The file cannot be read or contained incorrect data. A previous message should describe the problem. Sockd will continue to operate with cached data from the previous version of the file.

User Response: Correct the error in the indicated file.

ICA3132e Unknown flag -*value***.**

Explanation: The indicated flag is not recognized, server terminated.

User Response: Correct the syntax and restart sockd.

ICA3133e Unknown parameter *value***.**

Explanation: The indicated parameter is not recognized, server terminated.

User Response: Correct the syntax and restart sockd.

ICA3134e Conflicting options *option1* **and** *option2***.**

Explanation: The indicated options cannot be specified together, server terminated.

User Response: Correct the syntax and restart sockd.

ICA3135e Sockd error - *failing function***: return code =** *function return code*

Explanation: The sockd server detected an error in the indicated function. The server terminates.

User Response: Correct the indicated system problem and restart sockd.

ICA3136e Sockd error - *failing function***:** *error message*

Explanation: The sockd server detected an error in the indicated function. The server terminates.

User Response: Correct the indicated system problem and restart sockd.

ICA8050e *function* **failed.** *error_text*

Explanation: An error was encountered executing the function displayed in the message. Additional information about the error is given by the error text.

User Response: Correct the error specified in the message and, if necessary, retry the operation.

ICA8051e *function* **failed: return code =** *return_code*

Explanation: An error was encountered executing the function displayed in the message. The return code from the specified function is also displayed.

User Response: Correct the error specified in the message and, if necessary, retry the operation.

ICA8052i FWSTACKD activating filter logging for *stack_name***.**

Explanation: FWSTACKD is attempting to activate packet filter logging.

System Action: The program continues.

ICA8053e FWSTACKD cannot activate filter logging for *stack_name***.** *error_text*

Explanation: Activation of packet filter logging failed for the reason described in the accompanying error message.

System Action: Filter logging will not be performed.

User Response: Use the error message to correct the error, then reactivate filters logging with **fwfilter cmd=startlog**.

ICA8054i FWSTACKD activating NAT logging for *stack_name***.**

Explanation: FWSTACKD is attempting to activate network address translation (NAT) logging.

System Action: The program continues.

ICA8055e FWSTACKD cannot activate NAT logging for *stack_name***.** *error_text*

Explanation: Activation of network address translation (NAT) logging failed for the reason described in the accompanying error message.

System Action: Network address translation logging will not be performed.

User Response: Use the error message to correct the error, then reactivate network address translation logging with **fwnat cmd=startlog**.

ICA8056i FWSTACKD activating NAT for *stack_name***.**

Explanation: FWSTACKD is attempting to activate network address translation (NAT).

System Action: The program continues.

ICA8057e FWSTACKD cannot activate NAT for *stack_name***.** *error_text*

Explanation: Activation of network address translation (NAT) failed for the reason described in the accompanying error message.

System Action: Network address translation will not be performed.

User Response: Use the error message to correct the error, then reactivate network address translation with **fwnat cmd=update**.

ICA8058i FWSTACKD reactivating tunnel definitions for *stack_name***.**

Explanation: FWSTACKD is attempting to reactivate tunnel definitions that were active when the system was stopped.

System Action: The program continues.

ICA8059e FWSTACKD cannot reactivate tunnel definitions for *stack_name***.** *error_text*

Explanation: Activation of tunnel definitions failed for the reason described in the accompanying error message.

System Action: Tunnels definitions are not activated.

User Response: Use the error message to correct the error, then reactivate tunnel definitions with **fwtunnl cmd=activate**.

ICA8060i FWSTACKD activating filter and Socks rules for *stack_name*.

Explanation: FWSTACKD is attempting to activate the current set of packet filter rules and Socks rules.

System Action: The program continues.

ICA8061e FWSTACKD cannot activate filter and Socks rules for *stack_name*. *error_text*

Explanation: Activation of filter rules and Socks rules failed for the reason described in the accompanying error message.

System Action: Default filter rules will be in effect. Local access will be permitted, and all other access will be denied.

User Response: Use the error message to correct the error, then reactivate filters and Socks rules with **fwfilter cmd=update**.

ICA8062i FWSTACKD activating RealAudio support for *stack_name*.

Explanation: FWSTACKD is attempting to activate RealAudio support.

System Action: The program continues.

ICA8063e FWSTACKD cannot activate RealAudio support for *stack_name*. *error_text*

Explanation: Activation of RealAudio support failed for the reason described in the accompanying error message.

System Action: RealAudio services are unavailable.

User Response: Use the error message to identify the error, then fix the error and activate RealAudio with **fwaudio cmd=change**.

ICA8064e *function* failed. *error_text*

Explanation: An error was encountered executing the function displayed in the message. Additional information about the error is given by the error text.

User Response: Correct the error specified in the message and, if necessary, retry the operation.

ICA8100e Proposed life value (*proposed_life*) not in policy life range (*policy_minlife* - *policy_maxlife*).

Explanation: An ISAKMP negotiation failed because the life value specified by the initiator does not fall within the life value specification of the responder.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the life values specified fall within the responder's life values.

ICA8101e Transform *transform_id*: unable to store DH group (*DH_group_id*) in policy record.

Explanation: An ISAKMP negotiation failed because the verification of the transforms in a proposal was unable to store the specified Diffie-Hellman group identifier. The most likely reason is that there is a mismatch of group identifiers in the proposal.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the DH group values are the same.

ICA8102e Transform *transform_id*: unsupported life-type attribute (*lifetype_id*).

Explanation: An ISAKMP negotiation failed because the value specified for the life-type attribute is not SECONDS or KBYTES.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the only life-types that are specified are SECONDS or KBYTES.

ICA8103e Transform *transform_id*: key length specified for fixed key length algorithm.

Explanation: An ISAKMP negotiation failed because a key length is specified for an algorithm that has a fixed key length.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that key lengths are not specified for the hash algorithm, encryption algorithm, or authentication method for fixed key length algorithms.

ICA8104e Transform *transform_id*: life value not specified for life-type attribute.

Explanation: An ISAKMP negotiation failed because a transform specified a life-type attribute with no corresponding life value.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that life-type attribute has a matching life value.

ICA8105e Transform *transform_id*: unsupported attribute tag (*attribute*).

Explanation: An ISAKMP negotiation failed because a transform specified an attribute that is not known for the specified protocol.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initi-

ator and responder and ensure that all attributes are valid for the specified protocol(s).

ICA8106e Transform *transform_id*: attribute length (*attr_length*) too long for attribute (*attribute*).

Explanation: An ISAKMP negotiation failed because a transform specified an attribute length that is too large for the specified attribute.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that all attributes specify a valid attribute value. Most attributes are single valued attributes and contained within 4 bytes.

ICA8107e Transform *transform_id*: unable to read attribute(s).

Explanation: An ISAKMP negotiation failed because the attributes cannot be read in the specified transform.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that attributes are correctly formed. However, it is likely that an internal error has occured.

ICA8108e Transform *transform_id*: no encapsulation mode specified.

Explanation: An ISAKMP negotiation failed because the specified transform does not specify the encapsulation mode.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that attributes include encapsulation mode.

ICA8109e Transform *transform_id*: no authentication algorithm specified.

Explanation: An ISAKMP negotiation failed because the specified transform does not specify the authentication algorithm.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that attributes include authentication algorithm.

ICA8110e Unable to set *transform_function* function for *protocol* transform (*transform_id*).

Explanation: An ISAKMP negotiation failed because the ISAKMP server cannot identify the specified function to operate on transforms for the given protocol.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8111e Unable to sort *policy_proposal* attributes.

Explanation: An ISAKMP negotiation failed because the ISAKMP server cannot sort the attributes specified in either the proposal or in the local policy.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8114e Unable to layout *protocol* transform header: id = *transform_id*.

Explanation: An ISAKMP negotiation failed because the ISAKMP server cannot build the header for the specified protocol.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8115e Transform *transform_id*: attribute length (*attr_length*) for attribute (*attr_name*) is not valid.

Explanation: An ISAKMP negotiation failed because the length of the specified attribute in the transform is not valid.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that all attributes specify a valid attribute value. Most attributes are single valued attributes and contained within 4 bytes.

ICA8116e Transform *transform_id*: unsupported *protocol* transform.

Explanation: An ISAKMP negotiation failed because the transform specified for this protocol contains an unsupported encryption algorithm.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the encryption algorithm values are valid.

ICA8117e Transform *transform_id*: unsupported protocol *protocol*.

Explanation: An ISAKMP negotiation failed because the specified transform contained an unknown protocol.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the only protocols specified are AH, ESP, or IKE.

ICA8118e Unsupported algorithm (*algorithm*) found in transform.

Explanation: An ISAKMP negotiation failed because the algorithm specified in a transform is not supported.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that only valid hash, encryption, and authentication algorithms are specified.

ICA8119e Transform *transform_id*: too short.

Explanation: An ISAKMP negotiation failed because a transform specified in the policy or proposal does not contain all of the necessary data.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8120e Transform *transform_id*: unexpected attribute (*attribute*).

Explanation: An ISAKMP negotiation failed because a transform specified in the policy or proposal contains an unexpected attribute. The attrribute may be out of order.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8121e Transform *transform_id*: more than 2 SA life-type attributes specified.

Explanation: An ISAKMP negotiation failed because a transform specified more than the supported number of life-type attributes.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that no more than 2 life-type attributes are specified in each transform; one SECONDS life-type and one KBYTES life-type.

ICA8122e Transform *transform_id*: more than 1 *life_value* life value specified.

Explanation: An ISAKMP negotiation failed because a transform specified more than one life value for the specified life-type.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that there is only 1 life value for each life-type attribute.

ICA8123e Transform *transform_id*: attribute *attribute* should be fixed length.

Explanation: An ISAKMP negotiation failed because a transform specified an attribute that is expected to be fixed length, and is not fixed.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that all attributes specify a valid attribute value. Most attributes are single valued attributes and should be containable within 4 bytes.

ICA8124e Transform *transform_id*: duplicate attribute (*attribute*) specified.

Explanation: An ISAKMP negotiation failed because a transform specified the same attribute more than the allowed number of times.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that all attributes are specified correctly. With the exception of the life-value tag, all attributes can be specified only once in a single transform.

ICA8125e Transform *transform_id*: too long.

Explanation: An ISAKMP negotiation failed because a transform specified in the policy or proposal contains extra data.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8126e Incorrect attribute (*attribute_id*) specified in *policy_proposal*.

Explanation: An ISAKMP negotiation failed because an unknown or unsupported attribute was found in the policy or proposal data.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initi-

ator and responder and ensure that all attributes specified are valid for the specified protocol.

ICA8127e *function* **function not found for** *policy_proposal* **attribute (***attribute***).**

Explanation: An ISAKMP negotiation failed because the specified function cannot be found for the specified attribute.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8128e *function* **function failed for attribute (***attribute***).**

Explanation: An ISAKMP negotiation failed because the specified function failed for the specified attribute.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8129e *data* **was not found or specified.**

Explanation: An ISAKMP negotiation failed because the specified data was not provided or available.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8130e *function_name* **failed: rc =** *return_code***.**

Explanation: An ISAKMP negotiation failed because the specified function returned an unexpected return code.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8131e Phase I negotiation and protocol (*protocol_id***) is not ISAKMP.**

Explanation: An ISAKMP Phase I negotiation failed because the specified protocol is not ISAKMP (the only valid choice for Phase I negotiation).

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the protocol for all key policies is ISAKMP.

ICA8132e *function_name* **failed.**

Explanation: An ISAKMP negotiation failed because the specified function returned an unexpected result.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8133e No proposal data to lay out.

Explanation: An ISAKMP negotiation failed because the specified proposal is empty.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8134e Proposal and policy mismatch in *function_name***.**

Explanation: An ISAKMP negotiation failed because the specified function detected a mismatch in the proposed and local policies.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the chosen policies specify compatible attributes.

ICA8135e No proposal chosen in *function_name***.**

Explanation: An ISAKMP negotiation failed because the specified function detected that no proposal satisfies the local policy.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the chosen policies specify compatible attributes.

ICA8136e *function_name***(***input_value***) failed.**

Explanation: An ISAKMP negotiation failed because the specified function detected that no proposal satisfies the local policy.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the chosen policies specify compatible attributes.

ICA8140e Transform *transform_id*: no encryption algorithm specified.

Explanation: An ISAKMP negotiation failed because the specified transform does not specify the encryption algorithm.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that attributes include the encryption algorithm.

ICA8141e Transform *transform_id*: no hash algorithm specified.

Explanation: An ISAKMP negotiation failed because the specified transform does not specify the hash algorithm.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that attributes include the hash algorithm.

ICA8142e Transform *transform_id*: no DH group identifier specified.

Explanation: An ISAKMP negotiation failed because the specified transform does not specify the DH group identifier.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that attributes include the DH group identifier.

ICA8143e Transform *transform_id*: attribute (*attribute*) should be variable length.

Explanation: An ISAKMP negotiation failed because the specified transform specified an attribute that is expected to be variable length.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8145e *function_name* failed: *reason*.

Explanation: An ISAKMP negotiation failed because the specified function returned with an unexpected error string.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8146e Unsupported protocol (*protocol*) for Phase *SA_phase* SA negotiation.

Explanation: An ISAKMP negotiation failed because a protocol was found that is not valid in the current SA negotiation phase. Phase I negotiation supports only the ISAKMP (IKE) protocol and Phase II supports only the ESP and AH protocols.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the protocol is specified correctly.

ICA8147e Proposal (*proposal*) is out of order; last proposal was *last_proposal*.

Explanation: An ISAKMP negotiation failed because the current proposal number is out of order. Proposals must be processed in monotomically increasing order.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8148e ISAKMP protocol specified with non-ISAKMP protocol in same proposal.

Explanation: An ISAKMP negotiation failed because the current proposal contains both ISAKMP and non-ISAKMP protocol. During Phase I, the only protocol allowed is ISAKMP (IKE).

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the protocol is specified correctly.

ICA8149e Proposal contains *number* extra bytes.

Explanation: An ISAKMP negotiation failed because the current proposal contains extra data.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8150e Proposal does not match; no local policy data available.

Explanation: An ISAKMP negotiation failed because the current proposal cannot be matched to a local policy.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8151e Initiator of SA cannot specify MODE=ANY.

Explanation: An ISAKMP negotiation failed because the initiator of the ISAKMP exhange specified MODE=ANY.

System Action: The SA negotiation failed.

User Response: Check the policy of the initiator and ensure that the MODE specified is INITIATOR.

ICA8152e Unsupported exchange flag (*xchg_flag*) for Phase *phase* SA.

Explanation: An ISAKMP negotiation failed because the ISAKMP exchange flag specifies an exchange that is not valid for the current phase.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8153e Client ID(s) must be IPV4 type addresses.

Explanation: An ISAKMP negotiation failed because the the client IDs specified in the ISAKMP messages are not IPV4 type addresses.

System Action: The SA negotiation failed.

User Response: Check the policies of the initiator and responder of the ISAKMP exchange and ensure that the client IDs are IPV4 type addresses.

ICA8154e Static filter rule matched for dynamic connection from *src_id*:*src_id* (*src_protocol*/*src_port*) to *dst_id*:*dst_id* (*dst_protocol*/*dst_port*).

Explanation: An ISAKMP negotiation failed because a search of filter rules matched a static filter rule instead of an anchor or dynamic filter rule. The rules are defined or ordered incorrectly.

System Action: The SA negotiation failed.

User Response: Correct or reorder the filter rules using the **fwfrule**, **fwservice** and **fwconns** commands.

ICA8155e Bad proposal layout for protocol (*protocol*).

Explanation: An ISAKMP negotiation failed because the proposal layout for the specified protocol is not valid.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8156e Wrong SPI size (*SPI_size*) in protocol (*protocol*) in proposal (*proposal*).

Explanation: An ISAKMP negotiation failed because an SPI size in a proposal is not valid.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8157e More than 1 *object* in proposal reply for protocol (*protocol*).

Explanation: An ISAKMP negotiation failed because there is more than one transform specified on a reply or more than one proposal was chosen on a match of an offered proposal.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8158e Duplicate protocol (*protocol*) found in proposal (*proposal*).

Explanation: An ISAKMP negotiation failed because the reply proposal contains more than one transform with the specified protocol.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8159e Proposal (*proposal*) contains *number* too few bytes.

Explanation: An ISAKMP negotiation failed because the proposal does not contain enough data.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8160e No proposal specified or proposal is empty.

Explanation: An ISAKMP negotiation failed because a proposal does not exist or it contains no data.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8162e Protocol ID (*protocol*) mismatch.

Explanation: An ISAKMP negotiation failed because the proposal and policy contain conflicting protocols.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the protocol is specified correctly.

ICA8163e Proposal too short or SPI size too large.

Explanation: An ISAKMP negotiation failed because the proposal does not contain enough data or the SPI size in the transform is too large.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8164e Proposal contained *number_proposed* transforms; reply built *number_reply*.

Explanation: An ISAKMP negotiation failed because the proposal and reply transform numbers mismatch. A reply should pick at least one transform for each protocol that is offered.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8165e Reply check: proposal and policy transform mismatch.

Explanation: An ISAKMP negotiation failed because the proposal and reply transforms do not match. A reply should only pick a transform that matches one of the proposed transforms.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8166e Protocol *protocol* is not supported.

Explanation: An ISAKMP negotiation failed because the protocol identifier in the transform is not valid.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8167e DH group ID (*group_id*) is not supported.

Explanation: The ISAKMP server detected a Diffie-Hellman group that is not supported. The currently supported DH groups are 1 and 2.

System Action: The ISAKMP negotiation failed.

User Response: Check the policies of the initiator and responder of the ISAKMP exchange and ensure that the DH group identifiers are correct.

ICA8168e Unknown exception caught in *function_name*.

Explanation: An ISAKMP negotiation failed because the specified routine caught an unknown exception.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8169e Exception (*exception*) caught in *function_name*.

Explanation: An ISAKMP negotiation failed because the specified routine caught the specified exception.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8170e Alarm fired; no timer function to execute.

Explanation: An ISAKMP alarm fired and there is no timer function to execute.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8171e Attempted to insert a duplicate SA.

Explanation: An attempt to insert an SA into the SA hash bucket detected a duplicate SA.

System Action: The SA insertion failed.

User Response: Contact the service representative.

ICA8172e SA identifies an unknown negotiation phase (*phase*).

Explanation: An SA contains a negotiation phase that is not Phase I or Phase II.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8173e SA bucket is empty or does not exist.

Explanation: The hash bucket that should contain the SA is empty or does not exist.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8174e SA chain corruption detected.

Explanation: The hash bucket SA chain is currupted.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8175e Hash bucket does not exist.

Explanation: The hash bucket that should contain the specified SA does not exist.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8176e No certificate factory found for DOI *doi*.

Explanation: A certificate factory does not exist for the specified DOI.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8177e DOI (*doi*) is already registered.

Explanation: An attempt to register a DOI with the ISAKMP server determined that the DOI is already registered.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8178e Unable to build seed for cookie generation.

Explanation: An attempt to generate a seed to use in generating the SA cookies failed.

System Action: The SA negotiation failed.

User Response: Contact the service representative.

ICA8179e Unknown or incorrect address type (*address_type*).

Explanation: The specified address type is not supported by the ISAKMP server.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8180e Unknown or incorrect address length (*address_length*).

Explanation: The specified address length is not supported by the ISAKMP server.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8181e Key Server entry does not contain an IP address or host name.

Explanation: The key server entry for a remote or local key server does not specify either an IP address or a host name for the key server.

System Action: The SA negotiation failed.

User Response: Check the key server entries of both the initiator and responder and ensure that either an IP address or a host name are specified.

ICA8182e Phase I key exchange is not complete.

Explanation: The ISAKMP server attempted to send a notification of a Phase II SA and the Phase I key exchange is not complete.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8183e Initiator must process responder notification messages.

Explanation: The ISAKMP server received a notification message directed to the responder and is currently acting as the initiator.

System Action: The message request failed.

User Response: Contact the service representative.

ICA8184e Refresh of Phase II should be performed.

Explanation: The ISAKMP server received a notification message that a phase II refresh should have occurred.

System Action: The message request failed.

User Response: Contact the service representative.

ICA8185e Attribute length (*attribute_length*) is not valid.

Explanation: The ISAKMP server received a notification message with an incorrect attribute length.

System Action: The message request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8186e Unknown message type (*message_type*).

Explanation: The ISAKMP server received a message type that is not supported by ISAKMP.

System Action: The message request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8187e Cannot determine IP address from ID type (*id_type*).

Explanation: The ISAKMP server received a client ID that does not map to an IP address.

System Action: The request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8188i Dynamic tunnel *tunnel_name* refreshed.

Explanation: The ISAKMP server successfully refreshed a dynamic tunnel.

System Action: Processing continues.

User Response: No action required.

ICA8189e Must be in Phase I to initialize the KEP.

Explanation: The ISAKMP server received a message to start a KEP (Key Exchange Protocol) and the current phase is not phase I.

System Action: The request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8190e No transform specified in KEP initialization.

Explanation: The ISAKMP server received a message to start a KEP (Key Exchange Protocol) and no transform was specified.

System Action: The request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8191e No factory exists for KEP transform (*transform_id*).

Explanation: The ISAKMP server received a message to start a KEP (Key Exchange Protocol) and no factory exists for the specified transform.

System Action: The message request failed.

User Response: Contact the service representative.

ICA8192e Input request: length mismatch.

Explanation: The ISAKMP server received a message that contains a length that is not the expected length of the message.

System Action: The message request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8193e Unknown request type (*request_id*).

Explanation: The ISAKMP server received a message that contains an unknown message type.

System Action: The message request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8194e Cannot generate request to anchor.

Explanation: The ISAKMP server is unable to send a request to the anchor module.

System Action: The message request failed.

User Response: Contact the service representative.

ICA8195e Unexpected user request lengths: *header_length*, *message_length*.

Explanation: The ISAKMP server received a message that contains lengths that are not expected or are incorrect for the input request.

System Action: The message request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8199e Unable to locate SA by cookies.

Explanation: The ISAKMP server attempted to locate an SA by the Icookie/Rcookie pair and the SA was not found.

System Action: The ISAKMP negotiation fails.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8200e Rcookie is zero; message ID is non-zero.

Explanation: The ISAKMP server detected that an Rcookie is zero, but the message ID in the negotiation is non-zero. A non-zero message ID indicates a Phase II message (where the cookies are required).

System Action: The ISAKMP negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8201e Transform *transform_id*: DH group (*DH_group*) is not supported.

Explanation: An ISAKMP negotiation failed because the DH group identifier specified is not valid. The valid DH groups are 1 and 2.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the DH group is specified correctly.

ICA8203e Transform *transform_id*: unsupported authentication type (*auth_type*).

Explanation: An ISAKMP negotiation failed because the authentication type is not supported. The valid authentication types are pre-shared key and digital certificate.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the authentication method is pre-shared key or digital certificate.

ICA8204e Phase *SA_phase* SA already exists.

Explanation: The ISAKMP server detected a request to create an SA, and the SA already exists.

System Action: The ISAKMP request failed.

User Response: No action required.

ICA8205e *attribute_name* is already stored.

Explanation: The ISAKMP server detected a request to store an attribute that was already stored.

System Action: The ISAKMP negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8206e Unable to resume with CERT request message.

Explanation: The ISAKMP server attempted to resume with a certificate request that was unable to be resumed.

System Action: The certificate request failed.

User Response: Contact the service representative.

ICA8207e Phase *SA_phase* SA has expired.

Explanation: The ISAKMP server detected a request to access an SA, and the SA already expired.

System Action: The ISAKMP request failed.

User Response: To satisfy the request, the SA must be refreshed or re-started.

ICA8208e Phase II KEP missing required algorithm or method.

Explanation: The ISAKMP server detected a request for a Phase II initiation, and the required parameters are missing.

System Action: The ISAKMP negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the parameters required for a data connection are specified.

ICA8209e Error trying to compute or set Phase II IV.

Explanation: The ISAKMP server attempted to compute or set the initialization vector for a Phase II message exchange and the attempt failed.

System Action: The ISAKMP negotiation failed.

User Response: Contact the service representative.

ICA8210e Expected a life-type value; none was found.

Explanation: The ISAKMP server received a request to modify the life of an SA with no life-type value specified.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8211e Unknown life-type value (*life_value*) was detected.

Explanation: The ISAKMP server received a request to modify the life of an SA with no valid life-type value specified.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8212e Expected a life value; none was found.

Explanation: The ISAKMP server received a request to modify the life of an SA with no valid life value specified.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8213e Specified life value is too big.

Explanation: The ISAKMP server received a request to modify the life of an SA with no valid life value specified.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8214e Adjusting SA life required both life-type and life value; neither found.

Explanation: The ISAKMP server received a request to modify the life of an SA with no valid life values specified.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8215e Attribute length is too small.

Explanation: The ISAKMP server detected an attribute that is not big enough to store the value for the attribute.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8216e Too many attributes were specified.

Explanation: The ISAKMP server detected an attribute list that contains too many attributes for the requested action.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8217e Payload is too large.

Explanation: The ISAKMP server detected a payload that contains too much data for the expected payload layout.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8218e No relevant payload was received.

Explanation: The ISAKMP server detected a message that does not contain a payload required to handle the request.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8219e Received message is not encrypted.

Explanation: The ISAKMP server detected a message that was not encrypted.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8220e Received message is a replay message.

Explanation: The ISAKMP server detected a message that was already processed by ISAKMP.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8221e Message exchange is in unknown state.

Explanation: The ISAKMP server detected a message that is not in an expected state.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8222e Duplicate *payload* **payload encountered.**

Explanation: The ISAKMP server detected a payload that was already received, or received more payloads of this type than expected.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8223e Unexpected payload type (*payload_type***).**

Explanation: The ISAKMP server detected a payload type that is not expected or supported.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8224e Unexpected message; not a replay.

Explanation: The ISAKMP server received a message that is not expected and is not a replay message.

System Action: The ISAKMP message will be ignored.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8225e No *payload_name* **payload received.**

Explanation: The ISAKMP server detected that it has not processed a payload of the specified type and that it should have already received the payload.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8226e Unknown *object* **(***object_type***).**

Explanation: The ISAKMP server detected a transform or payload that is not supported.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8227i Dynamic tunnel *tunnel_name* **created from** *src_id***:***src_id* **protocol:***src_protocol* **port:***src_port* **to** *dst_id***:***dst_id* **protocol:***dst_protocol* **port:***dst_port* **encralg:***algorithm* **encr_authalg:***algorithm* **encr_mode:***transport_mode* **authalg:***algorithm* **auth_mode:***transport_mode* **SA lifetime:***seconds* **SA lifesize:***Kbytes* **connection lifetime:***seconds* **PFS:***DHgroup***.**

Explanation: The ISAKMP server successfully negotiated a dynamic tunnel. The log entry displays the fields that will most uniquely identify the new tunnel.

System Action: Processing continues.

User Response: No action required.

ICA8228e Certificate type (*cert_type*) is not supported.

Explanation: The ISAKMP server read a certificate that is not supported. The only supported certificate type is currently X.509 signature mode certificates.

System Action: The ISAKMP request failed.

User Response: Check the policies of both the initiator and responder and ensure that the correct certificate types are specified.

ICA8229e No entry found in DNS for remote ID (*remote_ID*).

Explanation: The ISAKMP server attempted to locate the host name for the specified remote ID and the DNS lookup failed.

System Action: The ISAKMP request failed.

User Response: Check the policies of both the initiator and responder and ensure that the IDs are specified correctly. If the problem continues, contact the service representative.

ICA8230e Pre-shared key contains non-hexadecimal data.

Explanation: The ISAKMP server detected a pre-shared key that contains non-hexadecimal data. A pre-shared key must consist of digits 0 through 9 and characters A through F, and contain an even number of bytes.

System Action: The ISAKMP request failed.

User Response: Check the pre-shared key database and ensure that the data specified is hexadecimal and formed correctly, containing an even number of bytes.

ICA8231e Incorrect payload size (*payload_len*).

Explanation: The ISAKMP server detected a payload that is not the correct size for the expected payload type.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8232e Unknown address type (*address_type*) detected.

Explanation: The ISAKMP server detected an address type that is not expected or supported by the ISAKMP server.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8233i Attempting to create dynamic connection from *source_nwobj* to *dest_nwobj*.

Explanation: The ISAKMP server is starting a dynamic connection between the specified network object identifiers. Upon successful completion of the dynamic connection, message ICA8227i will be displayed.

System Action: Processing continues.

User Response: No action required.

ICA8234e Connection name *short_connection_name* not found.

Explanation: The ISAKMP server attempted to locate the short name for a connection and the lookup failed.

System Action: The ISAKMP request failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8235e Retransmit time-out is reached.

Explanation: The ISAKMP server exhausted the re-transmit limit set for a single message re-transmission.

System Action: The ISAKMP request failed.

User Response: This is normal if there are network problems or the partner ISKAMP server is not responding. Make sure both ends of the connection are active and try the original request again.

ICA8236e Unknown parameter *value*.

Explanation: During initialization of the ISAKMP server, the value, specified as an option, is not recognized.

System Action: The ISAKMP server will not start.

User Response: Use the **fwdaemon** command to correct the **daemonopts** for the ISAKMP server.

ICA8237e Incorrect value (*value*) specified for *parameter*.

Explanation: The value shown contains characters outside the numeric range of 0 through 9 or exceeds the maximum allowed value.

System Action: The ISAKMP server will not start.

User Response: Use the **fwdaemon** command to correct the **daemonopts** for the ISAKMP server.

ICA8238e Value not specified for *parameter* option.

Explanation: A value must be supplied for the specified option.

System Action: The ISAKMP server will not start.

User Response: Use the **fwdaemon** command to correct the **daemonopts** for the ISAKMP server.

ICA8239e Unknown or unsupported option (*option*) specified.

Explanation: The option specified on the startup of the ISAKMP server is not supported.

System Action: The ISAKMP server will not start.

User Response: Use the **fwdaemon** command to correct the **daemonopts** for the ISAKMP server.

ICA8240e Public key algorithm value is not RSA encryption.

Explanation: An ISAKMP negotiation failed because the public key algorithm value found in the certificate is not RSA encryption. This occurred when verifying the signature of the remote ISAKMP server.

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's certificate contains the correct public key algorithm value. Try the negotiation again.

ICA8241e The notAfter time in the certificate expired.

Explanation: An ISAKMP negotiation failed because the "notAfter" time in the certificate expired. This occurred when verifying the initiator's or responder's signature.

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's certificate contains a valid "notAfter" time. Try the negotiation again.

ICA8242e The X.500 *attribute* attribute length (*length*) is not valid.

Explanation: The ISAKMP negotiation failed because the X.500 attribute length is either too long or too short. This occurred while DER encoding an X.500 character string.

System Action: The SA negotiation failed.

User Response: Contact a service representative.

ICA8243e Multiple instances of the X.500 *attribute* attribute are found.

Explanation: The ISAKMP negotiation failed because multiple instances of the specified X.500 attribute were found. This is not supported. This occurred while DER encoding an X.500 string.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8244e Responder mode in policy does not match initiator mode.

Explanation: The ISAKMP negotiation failed because the mode specified for the responder in the policy database is not consistent with the role specified by the initiator of the SA negotiation.

System Action: The SA negotiation failed.

User Response: Check the policies of both the initiator and responder and ensure that the initiator and responder modes are specified correctly.

ICA8245e The OSS call failed : *function*.

Explanation: The function returned incorrect information, or a failed return code.

System Action: The action will fail, if during a negotiation, the SA negotiation failed.

User Response: Contact a service representative.

ICA8246e The OCSF function *function name* failed: rc = *return code*.

Explanation: The Open Cryptographic Services Facility (OCSF) function call did not complete successfully.

System Action: If this occurs during the start up of the server, the server may not start or certificate actions may not be possible. If this occurs during a negotiation, the negotiation will fail.

User Response: Ensure that the OCSF package was installed correctly on the system, then contact a service representative.

ICA8247e Anchor filter not defined for connection from *src_id*:*src_id* (*src_protocol*/*src_port*) to *dst_id*:*dst_id* (*dst_protocol*/*dst_port*).

Explanation: An ISAKMP negotiation failed because an anchor filter that matched the selector values cannot not be found.

System Action: The SA negotiation failed.

User Response: Create or correct the anchor filter using the **fwfrule** command.

ICA8248e The notBefore time in the certificate expired.

Explanation: An ISAKMP negotiation failed because the "notBefore" time in the certificate expired. This occurred when verifying the initiator's or responder's signature.

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's certificate contains a valid "notBefore" time. Try the negotiation again.

ICA8249e Cannot be an initiator of a dynamic tunnel negotiation.

Explanation: The local ISAKMP server is attempting to initiate a dynamic tunnel and the local policy specifies that it can only act as a reponder.

System Action: The dynamic tunnel negotiation failed.

User Response: Check the ISAKMP server's initiation role using the **fwdyntun** command. If the local ISAKMP server should be able to activate a dynamic tunnel, use the **fwdyntun** command to change the server's **init** role to **local** or **either**.

ICA8250e Cannot be a responder in a dynamic tunnel negotiation.

Explanation: The local ISAKMP server is attempting to respond to a dynamic tunnel negotiation request and the local policy specifies that it can only act as an initiator.

System Action: The dynamic tunnel negotiation failed.

User Response: Check the ISAKMP server's initiation role using the **fwdyntun** command. If the local ISAKMP server should be able to be a responder in a dynamic tunnel negotiation, use the **fwdyntun** command to change the server's **init** role to **remote** or **either**.

ICA8251e Unsupported ID type *id_type***.**

Explanation: An ISAKMP negotiation failed because an ID of an unsupported type was specified.

System Action: The SA negotiation failed.

User Response: Ensure that the ID is defined correctly, then try the request again.

ICA8252e ID value too long.

Explanation: An ISAKMP negotiation failed because an ID was specified that exceeded the maximum allowed ID size.

System Action: The SA negotiation failed.

User Response: Ensure that the ID is defined correctly, then try the request again.

ICA8253e *extension* **certificate extension is not found.**

Explanation: An ISAKMP negotiation failed because the certificate extension was not found while decoding the certificate. This extension is required as part of the ISAKMP negotiation process for certificates.

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's certificate contains the specified certificate extension. Try the negotiation again.

ICA8254e Certificate version (*version***) is not supported.**

Explanation: An ISAKMP negotiation failed because the certificate version was not at the correct level. It must be at X.509 V3 (this equates to a value of 2 in the decoded certificate).

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's certificate is at the X.509 V3 level. Try the negotiation again.

ICA8255e No certificate extensions were found.

Explanation: An ISAKMP negotiation failed because no certificate extensions were found while decoding the certificate. Certain extensions such as the subject alternate name and key usage extensions are required as part of the ISAKMP negotiation process for certificates.

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's certificate contains certificate extensions such as the key usage and subject alternate name. Try the negotiation again.

ICA8258e Unmatched quote found in X.500 string (*string***).**

Explanation: An ISAKMP negotiation failed because an unmatched quote was found while DER encoding an X.500 character string.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured cor-

rectly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8259e Length of X.500 string (*length*) is too long.

Explanation: An ISAKMP negotiation failed because the length of an X.500 character string was greater than the maximum allowable size. This occurred while DER encoding an X.500 character string.

System Action: The SA negotiation failed.

User Response: Contact a service representative.

ICA8260e The X.500 string contains an unsupported type (*X.500_type*).

Explanation: The X.500 character string contains an unsupported type. This occurred while DER encoding the X.500 character string. The ISAKMP negotiation failed.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8261e Object ID (OID) is not support.

Explanation: An Object ID (OID) that is not currently supported was found during the decoding of a DER-encoded X.500 string. The ISAKMP negotiation failed.

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's X.500 DER-encoded ID is defined correctly. Try the negotiation again.

ICA8264e Usage value (*usage_value*) not supported.

Explanation: An ISAKMP negotiation failed because the usage value is not supported. The value currently supported is the integer representation of the digital signature flag. This occurred when verifying the signature of remote ISAKMP server's certificate.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8265e DigitalSignature flag not set in key usage certificate extension (*key_usage_value*).

Explanation: An ISAKMP negotiation failed because the "digitalSignature" flag was not set in the key usage certificate extension. This occurred when verifying the signature of the remote ISAKMP server's certificate.

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's certificate contains the correct key usage certificate extension. Try the negotiation again.

ICA8266e Local time (*time_in_seconds*) to notBefore time (*time_in_seconds*) comparison failed.

Explanation: An ISAKMP negotiation failed because the local time was less than the certificate's "notBefore" time. This occurred when verifying the initiator's or responder's signature.

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's certificate contains the correct "notBefore" time. Try the negotiation again.

ICA8267e Lifetime plus local time (*time_in_seconds*) to notAfter time (*time_in_seconds*) comparison failed.

Explanation: An ISAKMP negotiation failed because the lifetime plus the local time was greater than the certificate's "notAfter" time. This occurred when verifying the initiator's or responder's signature.

System Action: The SA negotiation failed.

User Response: Ensure that remote ISAKMP server's certificate contains the correct "notAfter" time. Try the negotiation again.

ICA8268e Usage value (*usage_value*) not supported.

Explanation: An ISAKMP negotiation failed because the usage value is not supported. The value currently supported is "SIG". This occurred when verifying the signature of remote ISAKMP server's certificate.

System Action: The SA negotiation failed.

User Response: Ensure that the initiator and responder of the ISAKMP exchange are configured correctly. Also ensure that the system which has generated the incorrect data is operating correctly.

ICA8269i The ISAKMP server is not set up to support certificates.

Explanation: The ISAKMP server cryptography level does not support certificates or certificate actions. This is because the OCSF cryptographic package is not fully installed or the **fwkeyring** configuration file does not have a valid key ring name.

System Action: The ISAKMP server will not process the command.

User Response: Verify that the OCSF cryptographic package is fully installed with trust policy, certificate library, and data library modules present. Then verify that your **fwkeyring** configuration file is set up correctly and has a valid key ring name.

ICA8270e No certificates in the Certificate Authority cache.

Explanation: The **fwcertauth** command could not complete because there are no certificates in the cache.

System Action: The command is ignored by the ISAKMP server.

User Response: Check the configuration file using the **fwcertauth** command.

ICA8271e OCSF cryptographic package is not installed correctly.

Explanation: OCSF cryptographic package is either not present or is not installed correctly on the system.

System Action: The ISAKMP server will not start.

User Response: Verify that the OCSF cryptographic package is fully installed with trust policy, certificate library, and data library modules present.

ICA8272i *file_name* **configuration file was not found, or is incorrect.**

Explanation: The configuration file was not found or is not correct.

System Action: ISAKMP server will not support certificate actions.

User Response: Verify that the specified file is present and valid.

ICA8273e setenv() of the RSA Compliance variable failed.

Explanation: The RSA compliance variable was not able to be set or unset in accordance with the RSA contract with Firewall.

System Action: The ISAKMP server will not start because the OCSF cryptography package will not be usable.

User Response: Contact the service representative.

ICA8274i The *file_name* **configuration file has an incorrect entry :** *id* **.**

Explanation: The configuration file specified has an incorrect entry for the ID specified. If the ID = 0, then the specific line is unknown to the ISAKMP server.

System Action: Continue parsing the file and ignore the specified entry.

User Response: Verify that the specified entry is correct.

ICA8275e Certificate Cache error, no Cache present.

Explanation: The certificate cache is not present.

System Action: The SA negotiation fails.

User Response: Verify that the certificates you require on the key ring specified by **fwkeyring** are attached. Verify that the OCSF package has been correctly installed on the system.

ICA8276e Certificate field value of algorithm or usage is incorrect.

Explanation: The ISAKMP server only supports RSA signing. The certificate or information is incorrect.

System Action: The SA negotiation fails.

User Response: Verify that all of the certificates are used for RSA signing. Verify that your peer server is using RSA signature certificates only.

ICA8277e Certificate field value of ID type is not valid.

Explanation: The ISAKMP server only supports four ID types; IPV4, FQDN, user@FQDN, X.500 Distinguised Name (DN). The ID type specified is not one of those specified.

System Action: The SA negotiaion fails.

User Response: Verify all of the certificates are using one of the specified ID types. Verify that your peer server is using only those specified ID types.

ICA8278e The ISAKMP server has no matching certificate entry for the specified ID(*ID_string***).**

Explanation: The ISAKMP negotiation failed because no matching certificate entry was found. This occurred while searching for a certificate that matched the local ISAKMP server ID.

System Action: The SA negotiation fails.

User Response: Verify that the correct local key server ID is being used when negotiating with this peer. If it is correct, obtain a certificate with the expected ID

of the local ISAKMP server. Once the certificate is obtained, add it to the ISAKMP key ring.

ICA8279e Cannot get the ID information from the peer's certificate.

Explanation: The certificate information was not retrievable from the peer's certificate.

System Action: The SA negotiation fails.

User Response: Verify that your peer's certificate is a valid certificate and that it is compatible with the ISAKMP server.

ICA8280e Peer's certificate is not valid.

Explanation: The certificate provided by your peer cannot be verified.

System Action: The SA negotiation fails.

User Response: Verify that you have the peer's CA certificate. Contact your peer and verify that this certificate is valid.

ICA8281e Certificate value specified is not found.

Explanation: The information provided in the command has no match in the certificate cache.

System Action: The command is ignored by the ISAKMP server, however, unless otherwise stated, the configuration file is updated.

User Response: Verify that the specified information is correct.

ICA8282e Hardware support through RACF using ICSF is not supported.

Explanation: The specified certificate stores the private key in ICSF. The system is not currently set up to support ICSF.

System Action: The SA negotiation fails.

User Response: Verify that the key is stored in ICSF. If it is stored in ICSF, configure the system to use ICSF, or use a certificate that does not store the private key in ICSF.

ICA8285i Certificate Authority Cache error, no Cache present.

Explanation: The certificate authority cache is not present.

System Action: The SA negotiation continues.

User Response: Verify that the certificates you require are specified in the **fwcertauth** command.

ICA8286i Certificate authority label does not match the entries in the cache: *label*.

Explanation: The certificate authority cache does not contain the object specified.

System Action: The SA negotiation continues.

User Response: Verify that the certificates you require are specified in the **fwcertauth** command and they are attached to the key ring specified in the **fwkeyring** command.

ICA8287e The RACF call IRRSDL00 failed the return codes are: saf rc = *saf_rc*, racf rc = *racf_rc*, racf rs = *racf_rs*.

Explanation: There is an error with the RACF database.

System Action: Certificate functions are not allowed. The server will function in pre-shared key mode only.

User Response: Contact the service representative.

ICA8289i Certificate authority not added to certificate request payload.

Explanation: An encoded certificate authority X.500 name was not added to the certificate request payload during the ISAKMP Phase 1 negotiation. The reason for this action is that the label of the certificate authority certificate specified in the **fwcertauth** command could not be found in the certificate authority cache. This label was determined from the **fwauthinfo** data associated with the remote key server.

System Action: The ISAKMP server continues processing the negotiation. However, the negotiation may fail as the ISAKMP server will not be able to verify the peer's certificate.

User Response: Verify that the **fwauthinfo** data associated with the remote key server points to the correct certificate authority (**fwcertauth**). Also, verify that the certificate authority certificate label specified in the **fwcertauth** command exists and it is attached to the key ring specified in the **fwkeyring** command.

ICA8290e The X.500 attribute contains all white spaces.

Explanation: The ISAKMP negotiation failed because the X.500 attribute contains all white spaces. This occurred while either searching the certificate cache or while verifying the initiator's or responder's identity.

System Action: The SA negotiation fails.

User Response: Check the key server entries for the local and remote ISAKMP servers. Ensure that these key server entries for the X.500 authentication ID strings do not contain all white spaces for any of the X.500 attributes such as CN or OU.

ICA8291e The X.500 attribute string length(*string_length*) is not valid.

Explanation: The ISAKMP negotiation failed because the end of the X.500 attribute string is less than the beginning of the string. This should not occur during regular ISAKMP server processing.

System Action: The SA negotiation fails.

User Response: Contact the service representative.

ICA8292e The X.500 ID(*X.500_string*) in the ID payload does not match the subject name(*X.500_string*) in the certificate.

Explanation: The ISAKMP negotiation failed because the X.500 DER-encoded strings mismatch. This mismatch occurred while verifying the initiator's or responder's identity.

System Action: The SA negotiation fails.

User Response: Check the key server entries for the local and remote ISAKMP servers. Ensure that these key server entries for the X.500 authentication ID strings match their associated certificates.

ICA8293e Internal error processing X.500 distinguished name(*internal_error*).

Explanation: The ISAKMP negotiation failed because of an internal error. This error occurred when comparing X.500 strings while either searching the certificate cache or while verifying the initiator's or responder's identity.

System Action: The SA negotiation fails.

User Response: Contact the service representative.

ICA8294e The ISAKMP server has no matching certificate entry for the specified ID(*ID_string*), signed by the certificate authority(*X.500_string*) as requested by the peer.

Explanation: The ISAKMP negotiation failed because no matching certificate entry was found. This occurred while searching for a certificate that matched both the local ISAKMP server ID and the certificate authority specified by the peer.

System Action: The SA negotiation fails.

User Response: Verify that the correct local key server ID is being used when negotiating with this peer. If it is correct, obtain a certificate with the expected ID of the local ISAKMP server and signed by the certificate authority specified by the peer. Once the certificate is obtained, add it to the ISAKMP key ring. Alternately, tell the ISAKMP peer to request a certificate signed by a different certificate authority.

ICA8295e The ID(*id_string*) in the ID payload does not match any of the subject alternate names in the certificate.

Explanation: The ISAKMP negotiation failed because the identity in the ID payload does not match any of the subject alternate names in the certificate. This mismatch occurred while verifying the initiator's or responder's identity.

System Action: The SA negotiation fails.

User Response: Check the key server entries for the local and remote ISAKMP servers. Ensure that the key server entries for the IPV4, FQDN, or USER@FQDN authentication ID strings match their associated certificates.

ICA8296i ISAKMP security association *SA_name* created from *src_id* to *dst_id* exchange type:*exch_type* auth method:*auth_method* encralg:*algorithm* hashalg:*algorithm* DHGroup:*DHgroup* lifetime:*seconds* lifesize:*Kbytes*.

Explanation: The ISAKMP server successfully negotiated an ISAKMP security association with a remote key server. The log entry displays the characteristics of the new security association.

System Action: Processing continues.

User Response: No action required.

ICA8297i ISAKMP security association *SA_name* refreshed.

Explanation: The ISAKMP server successfully refreshed an ISAKMP security association.

System Action: Processing continues.

User Response: No action required.

ICA8298i Attempting to create on demand connection from *source_ip_specification* to *dest_ip_specification*.

Explanation: The ISAKMP server is starting an on demand dynamic connection between the specified IP address specifications. Upon successful completion of the dynamic connection, message ICA8227i will be displayed.

System Action: Processing continues.

User Response: No action required.

ICA8299e *function* **failed.**

Explanation: The specified function returned an unexpected result. This function was not critical to the establishment of the current security association; however, it may inhibit the ISAKMP server from exploiting a performance optimization.

System Action: The SA negotiation continued.

User Response: If this happens repeatedly, notify IBM service.

ICA9006i Ending remote configuration server.

Explanation: Configuration server is ending.

ICA9008e Remote configuration server failed on getpeername(): error *errno***.**

Explanation: Configuration server is unable to obtain information about the client.

User Response: See your FW administrator.

ICA9009e Remote configuration server failed on getsockname(): error *errno***.**

Explanation: Configuration server is unable to obtain information about itself.

User Response: See your FW administrator.

ICA9010e Remote configuration server failed obtaining adapter information.

Explanation: Configuration server is unable to obtain adapter information.

User Response: See your FW administrator.

ICA9011e Configuration server not enabled for remote configuration.

Explanation: Configuration server has local=yes set in its configuration file and the client is on a remote machine.

User Response: See your FW administrator.

ICA9012e Remote configuration server unable to read logon request.

Explanation: Configuration server cannot read in the client logon request.

User Response: See your FW administrator.

ICA9013e Remote configuration server received incorrect logon request.

Explanation: Logon request contained incorrect information.

User Response: See your FW administrator.

ICA9032i NAT configuration updated at *time* **on** *date***.**

Explanation: NAT configuration is updated.

ICA9033i NAT support (level *version.release***) initialized at** *time* **on** *date***.**

Explanation: Firewall NAT support is initialized.

ICA9034i NAT support deactivated at *time* **on** *date***.**

Explanation: NAT support is disabled.

ICA9035i NAT unable to allocate Registered Address for Secured Address *Secured IP Address***.**

Explanation: Secured Address not translated because there are no available addresses in the Registered Address pool.

ICA9036i NAT released Registered Address *Registered IP Address* **to address pool.**

Explanation: Registered Address is released to registered IP address pool.

ICA9060e Fatal configuration server initialization error - socket(): *system error message*

Explanation: Configuration server initialization failed, server ended.

User Response: Correct the specified system problem and start the configuration server again.

ICA9061e Fatal configuration server initialization error - listen(): *system error message*

Explanation: Configuration server initialization failed, server ended.

User Response: Correct the specified system problem and start the configuration server again.

ICA9062e Fatal configuration server error - main accept(): *system error message*

Explanation: Configuration server main routine failed, server ended.

User Response: Correct the specified system problem and start the configuration server again.

ICA9063e Configuration server error - *failing function***: return code =** *function return code*

Explanation: The configuration server detected an error in the specified function. The server ends.

User Response: Correct the specified system problem and start the configuration server again.

ICA9064i Unknown option -*value* ignored.

Explanation: Specified option is not recognized.

ICA9065e Configuration server error - *failing function*: *system error message*

Explanation: The configuration server detected an error in the specified function. The server ends.

User Response: Correct the specified system problem and start the configuration server again.

ICA9066e Insufficient memory: configuration server: malloc(*bytes*) returned NULL in function *function_name*.

Explanation: Unable to allocate enough memory - malloc returned NULL.

ICA9068e -*value* option failed or was specified incorrectly.

Explanation: The indicated option failed or was specified incorrectly.

System Action: The configuration server ends.

User Response: Correct the usage of the specified option and start the configuration server again.

ICA9069e SSL Initialization failed.

Explanation: The SSL encryption environment was unable to be initialized or the handshake with the partner failed.

System Action: The configuration server ends.

User Response: See your Firewall administrator and ensure the SSL environment.

ICAC0001i Command completed successfully.

Explanation: The command completed successfully.

System Action: Command completed.

User Response: None.

ICAC0002e Parameter '*parameter*' is not needed.

Explanation: The specified parameter should not appear when issuing this command.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again without the extra parameter. To see the command's syntax, issue the command without any parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC0003w These objects are now empty.

Explanation: A 'delete' command was issued with the **force** parameter. The 'force' parameter resulted in all objects or rules referencing the deleted object or rule to be updated. A rule or object is now empty. For example, network group object ATEAM contains network object user1 as its only member. The deletion of user1 results in ATEAM becoming an empty network group object.

System Action: Command completes.

User Response: Verify that empty objects or rules should exist.

ICAC0004w Command completed, but no more non-secure adapters are defined.

Explanation: The command completed successfully. The last non-secure adapter was changed to secure. There is now no adapter that communicates with the non-secure network.

System Action: Command completed.

User Response: Verify that a non-secure adapter is not needed.

ICAC0005w '*parameter*' parameter is already set to '*value*'.

Explanation: The value specified for the parameter was previously set.

System Action: Processing of the command ends. The system status remains the same.

User Response: None.

ICAC1000e Command failed.

Explanation: The command failed. A message identifying the cause of the failure will also be issued.

System Action: Processing of the command is ended. The system status remains the same.

User Response: Issue the command again with appropriate parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1001e '*parameter*' is not a recognized parameter.

Explanation: The command was issued with an unknown parameter.

System Action: Processing of the command is ended. The system status remains the same.

User Response: Issue the command again making sure that all parameters are specified correctly. To see the command's syntax, issue the command without any parameters. For additional information, see the OS/390

SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1002e Required parameter '*parameter*' was not provided.

Explanation: The indicated parameter, which is required by the command, was not specified.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again specifying the required parameter. To see the command's syntax, issue the command without any parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1003e Incorrect syntax for '*parameter*' parameter.

Explanation: The command was issued with incorrect syntax for the indicated parameter.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again using the correct syntax. To see the commands syntax, issue the command without any parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1004e '*parameter*' parameter is not appropriate in this context.

Explanation: The parameter specified is syntactically valid but conflicts with another parameter or the current state of the system. For example, on the **fwadapter** command, the specified IP address is syntactically correct but does not represent an active IP address on the system.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct use of the parameter stated and issue the command again. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1006e Firewall cannot obtain enough memory.

Explanation: The firewall program was unable to obtain the memory needed for its operation.

System Action: Processing of the command or program ended. The system status remains the same.

User Response: Increase the amount of available dynamic storage and reissue the command.

ICAC1007e Command error.

Explanation: During the processing of the command, an internal error was encountered.

System Action: Processing of the command or program ended. The system status remains the same.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1008e Command '*command*' is not recognized.

Explanation: The value specified for the **cmd** parameter is not valid.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with a valid **cmd** value. For a list of valid **cmd** values, issue the command without any parameters. For additional information about the command, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1009e *filename* error: access denied or creation failed.

Explanation: The command encountered an error when attempting to access or create the specified file or the file associated with the file parameter.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify that the user has permission to the file specified, or the directory in which the file will be created. Then issue the command again.

ICAC1010e *filename* in use by another field or application.

Explanation: The file specified is already in use.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the state of the file and issue the command again.

ICAC1011e No output control block was provided.

Explanation: During the processing of the command, an internal error was encountered with the output control block structure. The output control block was not specified.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1012e No input control block was provided.

Explanation: During the processing of the command, an internal error was encountered with the input control block structure. The input control block was not specified.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1013e A request to obtain information from TCP/IP failed - return code *return-code***.**

Explanation: The firewall program requested information from TCP/IP and received an error return code. The return code represents the **errno** value.

System Action: Processing of the command ends. The system status remains the same.

User Response: Correct the error using the returned **errno** value and issue the command again.

ICAC1014e File '*filename***' is required but could not be found.**

Explanation: The file specified does not exist. The most likely reason is that the firewall default configuration was erased. Restore the file from a current backup.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify that the configuration file does not exist. The configuration program expects this file to exist. If a backup version is not available contact your service representative.

ICAC1019i Usage:

Explanation: A user has issued a command that requested help for a specific command. A usage statement for the command follows this header.

System Action: The system displays the usage statement and returns to normal processing.

User Response: None.

ICAC1042e *time-value* **value must be greater than or equal to zero.**

Explanation: The value specified is not within the valid range.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again specifying a value greater than or equal to 0.

ICAC1043e Tunnel *tunnel_id* **already exists.**

Explanation: The specified tunnel ID already exists.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the **fwtunnl** command again with a unique tunnel ID or issue the **fwtunnl cmd=change** command to modify attributes of the existing tunnel. To display the current contents of the tunnel, issue **fwtunnl cmd=list tunnel=***tunnel_id*.

ICAC1044e Tunnel *tunnel_id* **does not exist.**

Explanation: The specified tunnel ID does not exist.

System Action: Processing of the command ends. The system status remains the same.

User Response: Ensure you have specified the correct tunnel ID and issue the command again. To list the currently defined tunnels issue the **fwtunnl cmd=list** command.

ICAC1045e The two configuration files for *tunnel_id* **tunnel are not consistent.**

Explanation: The policy and context configuration files for the specified tunnel are not consistent. The files have become unusable for the specified tunnel.

System Action: Processing of the command ends. The system status remains the same.

User Response: The configuration files will need to be corrected. Delete the specified tunnel and recreate it. If the problem continues, contact the service representative.

ICAC1046e Tunnel *tunnel_id* **is active.**

Explanation: A request was received to activate the specified tunnel and the tunnel is already active.

System Action: Processing of the command ends. The system status remains the same.

User Response: None

ICAC1047e Tunnel *tunnel_id* **is not active.**

Explanation: The specified tunnel ID does not represent an active tunnel. For a list of active tunnels, issue **fwtunnl cmd=list**.

System Action: Command completes. The system status remains the same.

User Response: None

ICAC1048e Failure on system call: *command_string*.

Explanation: The requested configuration command attempted to call another command through a system call and encountered an error.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify sufficient virtual storage is available and issue the command again. If the problem continues, contact the service representative.

ICAC1049e Configuration file *filename* **is corrupted.**

Explanation: The indicated configuration file is not in a usable format. The contents are corrupt. The most likely reason for the corruption is that the file was manually edited and incorrect data added.

System Action: Processing of the command ends. The system status remains the same.

User Response: The configuration file will need to be recreated correctly. First display the contents the file (or make a viewable copy) then erase the original file. Reconfigure the file by using the appropriate firewall configuration command using the original file for reference, if necessary.

ICAC1050e Input value for '*value***' is not valid.**

Explanation: The inputted value was specified incorrectly.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct syntax and issue the command again. To see the command's syntax, issue the command without any parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1051e Optional parameter '*parameter***' is missing.**

Explanation: The command was issued in such a manner that the indicated optional parameter must be specified.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct syntax and issue the command again. To see the command's syntax, issue the command without any parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1053e Configuration file *filename* **is empty.**

Explanation: The indicated configuration file was either not found or the file was found, but it is empty. The most likely reason for the file not being found is that the configuration was not done for the issued command.

System Action: Command processing ends. The system status remains the same.

User Response: Verify the state of the configuration file. If the file exists, the configuration command expects this file to contain data. See the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835 for additional information.

ICAC1055e No input parameter is provided.

Explanation: A command was issued with a parameter that caused an internal error to be detected with the known list of valid keywords.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1056e Modification of file *filename* **failed.**

Explanation: An attempt to modify the indicated file failed. The most likely reason for the failure is an out-of-storage condition.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify that the file system that contains the indicated file is not full. If it is full, clean up the file system or move that data to a larger file system. Issue the command again. If the problem continues, contact the service representative.

ICAC1063e A connection definition is not valid and was not included in the configuration

Explanation: A connection definition was not valid because it contained an empty service list. The definition is skipped and is not included in the configuration. To be valid, a connection definition must contain at least one valid service rule ID value.

System Action: Processing of the command continues.

User Response: Issue the **fwconns cmd=list format=long** command to display all connection definitions. Examine the list output for any definitions with empty service list fields. Issue either **fwconns cmd=change** to update the service list field or **fwconns cmd=delete** to remove the connection definition. Issue the command again.

ICAC1064e Definition of connection *connection_name* contains errors and was not included in the configuration.

Explanation: While updating the filter and socks rules, it was detected that the indicated connection was not valid because it contained no defined service IDs. The connection definition is skipped and is not included in the configuration.

System Action: Processing of the command continues.

User Response: Issue the **fwconns cmd=list format=long name=***connection_name* command to display the connection definition. Verify the service definitions associated with the connection are valid by issuing **fwservice cmd=list**. If the connection is to be included in the configuration, update the definition accordingly and issue the command again.

ICAC1065e Definition of connection *connection_name* contains errors. One or more components was not included in the configuration.

Explanation: While updating the filter and socks rules, it was detected that the indicated connection definition or security policy was not valid. The connection definition or security policy is skipped and is not included in the configuration.

System Action: Processing of the command continues.

User Response: If the message indicates an incorrect connection definition, issue the **fwconns cmd=list format=long name=***connection_name* command to display the connection definition. Verify the service definitions associated with the connection are valid by issuing the **fwservice cmd=list** command. Update the connection definition accordingly and issue the command again. If the message indicates a security policy that is not valid, verify that the firewall predefined policies exist. Issue the **fwservice cmd=list id=n** command where n=20,21,23,34,25 and 26. If any of the above policies are not found, contact the service representative.

ICAC1066e Definition of service *service_name* in connection *connection_name* contains errors and was not included in the configuration.

Explanation: While updating the filter and socks rules, it was detected that the indicated service and connection definitions were not valid because the service list did not contain any valid rule ID values. The connection definition is skipped and is not included in the configuration.

System Action: Processing of the command continues.

User Response: Issue the **fwservice cmd=list format=long name=***service_name* command to display the service definition. Verify that is it valid. Update it accordingly and issue the command again.

ICAC1067e Errors in the security policy. It was not included in the configuration.

Explanation: While updating the filter and socks rules, it was detected that the active firewall security policy file did not contain any valid policy rules. The file has become unusable.

System Action: Processing of the command continues.

User Response: Reconfigure the defined security policies using the **fwsecpolicy** command.

ICAC1070e Operation could not obtain message.

Explanation: An internal error occurred when attempting to retrieve a message from the message catalog. The message was not found.

System Action: Command continues.

User Response: Verify that the firewall message catalogs were not modified. Issue the command again. If the problem continues, contact the service representative.

ICAC1083e Type parameter cannot change.

Explanation: The type field of a Network Address Translation (NAT) entry cannot be change.

System Action: Processing of the command ends. The system status remains the same.

User Response: To change the **type** field, the NAT entry must be deleted and recreated with a new type value.

ICAC1123e Multiple occurrences of '*parameter*' are not valid.

Explanation: The specified parameter may only be used once on this command.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct syntax and issue the command again. To see the command's syntax, issue the command without any parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1124e '*parameter*' must be in 'key=value' format.

Explanation: Command parameters must be specified in the 'key=value' format.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct syntax and issue the command again. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1125e '*parameter*' contains a blank before the equal sign.

Explanation: Command parameters cannot have blanks between the keyword and the equal sign.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct syntax and issue the command again. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1126e '*parameter*' does not contain a command keyword.

Explanation: The indicated parameter does not contain a command keyword. All command parameters must be entered in 'key=value' format.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct syntax and issue the command again. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1127e Maximum number of command parameters exceeded.

Explanation: The command was issued with too many parameters.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct syntax and issue the command again. To see the command's syntax, issue the command without any parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1131e Cannot create temporary file.

Explanation: An attempt to create a temporary file failed.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify space is available in the file system containing **/tmp** or the file system containing the directory specified by the TMPDIR environment variable. Issue the command again. If the problem continues, contact the service representative.

ICAC1143e A request to obtain or use a socket failed. *errno-string*

Explanation: A request to obtain a socket or use an existing socket failed. The reason for the failure is in the message.

System Action: Processing of the command ends. The system status remains the same.

User Response: Correct the reason for the failure and issue the command again. If the problem continues, contact the service representative.

ICAC1156e An error was encountered activating the filter configuration. See *filename* for more information.

Explanation: While updating the filter rules, an error was encountered. The indicated output file contains more information about the failure.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again after correcting the error indicated in the specified file.

ICAC1158e Object specified is not found.

Explanation: The specified object name or ID value does not exist.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again specifying a valid name or ID value.

ICAC1159e Default value cannot be changed or deleted.

Explanation: An attempt was made to delete a predefined rule or object. Predefined objects and rules with an id value of 500 or less cannot be changed or deleted.

System Action: Processing of the command ends. The system status remains the same.

User Response: None

ICAC1160e Object already exists.

Explanation: An attempt to create an object or rule failed because an object or rule with the specified name already exists. Names must be unique for single and group network objects. A group network object cannot be named the same as a single network object nor can a single network object be named the same as a group network object.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with a name not already assigned to an object or rule.

ICAC1161e Object is not valid.

Explanation: An attempt to move a definition before another definition in the configuration file failed because either the source or target definition does not exist.

System Action: Processing of the command ends. The system status remains the same.

User Response: Ensure you have specified the correct definition values and issue the command again.

ICAC1167e Unable to rename file *filename* **to** *filename***.**

Explanation: The firewall program was unable to rename the specified file.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1168e Unable to remove file *filename***.**

Explanation: The firewall program was unable to rename the specified file.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1170e DNS was not yet configured using the firewall configuration tool.

Explanation: List data about the firewall DNS was requested but the DNS was not configured.

System Action: Command completed.

User Response: Configure the bind DNS using the firewall DNS configuration command (**fwdns**). For additional information on the **fwdns** command, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1171e *parameter* **parameter cannot be retrieved from the** *filename***.**

Explanation: Configuration errors were found when attempting to display the DNS list information,

System Action: Processing of the command ends. No list data is returned.

User Response: If you have previously configured DNS, you may want to make backup copies of the following files: **/etc/fwnamed.cat.save**, **/etc/fwnamed.loc.save** and **/etc/fwresol.save** before trying to configure the DNS again. If you issue the DNS configuration command before backing up these files, your previous configuration will be lost.

ICAC1172i Authentication succeeded.

Explanation: The user has specified a valid userid and password and is authorized to perform firewall administration using the configuration GUI (Graghical User Interface).

System Action: User login is complete.

User Response: Proceed to use the GUI for firewall administration tasks.

ICAC1173e Authentication failed.

Explanation: The user has specified an incorrect userid/password combination. In order to use the Configuration GUI (Graphical User Interface), you must specify a userid and password that is known to Firewall's External Security Manager.

System Action: The user is not authorized to use the Configuration GUI.

User Response: Enter a valid userid and password on the logon screen.

ICAC1182e Parameter length is too long.

Explanation: The data specified for either the *name* or *desc* parameter was greater than 100 characters.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with a name and description value not greater than 100 characters in length.

ICAC1221e Import directory *directory_name* **does not contain tunnel information.**

Explanation: The directory specified on the **fwtunnl cmd=import** command was either empty or did not contain the policy and context files from the tunnel partner.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the directory and files in the directory are correct. If not, receive the tunnel partner export files again. Issue the command again.

ICAC1222e Export directory *directory_name* **already contains tunnel information.**

Explanation: The directory specified contains at least one tunnel export file. The **fwtunnl cmd=export** only works against a directory that does not contain any tunnel export files.

System Action: Processing of the command ends. The system status remains the same.

User Response: Erase the tunnel export files from the directory and issue the command again or issue the command with a different directory.

ICAC1223e Number of parameters exceeded the limit of the input control block.

Explanation: An internal error was encountered when building the input control block.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1229e Unable to copy file *from_file* **to** *to_file***.**

Explanation: The **fwmigrate** command encountered an error when attempting to copy the 'from_file' to the 'to_file'.

System Action: Command continues.

User Response: Verify that the HFS file system containing the 'from_file' is mounted read/write and that the file system is not full. Issue the command again.

ICAC1244r Are you sure the above record is the correct one to process?

Explanation: A request to delete or change a logging or archiving attribute has been received by the firewall. Before the action is completed, the attribute entry being modified is displayed for verification.

System Action: Command continues.

User Response: None. You will be prompted with another reply message.

ICAC1245r Please type: 1=Confirm or 2=Cancel.

Explanation: A command was issued which requires a user action response. Enter a '1' to confirm that the action should be processed or '2' if you want to cancel the request.

System Action: The command either continues or ends depending on the message reply.

User Response: Enter either '1' or '2' for desired action.

ICAC1246i Command cancelled by the user, nothing was done.

Explanation: The user requested that the command be canceled.

System Action: Command ended by user. The system status remains the same.

User Response: None.

ICAC1247e The value for parameter *parameter* **cannot be null.**

Explanation: A parameter is specified without a value on the command.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command with a value for the parameter. Make sure the parameter is specified in the 'key=value' format. To see the command's syntax, issue the command without any parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1298e The item you requested to delete is in use by the following group or groups: *group_name(s)***.**

Explanation: An attempt was made to delete a network object that belongs to a group or several groups. The groups are listed in the message content.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again specifying the **force=yes** option to delete the object and to remove it from the list of groups displayed in the message.

ICAC1299e The item you requested to delete is in use by the following service or services: *service_name(s)***.**

Explanation: An attempt was made to delete a filter rule or sock rule that is a member of a service or group of services.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again specifying the **force=yes** option to delete the rule and to remove it from the list of services displayed in the message.

ICAC1300e The item you requested to delete is in use by the following connection or connections: *connection_name(s)***.**

Explanation: An attempt was made to delete a network object that is a member of a connection or group of connections.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again specifying the **force=yes** option to delete the object and to remove it from the list of connections displayed in the message.

ICAC1327e The list of IpAddresses in the *IP_address* **parameter contains a duplicate IpAddress.**

Explanation: An attempt to change the IP addresses for the firewall Domain Name Server failed because duplicate IP addresses being specified in the parameter.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again specifying a unique list of IP addresses.

ICAC1330w The file access permissions for *filename* **cannot be set properly.**

Explanation: An attempt to set the appropriate permissions on the indicated file failed. The reason for the failure appears after the message.

System Action: Processing of the command continues.

User Response: Correct the error and issue the command again to ensure proper permissions for the configuration files.

ICAC1337e There is no activated tunnel. Shutdown will not be performed

Explanation: A request was received to deactivate all active tunnels on the firewall but no active tunnels exist.

System Action: Processing of the command ends. The system status remains the same.

User Response: None

ICAC1340e A VPN with the same Target Address and Target SDI is already active.

Explanation: A request to activate a virtual private network tunnel failed because it is already active.

System Action: Processing of the command ends. The system status remains the same.

User Response: None

ICAC1341e A VPN with that Tunnel ID already exists.

Explanation: An attempt to import a tunnel definition for the specified tunnel ID failed because one already exists for that tunnel ID.

System Action: Processing of the command ends. The system status remains the same.

User Response: Ensure the specified tunnel ID was correct and issue the command again.

ICAC1423e File system does not have enough free blocks for the file *filename***.**

Explanation: An attempt to create or change an object or rule failed because the file system does not have enough space for the data to be added to the specified file.

System Action: Processing of the command ends. The system status remains the same.

User Response: Clean up or increase the size of the file system containing the specified file.

ICAC1426e Source object for connection *connection_name* **cannot be found.**

Explanation: The indicated connection definition is not valid because it does not contain a source network object. The connection definition is being skipped and will not be included in the firewall configuration.

System Action: Processing of the command continues.

User Response: Issue the **fwconns cmd=list format=long name=***connection_name* command to display the connection definition. Correct the connection definition and issue the command again.

ICAC1427e Source object for connection *connection_name* **is not valid.**

Explanation: The indicated connection definition is not valid because it contains a source network object that does not exist. The connection definition is being skipped and will not be included in the firewall configuration.

System Action: Processing of the command continues.

User Response: Issue the **fwconns cmd=list format=long name=***connection_name* command to display the connection definition. Verify that the source object exists, correct the connection and issue the command again.

ICAC1428e Destination object for connection *connection_name* could not be found.

Explanation: The indicated connection definition is not valid because it does not contain a destination network object. A destination object is required for connections that contain service rules. The connection definition is being skipped and will not be included in the firewall configuration.

System Action: Processing of the command continues.

User Response: Issue the **fwconns cmd=list format=long name=***connection_name* command to display the connection definition. Verify that the destination object exists. Correct the connection and issue the command again.

ICAC1429e Destination object for connection *connection_name* is not valid.

Explanation: The indicated connection definition is not valid because it contains a destination network object that does not exist. The connection definition is being skipped and will not be included in the firewall configuration.

System Action: Processing of the command continues.

User Response: Issue the **fwconns cmd=list format=long name=***connection_name* command to display the connection definition. Verify that the destination object exists. Correct the connection and issue the command again.

ICAC1431e The item you requested to delete is in use by the following *object_type*: *object_name*.

Explanation: An attempt to delete a tunnel ID failed because the tunnel is currently in use by the specified filter or service rule.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove or update the indicated rule or service so that it does not reference the tunnel you are deleting, then issue the command again.

ICAC1432e Unknown return code *return_code*.

Explanation: An internal error was found. The indicated return code is not known.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1433e Archive file name does not contain a valid suffix (.a).

Explanation: The specified archive file name on the **fwarchive** command is not valid. The file name must be fully qualified and end with a .a suffix. For example, **/tmp/local4.info.a** represents a valid file name.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with a valid archive file name.

ICAC1434e Entry already exists.

Explanation: The rule being created already exists.

System Action: Processing of the command ends. The system status remains the same.

User Response: None

ICAC1445e The parameter '*parameter*' or '*parameter*' is required.

Explanation: One of the two parameters specified above is required.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with appropriate parameters. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1446e The configuration file '*filename*' does not exist.

Explanation: The indicated file was not found.

System Action: Processing of the command ends. The system status remains the same.

User Response: For additional information see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1447e The configuration map file '*filename*' does not exist.

Explanation: The indicated file was not found.

System Action: Processing of the command ends. The system status remains the same.

User Response: Contact the service representative.

ICAC1448e The parameter '*parameter*' is not supported.

Explanation: The specified parameter is not supported in this version.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct use of the parameter stated and issue the command again. For additional information, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1455e '*string1*' and '*string2*' are mutually exclusive.
You may specify either one, but not both.

Explanation: Mutually exclusive items were found.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1456e Unable to add a nonexistent single member to a group:

Explanation: The indicated object must exist and be a single network object in order to be added to a group.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1457e Please specify either '*this*' or '*that*', but not both.

Explanation: You may specify **this** or **that**, but not both.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1458i Duplicate member was found and ignored:

Explanation: The indicated member(s) were found to be duplicates. They are being ignored.

System Action: Command continues.

User Response: None.

ICAC1459e You cannot delete a group object using the fwnwobj command.

Explanation: You issued **fwnwobj cmd=delete**, but specified an object which was a group object. Group objects cannot be deleted using **fwnwobj**.

System Action: Command ends.

User Response: Issue **fwnwgrp cmd=delete** or pick the correct **fwnwobj** object to delete and issue the **fwnwobj cmd=delete** command again.

ICAC1460e You cannot delete a single object using the fwnwgrp command.

Explanation: You issued **fwnwgrp cmd=delete**, but specified an object which was a single object. Single objects cannot be deleted using **fwnwgrp**.

System Action: Command ends.

User Response: Issue **fwnwobj cmd=delete** or pick the correct **fwnwgrp** object to delete and issue the **fwnwgrp cmd=delete** command again.

ICAC1461e You cannot change a group object using the fwnwobj command.

Explanation: You issued **fwnwobj cmd=change**, but specified an object which was a group object. Group objects cannot be changed using **fwnwobj**.

System Action: Command ends.

User Response: Issue **fwnwgrp cmd=change** or pick the correct **fwnwobj** object to change and issue the **fwnwobj cmd=change** command again.

ICAC1462e You cannot change a single object using the fwnwgrp command.

Explanation: You issued **fwnwgrp cmd=change**, but specified an object which was a single object. Single objects cannot be changed using **fwnwgrp**.

System Action: Command ends.

User Response: Issue **fwnwobj cmd=change** or pick the correct **fwnwgrp** object to change and issue the **fwnwgrp cmd=change** command again.

ICAC1463e You cannot add or remove a single object using the fwnwgrp command.

Explanation: You issued **fwnwgrp cmd=add or remove**, but specified an object which was a single object. Single objects cannot be updated using **fwnwgrp**.

System Action: Command ends.

User Response: Pick the correct **fwnwgrp** object to add or remove and issue the **fwnwgrp cmd=add or remove** command again.

ICAC1464e You cannot change a user type into a non-user type, nor can you change a non-user type into a user type.

Explanation: You cannot change between type=user and type != user. If you must do something like this, just delete the user or non-user and create the non-user or user with the given name.

System Action: Command ends.

User Response: Issue another command if needed.

ICAC1465e This change command will not result in a change.

Explanation: A **change** command was issued without enough parameters to result in a change.

System Action: Command ends.

User Response: Issue the command again with enough parameters to effect a change.

ICAC1466e This add or remove command will not result in a change.

Explanation: An **add** or **remove** command was issued without enough other parameters to actually change anything.

System Action: Command ends.

User Response: Issue the command again with enough parameters to effect a change.

ICAC1467e Timeout value must be a number between 0 through 65535.

Explanation: The value for the **timeout** parameter was not valid.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1468i Object is not a member of the group, object ignored:

Explanation: The indicated object is not a member of the group. The object is ignored and the command continues.

System Action: Command continues.

User Response: None.

ICAC1470e Network Object not found:

Explanation: The indicated network object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1471e Filter Rule not found:

Explanation: The indicated filter rule was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1472e Socks Rule not found:

Explanation: The indicated socks rule was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1473e Service Object not found:

Explanation: The indicated service object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1474e Connection Object not found:

Explanation: The indicated connection object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1477e The security policy configuration file data is not valid.

Explanation: The security policy configuration file data is not valid.

System Action: Command completes.

User Response: Issue the **fwsecpolicy cmd=change** command. Then issue the **fwsecpolicy cmd=list** function to verify your data. If the problem continues, contact the service representative.

ICAC1478e The response data was not valid.

Explanation: The response to the prompts when issuing the **fwsecpolicy cmd=change** command was incorrect. For more information on the syntax and use of the **fwsecpolicy** command, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the **fwsecpolicy cmd=change** command again with the correct response values.

ICAC1479e The security policy configuration cannot be erased.

Explanation: The security policy configuration file cannot be erased. For more information on the syntax and use of the **fwsecpolicy** command, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the **fwsecpolicy** command again. If the problem continues, contact the service representative.

ICAC1480i Duplicate rule was found and ignored:

Explanation: The indicated rule was found to be a duplicate. The rule is being ignored.

System Action: Command continues.

User Response: None.

ICAC1481i Rule was not found and is being ignored:

Explanation: The indicated rule was not found in the list of rules. The rule is ignored and the command continues.

System Action: Command continues.

User Response: None.

ICAC1482i Duplicate object was found and ignored:

Explanation: The indicated object was found to be a duplicate. The object is being ignored.

System Action: Command continues.

ICAC1483i Service was not found and is being ignored:

Explanation: The indicated service was not found in the list of services. The service is ignored.

System Action: Command continues.

User Response: None.

ICAC1484i Socks rule was not found and is being ignored:

Explanation: The indicated socks rule was not found in the list of socks rules. The socks rule is ignored.

System Action: Command continues.

User Response: None.

ICAC1494i This move command will not result in any change.

Explanation: A **move** command was issued which would not result in any change in the list. The moved object and the "to" object were the same.

System Action: Command ends.

ICAC1496e The value for NAME is not valid.

Explanation: The value for NAME is not valid. It must be 1 to 100 characters.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1497e The value for DESC is not valid.

Explanation: The value for DESC is not valid. It must be 1 to 100 characters.

System Action: Command ends.

User Response: Issue the command with the correct syntax.

ICAC1503e Rule '*rulestring*' was not formatted correctly.

Explanation: The rule (or rulelist) was not formatted correctly. See the syntax for the correct format.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1504e Specify both '*this*' and '*that*' together.

Explanation: If you specify **this** or **that**, you must specify them both together.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1505e Specify at least one of the following parameters
required for the 'timefilter' parameter: 'time', 'month', and/or 'weekday'.

Explanation: The **timefilter** parameter requires more information such as time, month, and/or weekday.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1507e Unable to parse network object information from the configuration file.

Explanation: The network object configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1509e Unable to parse services from the configuration file.

Explanation: The services configuration file was corrupted and can no longer be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1510e Unable to parse connections from the configuration file.

Explanation: The connections configuration file was corrupted and can no longer be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1511e Unable to parse socks rules from the configuration file.

Explanation: The socks rules configuration file has been corrupted and can no longer be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1512e The '*parameter1***' and '***parameter2***' values are incompatible.**

Explanation: The values that were specified for *parameter1* and *parameter2* are incompatible.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1513e The value for '*parameter1***' was changed to '0'. When the '***parameter2***' value is 'any', the '***parameter1***' value must be '0'.**

Explanation: If the value for *parameter2* is 'any', then the value of *parameter1* must be '0'. The value for *parameter1* was changed to '0'.

System Action: Command completes.

User Response: None.

ICAC1514i The values for '*parameter1***' and '***parameter2***' are blank, so the values for both parameters will be set to the default values.**

Explanation: **Parameter1** and **parameter2** were specified with blank values. The values for both parameters will be set to the default values.

System Action: Command completes.

User Response: None.

ICAC1515e The NAT configuration file cannot contain more than *max_number_of_entries* **entries.**
The entry was not added to the NAT configuration file.

Explanation: The NAT configuration file contains the maximum number of entries allowed.

System Action: Command ends.

User Response: Use the fwnat cmd=delete command to delete an entry in the NAT configuration file. Then issue the fwnat cmd=add command again.

ICAC1516e *adapter_address* **interface not present.**

Explanation: The **fwadapter** command was issued to display the status of an adapter address and the adapter was not found or it was a loopback adapter.

System Action: The command ends and the system status remains the same.

User Response: Issue the command again with a valid adapter address or issue the command without the **'addr'** parameter.

ICAC1517e The values for 'destopcode' and 'destport' must be 'any' and '0' when the value for 'protocol' is 'ospf'.

Explanation: When the **protocol** is OSPF, the **destopcode** value must be ANY and the **destport** value must be 0.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1518e Destination is required for connections with services.

Explanation: When specifying connections with services, the destination object is required.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1519e Blanks are not allowed in the value for '*parameter***'.**

Explanation: Blanks are not allowed in the value for the parameter specified.

System Action: Command ends.

User Response: Issue the command again without blanks in the parameter value.

ICAC1520e An OS/390 API returned this error number: *errno*.

Explanation: The firewall command was not able to rename an internal file. The cause of the failure appears in the message text.

System Action: Command ends.

User Response: Correct the error indicated in the message text. If the problem continues, contact your service representative.

ICAC1521w The set of SPI values in the range of 1 through 255 are reserved to the Internet Assigned Numbers Authority (IANA) for future use. Confirm that you want to use this SPI value.

Explanation: A request to use a SPI of 1 through 255 was received. This SPI may collide with a SPI used by the Internet.

System Action: Command continues.

User Response: None. User will be prompted with another reply message.

ICAC1522e Vertical bars are not allowed in the value for '*parameter***'.**

Explanation: The vertical bar cannot be specified in the value of the parameter.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1523e The statvfs() request to obtain information about the file system failed : '*error_message***'**

Explanation: The **statvfs()** function was unable to obtain information about the file system status. The error message will provide additional information.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1525e The setuid routine failed when switching to superuser (uid=0). The errno text string is:

Explanation: The setuid(0) failed. See the errno text string for more details, it is displayed after the message following the colon.

System Action: Command ends.

User Response: Run the command again with the correct authorization.

ICAC1526e Unknown host name specified. *errnostring***.**

Explanation: A call to **gethostbyname()** failed when attempting to verify the specified host name. The reason for the failure is in the message. The most likely reason for the failure is that the specified host name is not known to the name server.

System Action: Processing of the command ends. The system status remains the same.

User Response: Correct the error and issue the command again.

ICAC1527e Predefined object cannot be changed or deleted.

Explanation: Objects that are predefined in the configuration files cannot be changed or deleted. Predefined objects have id values less than or equal to 500.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1528e Group specified is not found.

Explanation: The group specified does not exist.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1529e Unable to identify Firewall stack.

Explanation: An attempt to inform TCP/IP of the firewall stack name failed. See subsequent message(s) for more details.

System Action: Processing of the command ends. The system status remains the same.

User Response: Correct the error and issue the command again.

ICAC1530e Connections must have at least one service or socks rule.

Explanation: A **fwconns** command was issued which would result in a connection definition without at least one service or one socks rule included. A connection must have at least one rule included.

System Action: Command ends.

User Response: Issue the command and make sure the connection contains at least one rule.

ICAC1531w Unable to inform the sock daemon to refresh configuration data.

Explanation: The issuer of the **fwfilter** command to modify the **sockd** configuration file was unable to inform the sock daemon that the configuration data needs to be read again. The most likely reason for the error was FWKERN or SOCKD was not started and in a state to receive communication.

System Action: Processing of the command continues.

User Response: Verify that FWKERN and SOCKD were started. If not, start them.

ICAC1532e A day range is only valid when a month range has also been specified.

Explanation: The day value may only be expressed as a range when the month value has also been specified as a range.

System Action: Command terminates.

User Response: Issue the command again with the correct syntax.

ICAC1533e The source parameter and the destination parameter cannot refer to the same network object or group, unless it is the pre-defined object: The World (id=1).

Explanation: The source and the destination parameters must refer to different network objects or groups. The exception is the pre-defined object **The World** (id=1).

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1535e The value '*value***' for parameter '***parameter***' must be a positive integer.**

Explanation: A value was specified for a parameter which was not a positive integer. The value must be a positive integer.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1536e Group object '*name_or_id***' cannot be added as a member of a group.**

Explanation: Group objects can only contain single objects. You can not add groups to groups.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1537e The value '*value***' for parameter '***parameter***' is not valid.**

Explanation: A value was specified for a parameter which was not a positive integer. The value must be a positive integer.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1538e Please specify '*this***' and/or '***that***'.**

Explanation: You may specify **this** or **that**, or both.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1539e The Shutdown service interface(id=26) and Test IP Routing(id=25) values are mutually exclusive. They cannot both be specified together in servicelist.

Explanation: In the servicelist parm, Shutdown service interface cannot be specified with Test IP Routing.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1540e The parameter 'encrypthow' is required unless policy=auth.

Explanation: Policies with encryption require the 'encrypthow' information.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1541e The parameter 'algorithm' is required unless policy=encr.

Explanation: Non 'encr' policies require the 'algorithm' information.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1543r 'fwdns' is currently using *filename* **as the OE resolver. Are you sure that you want to change the OE resolver to** *filename***?**

Explanation: A request was received to change the OE resolver used by the Firewall. Before Action is taken, the file name of the current OE resolver is displayed for verification.

System Action: The command continues.

User Response: User will be prompted with another reply message.

ICAC1544e *filename* **is not a fully qualified HFS file name.**

Explanation: The file name specified for the OE resolver must be a fully qualified HFS file name. The fwdns command does not support TSO partitioned or sequential data sets.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again, specifying a fully qualified HFS file name as the OE resolver or use the default OE resolver '/etc/security/fwstacks/<stack>/fwresolv.conf' another system.

ICAC1545w The specified OE resolver file *filename* **has a different stack name than the one specified in the stack parameter.**

Explanation: The stack name in the specified OE resolver file is different from the stack name specified in the **STACK** parameter.

System Action: Processing of the command continues.

User Response: Verify that the specified OE resolver file should be used with the Firewall. The specified resolver was changed. The changes effect the TCPIPJOBNAME, domain, and nameserver statements. If the specified resolver already contained any TCPIPJOBNAME, domain, and/or nameserver statements, the original statements have been commented out.

ICAC1545e Daemon *daemon_name* **cannot be started due to it not being configured.**

Explanation: The **fwdaemon** command was issued to start a daemon that was not configured.

System Action: Processing of the command ends. The system status remains the same.

User Response: Do the following, then try the request again:

- Ensure that the command syntax is correct.
- Ensure that the specified daemon was the daemon to be started.
- If the specified daemon was the daemon to be started, use the **fwdaemon** command to configure the daemon.

ICAC1546e Parameter 'maxconns' is valid only when 'daemon' is either SOCKD or PFTPD.

Explanation: The **fwdaemon** command was issued to change the configuration 'maxconns' value. The 'maxconns' value can only be modified for the SOCKD and PFTPD deamons.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the correct daemon was specified and issue the command again.

ICAC1547e Changes requested to daemon *daemon_name* **will result in the configuration entry being greater than 255 bytes.**

Explanation: The **fwdaemon** command was issued to change the configuration attributes of the specified daemon. The changes will result in the configuration entry being greater than 255 bytes in length.

System Action: Processing of the command ends. The system status remains the same.

User Response: Verify the parameter values and issue the command again.

ICAC1549i Daemon *daemon_name* **is not configured.**

Explanation: The **fwdaemon** command was issued to query the state of the specified daemon, but the daemon was not configured.

System Action: Command completes.

User Response: None

ICAC1551e One or more errors were found in the input parameters.

Explanation: One or more errors were found in the input parameters. See the specific error message for the parameter in error for more information.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1552e The firewall TCP/IP stack '*stackname***' was not valid or not active.**

Explanation: The stack was not a valid firewall stack or it was not active. The **setibmopt** call or the **ioctl** call failed.

System Action: Command ends.

User Response: Issue the command again with the correct stack or use the **force=yes** option.

ICAC1553e The firewall TCP/IP stack '*stackname***' already exists.**

Explanation: The stack being added already exists.

System Action: Command ends.

User Response: Issue the command again with the correct stack.

ICAC1554e Unable to make directory '*directory***'.**

Explanation: A **mkdir** command was issued to make the specified directory. This command was unable to make the directory. Perhaps the syntax for the directory was in error or the directory was read only or the permissions prevented the command from working.

System Action: Command ends.

User Response: Fix the problem and issue the command again.

ICAC1555e The firewall TCP/IP stack '*stackname***' is active.**

Explanation: The stack was active. Use the **force=yes** option to delete this stack anyway.

System Action: Command ends.

User Response: Issue the command again with **force=yes** or bring down the TCP/IP firewall stack first, then issue the command again.

ICAC1556e The firewall TCP/IP stack '*stackname***' was not found.**

Explanation: The stack specified on the stack parameter was not found in the list of stacks defined by the **fwstack** command.

System Action: Command ends.

User Response: Issue the command again with the correct stack.

ICAC1557e Unable to rename directory '*old directory***' to '***new directory***'.**

Explanation: A **mv** command was issued to rename the specified directorys. This command failed, perhaps the syntax for a directory was in error or the new directory was read only or the permissions prevented the command from working.

System Action: Command ends.

User Response: Fix the problem and issue the command again.

ICAC1558e Parameter 'timefilter' is required when any of the following parameters are specified: 'time', 'month', 'day', or 'weekday'.

Explanation: The **timefilter** parameter is required when time, month, day, or weekday is specified.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1559e Parameter '*parameter1***' is required when parameter '***parameter2***' is specified.**

Explanation: The parameter **parameter1** is required when parameter **parameter2** is specified on the command.

System Action: Command ends.

User Response: Issue the command again with the correct syntax.

ICAC1560e No active firewall TCP/IP stacks were found.

Explanation: None of the stacks defined from the **fwstack** command were active.

System Action: Command ends.

User Response: Start the firewall TCP/IP stack(s) and issue the command again.

ICAC1561e Tunnel *tunnel_id* not found.

Explanation: The specified tunnel ID does not represent a defined tunnel. For a list of defined tunnels, enter **fwtunnl cmd=list**.

System Action: Processing of the command is stopped. The system status remains the same.

User Response: Enter the command again with known tunnel IDs.

ICAC1562e File modification for tunnel *tunnel_id* failed.

Explanation: When updating the tunnel definition and policy files, an error was found for the specified tunnel.

System Action: Processing of the command is stopped.

User Response: Enter the command again with the specified tunnel ID. If the problem continues, contact the service representative.

ICAC1563e No tunnel definitions exist.

Explanation: A request to activate all defined tunnels was received. Currently, no tunnel definitions exist.

System Action: Processing of the command is stopped.

User Response: None

ICAC1564e Both algorithm parameters must be set to KEYED_MD5.

Explanation: When creating a tunnel and KEYED_MD5 authentication algorithm is used, both the source and destination authentication algorithms must be set to KEYED_MD5.

System Action: Processing ends.

User Response: Enter the command again with the correct authentication algorithm values.

ICAC1565e Tunnel definition requires that new header be 'yes'.

Explanation: When creating a tunnel, the new header field is required when the tunnel definition consists of one or more of the following attributes:

- Source authentication algorithm is not KEYED_MD5
- Destination authentication algorithm is not KEYED_MD5
- Source encryption algorithm is 3DES_CBC or ESP_NULL
- Destination encryption algorithm is 3DES_CBC or ESP_NULL
- Replay is requested
- Source ESP authentication is requested
- Destination ESP authentication is requested

System Action: Processing of the command is stopped.

User Response: Enter the command again specifying the new header value set to 'yes'.

ICAC1566e New header format not allowed when AH algorithm is KEYED_MD5.

Explanation: When creating a tunnel and KEYED_MD5 authentication algorithm is used, both source policy and destination authentication algorithms must be set to KEYED_MD5.

System Action: Processing of the command is stopped.

User Response: Enter the command again with new header value set to 'no'.

ICAC1567e ESP_NULL encryption type requires ESP authentication algorithm to be specified.

Explanation: When specifying a ESP_NULL encryption type, its corresponding source or destination ESP authentication algorithm must be specified.

System Action: Processing of the command is stopped.

User Response: Enter the command again with the appropriate ESP authentication algorithm.

ICAC1568e 'spi' and 'destESPspi' are mutually exclusive. You must specify one, but not both.

Explanation: When creating a tunnel the destination ESP spi is required. Specifying the 'spi' value is done through the 'spi' or 'destESPspi' parameters. When changing a tunnel's destination ESP spi, 'spi' or 'destESPspi' can be used. These parameters represent the same value and cannot both be specified.

System Action: Processing of the command is stopped.

User Response: Enter the command again specifying either 'spi' or 'destESPspi'.

ICAC1569e Required parameter 'spi' or 'destESPspi' was not provided.

Explanation: When creating a tunnel the destination ESP spi is required. Specifying the spi value is done through the 'spi' or 'destESPspi' parameters.

System Action: Processing of the command is stopped.

User Response: Enter the command again specifying either 'spi' or 'destESPspi'.

ICAC1570e Existing key not valid for new *parameter_name* **value.**

Explanation: When changing an algorithm in an existing tunnel definition, it was detected that the current key associated with the algorithm being changed is not appropriate for the new algorithm. For example, if the source authentication algorithm is changed from HMAC_MD5 to HMAC_SHA, the authentication key size for the new algorithm is different than the key size for the old algorithm.

System Action: Processing of the command is stopped.

User Response: Enter the command specifying a new key for the algorithm being changed. To have a system generated key assigned, enter the parameter without any value.

ICAC1571e Algorithm values must be both KEYED_MD5 or not KEYED_MD5.

Explanation: When changing an authentication algorithm, KEYED_MD5 requires both source and destination authentication algorithms to be KEYED_MD5. If one of the algorithms is not KEYED_MD5, neither of the algorithms can be KEYED_MD5.

System Action: Processing of the command is stopped.

User Response: Enter the command specifying both algorithms as KEYED_MD5 or not KEYED_MD5.

ICAC1572e DNS has not been configured yet for firewall TCP/IP stack '*stackname***'.**

Explanation: List data about the firewall DNS was requested but the DNS was not configured for the specified stack.

System Action: Command completed.

User Response: Configure the bind DNS using the firewall DNS configuration command (**fwdns**). For additional information on the **fwdns** command, see the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1573e 'destencrypthow' parameter value required unless destination policy is 'auth'.

Explanation: When specifying the destencrypthow parameter on the change command, policies with encryption require encryption information.

System Action: Processing of the command is stopped.

User Response: Enter the command again specifying a destination encryption algorithm.

ICAC1574e 'destalgorithm' parameter value required unless destination policy is 'encr'.

Explanation: When specifying the destalgorithm parameter on the change command, policies with authentication require authentication information.

System Action: Processing of the command is stopped.

User Response: Enter the command again specifying a destination authentication algorithm.

ICAC1575e Export failed for tunnel *tunnel_id***. Tunnel definition contains information not supported in previous versions.**

Explanation: An attempt to export the specified tunnel failed because the tunnel definition contains information that is only valid for tunnels using the new header format or the tunnel definition had a timeout of 0.

System Action: Processing of the command is stopped.

User Response: Verify the specified tunnel IDs are correct and issue the command again.

ICAC1576e KEYED_MD5 is not compatible with the value specified in *field_descriptor***.**

Explanation: An attempt to create a new tunnel definition failed because the value specified is not allowed when the authentication algorithms are KEYED_MD5. KEYED_MD5 requires old header while the specified parameter or its value requires new header.

System Action: Processing of the command is stopped.

User Response: None

ICAC1577i Processing firewall TCP/IP stack *stackname*:

Explanation: The stack is being processed. This is an informational message showing which stack (usually in a loop through all the stacks) is being processed. This message can be combined with other messages that are associated with the specified stack.

System Action: Processing continues, this is an informational message.

User Response: None

ICAC1578i Stack *stackname* is not active, it is being skipped.

Explanation: The stack was not active. This stack is being skipped.

System Action: Processing continues, this is an informational message.

User Response: None

ICAC1579e NAT rules for this stack conflict with stack *stackname*.

Explanation: NAT rules for the specified stack conflict with NAT rules for another stack. Rules for one stack cannot conflict with those of another stack.

System Action: Command ends.

User Response: Issue the command again with non conflicting data.

ICAC1580e Unable to deactivate tunnels for stack *stackname*.

Explanation: **FWKERN** requested tunnels belonging to the specified stack to be reactivated. The reactivation failed. During cleanup of from the failure, the update of the tunnel active file to mark the tunnels as not active failed.

System Action: Command ends.

User Response: Verify the state of the active tunnels by issuing the **fwtunnl cmd=list** command. For any tunnels that should be active that appear to be in the inactive state, issue the **fwtunnl cmd=activate tunnel=tunnel_list** to activate them.

ICAC1581i FIREWALL MIGRATION STARTED.

Explanation: The firewall migration has started.

System Action: Command continues.

User Response: None

ICAC1582i FIREWALL MIGRATION COMPLETED SUCCESSFULLY.

Explanation: The **fwmigrate** firewall migration completed successfully.

System Action: Command continues.

User Response: None

ICAC1583i Initial firewall stack migration started.

Explanation: The migration of the initial firewall stack has started.

System Action: Command continues.

User Response: None

ICAC1584i Initial firewall stack migration completed.

Explanation: The **fwmigrate** migration of the initial firewall stack completed successfully.

System Action: Command continues.

User Response: None

ICAC1585i Tunnel migration started.

Explanation: The migration of the tunnel definitions has started.

System Action: Command continues.

User Response: None

ICAC1586i Tunnel migration complete.

Explanation: The **fwmigrate** migration of the existing tunnel definitions completed successfully.

System Action: Command continues.

User Response: None

ICAC1587i NAT migration started.

Explanation: The migration of the Network Address Translation files has started.

System Action: Command continues.

User Response: None

ICAC1588i NAT migration completed.

Explanation: The **fwmigrate** migration of the Network Address Translation files completed successfully.

System Action: Command continues.

User Response: None

ICAC1589i DNS migration started.

Explanation: The migration of the Domain Name Server files has started.

System Action: Command continues.

User Response: None

ICAC1590i DNS resolver file *filename* **moved to** *directory*.

Explanation: The **fwmigrate** migration of the Domain Name Server files resulting in moving the DNS resolver file. You may need to update an JCL/TSO procs to reflect the new location of the DNS resolver file.

System Action: Command continues.

User Response: None

ICAC1591i DNS migration completed.

Explanation: The **fwmigrate** migration of the Domain Name Server files completed successfully.

System Action: Command continues.

User Response: None

ICAC1592i Filter migration started.

Explanation: The migration of the filter rules file has started.

System Action: Command continues.

User Response: None

ICAC1593i Filter migration completed.

Explanation: The **fwmigrate** migration of the filter rules file completed successfully.

System Action: Command continues.

User Response: None

ICAC1594e Tunnel configuration files are corrupt.

Explanation: The **fwmigrate** attempted to migrate the tunnel definitions an encountered an inconsistency between the tunnel definition files.

System Action: Command ends.

User Response: Issue the **fwmigrate** command again. If the problem continues, contact the service representative.

ICAC1595e Tunnel migration failed, old files restored.

Explanation: The **fwmigrate** command encountered an error while migrating the tunnel definitions to their new format. The tunnel definitions have been restored to their original values. A message identifying the failure will also be displayed.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1596e Tunnel *tunnel_id* **is active. Cannot be changed or deleted.**

Explanation: An active tunnel cannot be changed or deleted.

System Action: Command ends.

User Response: Deactivate the tunnel and issue the command again.

ICAC1597e *Parameter_name* **requires an ESP authentication algorithm to be specified.**

Explanation: The indicated ESP authentication key parameter was specified without its corresponding ESP authentication algorithm.

System Action: Command ends.

User Response: Issue the command again specifying an ESP authentication algorithm.

ICAC1598e FIREWALL MIGRATION FAILED, PLEASE CORRECT THE PROBLEMS!

Explanation: The **fwmigrate** command encountered an error while migrating a part of firewall files. See preceding messages for more details.

System Action: Processing of the command is stopped.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1599e Initial firewall stack migration failed, old files restored.

Explanation: The **fwmigrate** command encountered an error while migrating the initial firewall stack. During migration, the initial stack name is gotten from the **/etc/security/fwstack.name** file. The fwstack.name file is not used after migration starting with OS/390 Version 2 Release 7 Firewall.

System Action: Processing of the command is stopped.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1600i Removing firewall .active status files started.

Explanation: The migration to remove .active status files has started.

System Action: Command continues.

User Response: None

ICAC1601i Removing firewall .active status files completed.

Explanation: The **fwmigrate** migration to remove the .active status files completed successfully.

System Action: Command continues.

User Response: None

ICAC1602e NAT migration failed, old files restored.

Explanation: The **fwmigrate** command encountered an error while migrating the Network Address Translation files.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1603e Filter migration failed, old files restored.

Explanation: The **fwmigrate** command encountered an error while migrating the filter rules file.
SystemAction : Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1604e DNS migration failed, old files restored.

Explanation: The **fwmigrate** command encountered an error while migrating the Domain Name Server files.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1605e Unable to find keyword(s): '*keyword(s)***' in file '***filename***'.**

Explanation: The GUI passed in keywords from the Command Viewer part of the main panel, but the keywords were not found in the fwguicmds.$LANG file. Perhaps the file is missing, corrupt, or access permissions were not correct.

System Action: GUI shows error and continues. API ends.

User Response: Fix the problem and issue the GUI command again. You may need to logoff and log back on to the GUI to pick up the command keyword list.

ICAC1606e Start time '*start hhmm***' must not be after stop time '***stop hhmm***'.**

Explanation: The TIME=hhmm-hhmm parameter was specified with a start time which was later than the stop time. Start time hhmm must be <= stop time hhmm.

System Action: Command ends.

User Response: Issue the command again with the correct time range.

ICAC1607e The STACK parameter is required when more than one Firewall TCP/IP stack is defined.

Explanation: When more than one stack is defined using the fwstack command, then the current command issued requires the stack parameter to select which stack to configure.

System Action: Command ends.

User Response: Reissue the command with the stack parameter.

ICAC1608e Resolver file '*filename***' conflicts with resolver file for Firewall TCP/IP stack '***stackname***'.**

Explanation: The resolver file name used for this stack is the same as the resolver file name for another stack. These names must be different.

System Action: Command ends.

User Response: Reissue the command with a different stack or resolver parameter.

ICAC1609e Request is not valid

Explanation: An internal error occurred when attempting to process a request from the configuration client. Request is not valid.

System Action: Command continues.

User Response: Verify that the firewall configuration server and configuration client are at the same service level. If they are not, apply the missing service. Issue the command again. If the problem continues, contact the service representative.

ICAC1610e Tunnel configuration files have become corrupted.

Explanation: The tunnel configuration files are not in a usable format. The contents are corrupt.

System Action: Command ends.

User Response: Contact the service representative.

ICAC1611e Import tunnel configuration files for tunnel *tunnel_id* contains incorrect tunnel data or non-supported function.

Explanation: The tunnel configuration files for the specified tunnel contains non-usable data or function that is not supported.

System Action: Command ends.

User Response: Verify the configuration files were successfully transferred from the tunnel owner machine to the tunnel partner machine. Verify that the tunnel definition consists of function that is currently supported.

ICAC1612e Tunnel activation for tunnel *tunnel_id* failed. IP address not associated with an active firewall TCP/IP stack.

Explanation: A request was issued to activate a tunnel which contained an IP address that was not associated with any active firewall TCP/IP stacks.

System Action: Command ends.

User Response: Start the TCP/IP stack associated with the tunnel's IP address and issue the request again.

ICAC1613i Predefined configuration migration started.

Explanation: The migration of predefined configuration has started.

System Action: Command continues.

User Response: None

ICAC1614e Predefined configuration migration failed, old file(s) restored.

Explanation: The **fwmigrate** command encountered an error while migrating the predefined configuration.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1615i Predefined configuration migration completed.

Explanation: The **fwmigrate** migration of the predefined configuration completed successfully.

System Action: Command continues.

User Response: None

ICAC1616e DNS has already been configured for firewall TCP/IP stack '*stackname*'.

Explanation: DNS configuration for the given stack has already been provided.

System Action: Command ends.

User Response: Select another new stack or modify the existing DNS configuration for that stack.

ICAC1617e The Firewall TCP/IP stack '*stackname*' is not active.

Explanation: The stack is not active. The action will take place the next time the stack comes back up.

System Action: Command ends.

User Response: Bring up the TCP/IP stack.

ICAC1619e Unable to parse key transform information from the configuration file.

Explanation: The key transform configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1621e Unable to parse key proposal information from the configuration file.

Explanation: The key proposal configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1623e Unable to parse key policy information from the configuration file.

Explanation: The key policy configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1625e Unable to parse ah transform information from the configuration file.

Explanation: The ah transform configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1627e Unable to parse esp transform information from the configuration file.

Explanation: The esp transform configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1628e Ah Transform Object not found:

Explanation: The specified Ah Transform object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1629e Esp Transform Object not found:

Explanation: The specified Esp Transform object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1631e Unable to parse data proposal information from the configuration file.

Explanation: The data proposal configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1633e Unable to parse data policy information from the configuration file.

Explanation: The data policy configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1634e Dynamic tunnel not found:

Explanation: An attempt to create or change an anchor filter definition failed because the specified dynamic tunnel was not found.

System Action: Command ends.

User Response: Issue the command again with a valid dynamic tunnel definition.

ICAC1635e Filter rules that contain an anchor type cannot be changed to a type of permit or deny.

Explanation: An attempt to change an anchor filter definition to a type other than anchor is not allowed.

System Action: Command ends.

User Response: To change the type value, the definition must first be deleted and then created again.

ICAC1636e Filter rules that contain a permit or deny type cannot be changed to a type of anchor.

Explanation: An attempt to change a permit or deny filter definition to a type of anchor is not allowed.

System Action: Command ends.

User Response: To change the type value, the definition must first be deleted and then created again.

ICAC1637e '*Parameter*' **is not supported for anchor filter rules.**

Explanation: An attempt to change an anchor filter definition failed because an unsupported parameter was specified.

System Action: Command ends.

User Response: Issue the command again with valid parameters.

ICAC1639e Unable to parse authentication information from the configuration file.

Explanation: The authentication information configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1640e Remote Key Server Object not found:

Explanation: The specified Remote Key Server object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1641e Maximum size would be exceeded, action canceled.

Explanation: The requested action will cause the object to exceed the allowed object size limit. The action was canceled to prevent storage overlay.

System Action: Command ends.

User Response: Create new objects or groups as

needed, or ensure that the object size is within the object limits.

ICAC1642e You cannot define more than *number* Firewall TCP/IP stacks.

Explanation: Only **number** of Firewall TCP/IP stacks can be defined using the **fwstack** command.

System Action: Command ends.

User Response: Delete or rename an existing stack definition.

ICAC1643e Key Transform Object not found:

Explanation: The specified Key Transform object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1644e Key Proposal Object not found:

Explanation: The specified Key Proposal object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1645e Data Proposal Object not found:

Explanation: The specified Data Proposal object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1646e Unable to parse key server group information from the configuration file.

Explanation: The key server group configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1647e Unable to parse dynamic connection information from the configuration file.

Explanation: The dynamic connection configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1649e Unable to parse key server information from the configuration file.

Explanation: The key server configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1650e The item you specified to delete is in use by the following key proposal(s): *key_proposoal_name(s)*.

Explanation: An attempt was made to delete an item that is a member of a key proposal. Only orphan items may be deleted.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the container first, then remove the items it contained.

ICAC1651e The item you specified to delete is in use by the following key policies: *key_policy_name(s)*.

Explanation: An attempt was made to delete an item that is a member of a key policy. Only orphan items may be deleted.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the container first, then remove the items it contained.

ICAC1652e The item you specified to delete is in use by the following data proposal(s): *data_proposal_name(s)*.

Explanation: An attempt was made to delete an item that is a member of a data proposal. Only orphan items may be deleted.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the container first, then remove the items it contained.

ICAC1653e The item you specified to delete is in use by the following key server group(s): *key_server_group_name(s)*.

Explanation: An attempt was made to delete an item that is a member of a key server group. Only orphan items may be deleted.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the container first, then remove the items it contained.

ICAC1654e The item you specified to delete is in use by the following dynamic tunnel(s): *dynamic_tunnel_name(s)***.**

Explanation: An attempt was made to delete an item that is a member of a dynamic tunnel. Only orphan items may be deleted.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the container first, then remove the items it contained.

ICAC1656e The item you specified to delete is in use by the following dynamic connection(s): *dynamic_connection_name(s)***.**

Explanation: An attempt was made to delete an item that is a member of a dynamic connection. Only orphan items may be deleted.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the container first, then remove the items it contained.

ICAC1655e The item you specified to delete is in use by the following data policies: *data_policy_name(s)***.**

Explanation: An attempt was made to delete an item that is a member of a data policy. Only orphan items may be deleted.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the container first, then remove the items it contained.

ICAC1660e Key Policy Object not found:

Explanation: The specified Key Policy object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1661e Key Server Object not found:

Explanation: The specified Key Server object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1662e Key Server Group Object not found:

Explanation: The specified Key Server Group object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1663e Preshared key is not valid. The key must contain an even number of hexadecimal digits.

Explanation: The preshared key that was specified is not valid. A valid preshared key must contain an even number of hexadecimal digits. A hexadecimal digit must be one of the following characters:
0-9, a-f, A-F

System Action: Command ends.

User Response: Issue the command again with a valid key.

ICAC1664e Key server '*keysrv***' is already associated with authentication information '***authinfo***'.**

Explanation: Only one authentication information object can point to a given key server.

System Action: Command ends.

User Response: Issue the command again with a different key server.

ICAC1666e Unable to parse dynamic tunnel information from the configuration file.

Explanation: The dynamic tunnel configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1667e Data Policy Object not found:

Explanation: The specified Data Policy object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1669e The item you specified to delete is in use by the following filter rules(s): *filter_rule_name(s)***.**

Explanation: An attempt was made to delete a dynamic tunnel that is referenced by a filter rule. Only orphan dynamic tunnels may be deleted.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Remove the reference from each listed filter rule, then delete the dynamic tunnel.

ICAC1670e The key server group does not contain the remote key server.

Explanation: The key server group must contain the remote key server.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Try the command again using the correct information.

ICAC1671e Unable to find the first key server group containing the specified key server:

Explanation: A key server group that contains the key server was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1672e You cannot remove the last item from *list name***.**

Explanation: An empty list is not valid, so the last item cannot be removed.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Issue the command again with the correct list of items to delete.

ICAC1673e You cannot remove the last item from *changed list name***, because removing this item**
results in an empty *changed list name* **and an empty** *unchanged list name*

Explanation: Both lists cannot be empty, so the last item cannot be removed.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Issue the command again with the correct list of items to delete.

ICAC1674e Unable to store *filename* **because of insufficient disk space.**

Explanation: The file system in which the file belongs does not have enough space to store the file's new version.

System Action: Processing of the command ends. The system status remains the same.

User Response: Increase the disk space allocated to the file system, then issue the command again.

ICAC1675e Dynamic Connection not found:

Explanation: The specified Dynamic Connection was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1676e Authentication Information object not found:

Explanation: The specified Authentication Information object was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1677e '*parameter name***' and '***parameter name***' are mutually exclusive.**
This change causes both to be defined.

Explanation: This change results in both mutually exclusive parameters being set.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1678e Combination for ADDR/MASK or STARTADDR/ENDADDR is not valid.

Explanation: Network objects contain (ADDR and MASK) or (STARTADDR and ENDADDR). Combining of these parameters is not valid for the add command and not valid for the change command when combined with the existing information. To switch between these types of network objects on the change command, null out the old parameter values that cause the combination to be incorrect.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Try the command again using the correct information.

ICAC1679e Range not valid: value for '*parm1***' must be less than or equal to value for '***parm2***'.**

Explanation: Range must be parm1 <= parm2.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Try the command again using the correct information.

ICAC1681e Unable to parse certificate information from the configuration file.

Explanation: The certificate configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1682e Certificate not found:

Explanation: The specified Certificate was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1683e IP address for tunnel(s) *tunnel id* **not associated with an active Firewall TCP/IP stack.**

Explanation: The specified tunnels do not contain an IP address that belongs to one of the active Firewall TCP/IP stacks.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Try the command again using a tunnel whose IP address belongs to an active stack.

ICAC1684e *Ah_or_Esp* **Transform Object '***Object***' is not in list for specified data proposal.**

Explanation: The specified transform was not found in the list for the specified data proposal.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1685e The value for *parameter name* **is not valid because of the value specified for** *parameter name***.**

Explanation: The first parameter is validated based on the value specified for the second parameter. The value specified for either the first or second parameter is not valid.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Try the command again using valid values for both parameters.

ICAC1686e The value for *keyword name* **is not valid (***RDN value***).**

Explanation: The value specified for an RDN keyword must be within the maximum length allowed. If it is preceded with a double quote, it must be followed by another double quote. The last value specified may not be followed by a comma.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Try the command again using a valid value.

ICAC1687e The RDN *keyword name* **is out of order.**

Explanation: The RDNs within an X.500 distinguished name must be in the order specified in RFC 1845.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Try the command again using the correct order.

ICAC1688e '*unknown keyword***' is an unknown RDN keyword.**

Explanation: The specified RDN keyword is unknown. Perhaps blanks were placed between the comma delimited items or a typographical error was found.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Try the command again using a valid RDN keyword name.

ICAC1689e The value (*value***) for RDN keyword** *keyword* **must be surrounded by double quote characters.**

Explanation: The specified value must be surrounded by double quote characters because it contains leading, trailing, or consecutive blanks, or it contains at least one of the following characters: ',', '+', '=', '"', '\n', '\r', '<', '>', '#', or ';'.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Try the command again using the correct value for the keyword.

ICAC1690e Key Transform Object '*Object*' is not in list for specified key proposal.

Explanation: The specified key transform was not found in the list for the specified key proposal.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1691e Data Proposal Object '*Object*' is not in list for specified data policy.

Explanation: The specified data proposal was not found in the list for the specified data policy.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1692e Key Server Object '*Object*' is not in list for specified key server group.

Explanation: The specified key server was not found in the list for the specified key server group.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1693i ISAKMP is processing the command.

Explanation: ISAKMP has received the request and processing of the command is underway.

System Action: Processing continues.

User Response: If the command involved dynamic connection management (activate, deactivate, refresh or shutdown), issue the **fwdynconns cmd=listactive** to verify the active connection information.

ICAC1694e Connection name not known to ISAKMP.

Explanation: ISAKMP was not able to retrieve information for the connection due to the specified connection name not being found.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Issue the dynamic connection listactive request (**fwdynconns cmd=listactive**) to receive a list of valid dynamic connection names. Issue the **fwdynconns** specifying a connection name from the list.

ICAC1695e ISAKMP did not accept the user request.

Explanation: ISAKMP did not accept the user request.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Verify that the ISAKMP server has been started and is in READY state. If server is down, start the server. Issue the request again.

ICAC1696i ISAKMP has accepted the request but processing has not started.

Explanation: ISAKMP has accepted the request but processing has not been started due to either the current workload on the server or the server has been stopped.

System Action: Processing of the command continues.

User Response: If the server has been stopped, restart the server and if activating a connection, issue the request again. If the server is in READY state, wait a little while and query the state of the dynamic connections (**fwdynconns cmd=listactive**).

ICAC1697e Unexpected error received from ISAKMP.

Explanation: During the processing of the command, an internal error was encountered. See the syslog records for more details on the error.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1698e Error received when sending user request to ISAKMP.

Explanation: During the processing of the command, an internal error was encountered.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Verify that fwkern has been started and issue the command again. If the problem continues, contact the service representative.

ICAC1699e Error received when building dynamic connection request for ISAKMP.

Explanation: During the processing of the command, an inconsistency was detected in the dynamic connection configuration.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Validate the current dynamic connection configuration and issue the command again.

ICAC1700e ISAKMP encountered an error processing the request. See syslog records or fwtrace records for more details.

Explanation: During the processing of the command, an error was encountered. See the syslog records or the fwtrace records for more details on the error.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Fix the cause of the failure indicated in either the syslog or fwtrace records and issue the command again.

ICAC1701w No tunnel definitions associated with the active Firewall TCP/IP stack(s).

Explanation: An attempt to activate all defined manual tunnel definitions resulted in no additional tunnels being activated. The current tunnel definitions contain IP addresses that are not associated with any of the Firewall TCP/IP stacks currently active.

System Action: Processing of the command ends.

User Response: Verify the state of the Firewall TCP/IP stacks. Restart any stacks that should be running and issue the command again.

ICAC1703e Tunnel *tunnel_id* **contains an unsupported encryption algorithm.**

Explanation: An attempt to activate the specified manual tunnel failed. The tunnel definition contains an encryption algorithm that is not supported by the Firewall TCP/IP stack.

System Action: Processing of the command continues.

User Response: Verify that the encryptions algorithms in the tunnels are correct according to the installation support by TCP/IP.

ICAC1704e Internal error occurred when activating tunnel *tunnel_id***.**

Explanation: An attempt to activate the specified manual tunnel failed. An internal error was found by the Firewall TCP/IP stack.

System Action: Processing of the command continues.

User Response: Contact the service representative.

ICAC1705e Activation of tunnel *tunnel_id* **failed because of an out of memory condition in the Firewall TCP/IP stack.**

Explanation: An attempt to activate the specified manual tunnel failed due to an out of shortage condition found by the Firewall TCP/IP stack.

System Action: Processing of the command continues.

User Response: Try the request again.

ICAC1706e Activation of tunnel *tunnel_id* **failed because of an unexpected error. RetCode=***return_code***.**

Explanation: An attempt to activate the specified manual tunnel failed because of an unexpected error.

System Action: Processing of the command continues.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1707e Activation for tunnel(s) *tunnel_ids* **failed. Error on socket open for stack (***stackname***).**

Explanation: An attempt to activate the specified manual tunnels failed because of an error opening a socket to the specified TCP/IP stack.

System Action: Processing of the command continues.

User Response: Issue the request again specifying the failing tunnel numbers.

ICAC1708i Manual tunnel activation not complete.

Explanation: An attempt to activate manual tunnels did not complete in its entirety. See previous messages to explanations on tunnels that were not activated.

System Action: Processing of the command ends.

User Response: Correct the problem stated in the previous messages and issue the request again.

User Response: Issue the command again with the correct move syntax.

ICAC1709e Activation of tunnel *tunnel_id* **failed because of an error in the tunnel definition.** *field_descriptor* **is not valid.**

Explanation: An attempt to activate the specified manual tunnel failed because the definition contains a value that is not valid. The field that contains the value that is not valid is specified in the error message. The most likely cause for the failure is because an imported

tunnel contains data that is not valid for the OS/390 implementation of manual tunnels.

System Action: Processing of the command continues.

User Response: Correct the contents of the tunnel definition and reactivate the tunnel. If the tunnel definition was imported, notify the tunnel owner to correct the tunnel definition and correct or import the tunnel definition again before activating the tunnel.

ICAC1710e Activation of tunnel *tunnel_id* failed. 3DES_CBC and ESP_NULL requires new header format.

Explanation: An attempt to activate the specified manual tunnel failed because the definition contains parameter values that are not compatible. An encryption type of 3DES_CBC or ESP_NULL requires new header. The mostly likely reason for the failure is because an imported tunnel contains data that is not valid for the OS/390 implementation of manual tunnels.

System Action: Processing of the command continues.

User Response: Correct the contents of the tunnel definition and reactivate the tunnel. If the tunnel definition was imported, notify the tunnel owner to correct the tunnel definition and correct or import the tunnel definition again before activating the tunnel.

ICAC1711e Activation of tunnel *tunnel_id* failed. Authentication new header combination not allowed.

Explanation: An attempt to activate the specified manual tunnel failed because the definition contains parameter values that are not compatible. Authentication type of KEYED_MD5 cannot be specified with new header format. Authentication types HMAC_MD5 and HMAC_SHA require new header format. The most likely reason for the failure is because an imported tunnel containing data that is not valid for the OS/390 implementation of manual tunnels.

System Action: Processing of the command continues.

User Response: Correct the contents of the tunnel definition and reactivate the tunnel. If the tunnel definition was imported, notify the tunnel owner to correct the tunnel definition and correct or import the tunnel definition again before activating the tunnel.

ICAC1712e Activation of tunnel *tunnel_id* failed. ESP_NULL encryption requires an ESP authentication algorithm.

Explanation: An attempt to activate the specified manual tunnel failed because the definition containing parameter values that are not compatible. An encryption type of ESP_NULL requires its corresponding source or destination ESP authentication algorithm to be specified. The most likely reason for the failure is because an imported tunnel contains data that is not valid for the OS/390 implementation of manual tunnels.

System Action: Processing of the command continues.

User Response: Correct the contents of the tunnel definition and reactivate the tunnel. If the tunnel definition was imported, notify the tunnel owner to correct the tunnel definition and correct or import the tunnel definition again before activating the tunnel.

ICAC1714e Activation of tunnel *tunnel_id* failed. Replay requires new header format.

Explanation: An attempt to activate the specified manual tunnel failed because the definition containing parameter values that are not compatible. Replay requires new header format. The most likely reason for the failure is because an imported tunnel contains data that is not valid for the OS/390 implementation of manual tunnels.

System Action: Processing of the command continues.

User Response: Correct the contents of the tunnel definition and reactivate the tunnel. If the tunnel definition was imported, notify the tunnel owner to correct the tunnel definition and correct or import the tunnel definition again before activating the tunnel.

ICAC1715e More than one key transform found in key proposal.

Explanation: If aggressive mode is specified for the initiator on a key policy, then the initiator key proposal must only specify one key transform. More than one key transform was found in the specified initiator key proposal.

System Action: Processing of the command ends.

User Response: Correct the key proposal.

ICAC1716e ESP Transform '*Object*': Authentication algorithm must be specified if encryption type is ESP_NULL.

Explanation: An encryption type of ESP_NULL requires that an authentication algorithm be specified.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1717e Tunnel *tunnel_id* is not valid or does not exist.

Explanation: The specified manual tunnel does is not a valid tunnel ID name or the tunnel does not exist.

System Action: Command ends.

User Response: Issue the command again with a valid tunnel ID. If the tunnel does not exist, create the tunnel and issue the command again.

ICAC1719i Name migration started.

Explanation: The migration of the configuration files has started. Each configuration file is having its name field adjusted to meet current standards.

System Action: Command continues.

User Response: None

ICAC1720e Name migration failed, old files restored.

Explanation: The **fwmigrate** command encountered an error while migrating the configuration files to their new format. The files were restored to their original state. A message identifying the failure will also be displayed.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1721i Name migration complete.

Explanation: The **fwmigrate** migration of the existing configuration files for name field changes completed successfully.

System Action: Command continues.

User Response: None

ICAC1722i Objects migration started.

Explanation: The migration of the objects file has started.

System Action: Command continues.

User Response: None

ICAC1723e Objects migration failed, old file restored.

Explanation: The **fwmigrate** command encountered an error while migrating the objects file to its new format. The objects file were restored to its original state. A message identifying the failure will be displayed.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1724i Objects migration complete.

Explanation: The **fwmigrate** migration of the existing objects file completed successfully.

System Action: Command continues.

User Response: None

ICAC1726i Configuration file was updated but communication with daemon failed or did not complete.

Explanation: The certificate authority configuration file was updated successfully but either the ISAKMP server encountered and error or did not respond to the request.

System Action: Processing of the command continues.

User Response: Start the ISAKMP server if it is not in the READY state. If ISAKMP server is in the READY state, either delete and add the certificate authority again or recycle the ISAKMP server.

ICAC1728e The specified key ring is not valid.

Explanation: A valid key ring name: contains a slash ('/') character between the userid and key ring name, does not contain an asterisk ('*') character, and is not longer than 237 characters.

System Action: Command ends.

User Response: Issue the command again with a valid key ring name.

ICAC1729e Authentication ID Type cannot be changed unless Authentication ID is also changed.

Explanation: To change the Authentication ID Type, the Authentication ID must also be changed.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1730e Key Server Object '*remkeysrv*' cannot be removed from the specified Key Server Group because both are in use by the following dynamic connections: *dynconnlist*.

Explanation: An attempt was made to remove a key server object from a key server group. Both the key server and the key server group are in use by a dynamic connection, so the key server cannot be removed.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the key server from the dynamic connection, then issue the command again.

ICAC1731e Data proposals must have at least one ah or esp transform.

Explanation: The **fwdataprop** command issued will result in a data proposal definition with no ah or esp transform value. A data proposal must have at least one transform value specified.

System Action: Command ends.

User Response: Issue the command again specifying at least one ah or esp transform.

ICAC1732i Predefined services migration started.

Explanation: The migration of predefined services has started.

System Action: Command continues.

User Response: None

ICAC1733e Predefined services migration failed, old file(s) restored.

Explanation: The **fwmigrate** command encountered an error while migrating the predefined services.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1734i Predefined services migration completed.

Explanation: The **fwmigrate** migration of the predefined services completed successfully.

System Action: Command continues.

User Response: None

ICAC1735i Predefined rules migration started.

Explanation: The migration of predefined filter rules has started.

System Action: Command continues.

User Response: None

ICAC1736e Predefined rules migration failed, old file(s) restored.

Explanation: The **fwmigrate** command encountered an error while migrating the predefined filter rules.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1737i Predefined rules migration completed.

Explanation: The **fwmigrate** migration of the predefined filter rules completed successfully.

System Action: Command continues.

User Response: None

ICAC1738i Daemon configuration migration started.

Explanation: The migration of daemon configuration has started.

System Action: Command continues.

User Response: None

ICAC1739e Daemon configuration migration failed, old file(s) restored.

Explanation: The **fwmigrate** command encountered an error while migrating the daemon configuration.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1740i Daemon configuration migration completed.

Explanation: The **fwmigrate** migration of the daemon configuration completed successfully.

System Action: Command continues.

User Response: None

ICAC1741e Manual processing required for daemon configuration migration.

Explanation: The daemon configuration must be manually migrated.

System Action: Command continues.

User Response: For additional information, refer to OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1742i Manual migration already performed for daemon configuration.

Explanation: The daemon configuration has already been manually migrated.

System Action: Command continues.

User Response: For additional information, refer to OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAC1743e This change would cause Dynamic Connection '*dynconn***' to not be valid. Remote Key Server '***remkeysrv***' must remain in Key Server Group '***keysrvgrp***'.**

Explanation: The change would result in removing a key server from the key server group which is in use by a dynamic connection. The change was not done.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the key server from the dynamic connection, then issue the command again.

ICAC1744i *parm* may have syntax problems. It is missing the '*char*' character.

Explanation: This is just an informational. The value for the specified variable may have a syntax problem. Normally the value would contain the character shown as missing. However in some cases that character may not be needed.

System Action: The command continues.

User Response: Correct the syntax if needed, otherwise no action is required.

ICAC1745i *parm* may have syntax problems. It contains the '*char*' character.

Explanation: This is just an informational. The value for the specified variable may have a syntax problem. Normally the value would not contain the character shown here. However in some cases that character may be needed.

System Action: The command continues.

User Response: Correct the syntax if needed, otherwise no action is required.

ICAC1746e Unable to parse certificate authorities from the configuration file.

Explanation: The certificate configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1747e Certificate Authority not found:

Explanation: The specified Certificate Authority was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1748e The item you specified to delete is in use by the following Authority Information(s): *authority_information_name(s)*.

Explanation: An attempt was made to delete an item that is a member of an authority information. Only orphan items may be deleted.

System Action: Processing of the command ends. The system status remains the same.

User Response: Remove the container first, then remove the items it contained.

ICAC1749e RACF Userid '*userid*' is not valid. It must be from 1 to 8 characters composed of characters A-Z, 0-9, '#', '$', and '@'.

Explanation: The RACF userid was not valid. A valid RACF userid must be from 1 to 8 characters in length and contain any combination of A-Z, 0-9, '#', '$', or '@'.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with the correct syntax.

ICAC1750i For more informational messages, see the syslog records.

Explanation: During the processing of the command, additional information became available. See the syslog records for more informational details.

System Action: Processing of the command continues.

User Response: No action required.

ICAC1751e For more error messages, see the syslog records.

Explanation: During the processing of the command, an error was encountered. See the syslog records for more details on the error.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1752i For more informational messages, see the fwtrace records.

Explanation: During the processing of the command, additional information became available. See the fwtrace records for more informational details.

System Action: Processing of the command continues.

User Response: No action required.

ICAC1753e For more error messages, see the fwtrace records.

Explanation: During the processing of the command, an error was encountered. See the fwtrace records for more details on the error.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1754i For more informational messages, see the syslog or fwtrace records.

Explanation: During the processing of the command, additional information became available. See the syslog records or the fwtrace records for more informational details.

System Action: Processing of the command continues.

User Response: No action required.

ICAC1755e For more error messages, see the syslog or fwtrace records.

Explanation: During the processing of the command, an error was encountered. See the syslog records or the fwtrace records for more details on the error.

System Action: Processing of the command ends. The system status remains unchanged.

User Response: Issue the command again. If the problem continues, contact the service representative.

ICAC1756i Processing *keyword*: *object*

Explanation: This is just an informational message indicating what object is being processed. This can be used to identify which objects contains errors on following error message which multiple objects are processing in a series.

System Action: Command continues.

User Response: None.

ICAC1757i Currently the following Key Servers may be associated with this Authentication Information: <a new one>

Explanation: Each authentication information object must be associated with only one remote key server. This is an informational message showing additional key servers which are currently not associated with another authentication information object. Also remote key servers associated with authentication information objects must have an IP address or a Host Name in its definition.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with the valid remote key server.

ICAC1758e This label already exists in Certificate Authority '*certauth***'. Please use a unique label.**

Explanation: These labels must be unique. The given label was already defined.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with a unique label.

ICAC1759e The Remote Key Server for Authentication Information Objects must have an IP address or a Host Name defined.

Explanation: Key Servers associated with an Authentication Information Object must have an IP address or a Host Name defined.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with a valid remote key server.

ICAC1760e Authentication information objects must have a shared key and/or a certificate authority value.

Explanation: The requested change to the authentication information object will result in an authentication information object that is not valid. Authentication information objects must contain either a shared key or certificate authority or both to be valid.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again specifying a shared key or certificate authority value.

ICAC1761e Certificate authority configuration file updated but label was not found on key ring.

Explanation: The certificate authority configuration file was successfully updated but the ISAKMP server detected a problem with the specified label. The label was not found on the currently defined key ring. If the request was to change the label within the certificate authority entry, ISAKMP no longer knows about the old label.

System Action: Command ends.

User Response: Verify that the specified label exists on the key ring. If label does exist on the key ring, delete and add the certificate authority again. If the label does not exist on the key ring, issue the command again specifying a label from the key ring.

ICAC1762e Protocol '*protocol*' requires source and destination ports to be zero.

Explanation: An attempt to create or change a dynamic connection definition failed due to a conflict between the protocol and port values. A protocol of **all** or **ipip** requires that the values specified in the source and destination port fields to be zero.

System Action: Command ends.

User Response: Issue the command again with valid values.

ICAC1763e Unable to parse ondemand information from the configuration file.

Explanation: The ondemand configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1765e Ondemand definition not found:

Explanation: The specified Ondemand definition was not found.

System Action: Command ends.

User Response: Issue the command again with the correct information.

ICAC1767e The item you specified to delete is in use by the following ondemand(s): *ondemand_name(s)*.

Explanation: An attempt was made to delete an item that is a member of a ondemand definition. Only orphan items may be deleted.

System Action: Processing of the command ends. The system status remains the same

User Response: Remove the container first, then remove the items it contained.

ICAC1768e Ondemand definition not found for connection *connection_name*.

Explanation: An attempt was made to expand and update the active filter rules. While processing the specified connection an error was detected. The ondemand object associated with the connection does not exist. The connection has been skipped.

System Action: Processing of the command continues.

User Response: Update the connection to contain a valid ondemand object and reactivate the filter rules.

ICAC1769e Unable to parse ondemand information for connection *connection_name*.

Explanation: The ondemand configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1770i Connections migration started.

Explanation: The migration of the connection configuration file has started. Each line is being modified for the ondemand value.

System Action: Command continues.

User Response: None

ICAC1771e Connections migration failed, old file restored.

Explanation: The **fwmigrate** command encountered an error while migrating the connection file to its new format. The connection file wasrestored to its original state. A message identifying the failure will be displayed.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1772i Connections migration complete.

Explanation: The **fwmigrate** migration of the existing connections file completed successfully.

System Action: Command continues.

User Response: None

ICAC1773e Key server (*keyserver*) you specified to remove from the key server group is in use by the following ondemand(s): *ondemand_name(s)*.

Explanation: An attempt was made to remove a key server from a key server group definition that was being reference by an ondemand object. The ondemand object requires the specified gateway key server to belong to the key server group.

System Action: Processing of the command ends. The system status remains the same

User Response: Remove or replace the gateway key server value in the ondemand object or remove the ondemand object. Issue the command again.

ICAC1774e This change would cause the On-demand definition '*ondemand*' not to be valid. Key Server '*gatekeysrv*' must remain in Key Server Group '*keysrvgrp*'.

Explanation: The change would result in removing a key server from the key server group which is in use by an on-demand definition. The change was not done. SystemResponse : Processing of the command ends. The system status remains unchanged.

User Response: Replace the gateway key server in the on-demand definition with another key server in the key server group and issue the command again.

ICAC1775e The key server group does not contain the gateway key server.

Explanation: The key server group must contain the gateway key server.

System Action: Processing of the command ends. The system status remains the same.

User Response: Try the command again using the correct information.

ICAC1776e The log file must be an existing fully qualified file name.

Explanation: The specified log file name is not valid. The file name must be fully qualified and represent an existing file.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with a valid log file name.

ICAC1777e The archive file must be a fully qualified file name.

Explanation: The specified archive file name is not valid. The file name must be fully qualified. The most likely reason for the error was that

- specified path was not fully qualified
- a directory within the path does not exist
- the specified name represents a directory.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with a valid archive file name.

ICAC1778e The workspace must be an existing fully qualified directory name.

Explanation: The specified workspace directory is not valid. The directory name must be fully qualified and represent an existing directory.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with a valid workspace name.

ICAC1779e Log file name already exists.

Explanation: The specified log file name already exists in another log archive definition. A log file can only be specified in at most one definition.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with another log file name.

ICAC1780e Archive file name already exists.

Explanation: The specified archive file name already exists in another log archive definition. An archive file can only be specified in at most one definition.

System Action: Processing of the command ends. The system status remains the same.

User Response: Issue the command again with another archive file name.

ICAC1781e Unable to parse archive information from the configuration file.

Explanation: The archive configuration file was corrupted and cannot be parsed.

System Action: Command ends.

User Response: Contact your service representative.

ICAC1782i Syslog configuration migration started.

Explanation: The migration of syslog configuration has started.

System Action: Command continues.

User Response: None

ICAC1783e Syslog configuration migration failed, old file(s) restored.

Explanation: The **fwmigrate** command encountered an error while migrating the syslog configuration.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1784i Syslog configuration migration completed.

Explanation: The **fwmigrate** migration of the syslog configuration completed successfully.

System Action: Command continues.

User Response: None

ICAC1785i Log file archive configuration migration started.

Explanation: The migration of log file archive configuration has started.

System Action: Command continues.

User Response: None

ICAC1786e Log file archive configuration migration failed, old file(s) restored.

Explanation: The **fwmigrate** command encountered an error while migrating the log file archive configuration.

System Action: Command continues.

User Response: Correct the error and issue the **fwmigrate** command again.

ICAC1787i Log file archive configuration migration completed.

Explanation: The **fwmigrate** migration of the log file archive configuration completed successfully.

System Action: Command continues.

User Response: None

ICAM0001e Error opening catalog (*catalog_name***).** *error_text*

Explanation: The specified message catalog cannot be opened. Additional information about the error is given by the C library error text, which is returned from the **catopen()** function.

System Action: The program ends.

ICAM0002w Initialization error calling ***function*****.**

Explanation: An error occurred when calling the specified function.

ICAM0003e User is not authorized to issue *command* **command**

Explanation: A user has attempted to issue a command without the proper authority. The most common error is that the command is a privileged command that must be issued by the Firewall administrator or a super user.

System Action: The command exits.

User Response: Check the authorization of the user issuing the command and issue the command again. If the problem continues and the user has the appropriate authorization, contact the service representative.

ICAM0004e Command syntax is not valid for *command* **command**

Explanation: A user has attempted to issue a command that does not conform to the documented command syntax.

System Action: The command exits.

User Response: Check the syntax of the command and issue the command again with the correct syntax.

If the problem continues, contact the service representative.

ICAM0005i Usage:

Explanation: A user has attempted to issue a command that:

- requested help for a specific command, or
- did not specify the correct syntax for the command. A usage statement for the command will follow this header.

System Action: The system displays the usage statement and returns to normal execution.

User Response: If the syntax of the command was not correct, issue the command with the appropriate syntax. If the problem continues, contact the service representative.

ICAM0006e Operand *operand* **for** *option* **option on** *command* **command is not valid**

Explanation: A command was issued with an option/operand pair specified and the operand was not valid for this option.

System Action: The system displays the usage statement and returns to normal execution.

User Response: Correct the syntax of the command as documented in the command usage statement and issue the command again. If the problem continues, contact the service representative.

ICAM0007i Trace configuration file does not exist, creating configuration file *file_name*

Explanation: An **fwtrace** command was issued and the fwtrace configuration file did not previously exist.

System Action: The fwtrace configuration file will be created and processing of the command will continue.

ICAM0008e File name specified on *operand* **operand of** *command* **command does not match current trace file name**

Explanation: The file name specified on the file operand is not the name of the current fwtrace data file.

System Action: The fwtrace command exits with a failure return code.

User Response: Specify the name of the current trace configuration file, but do not specify the file operand. Or, perform the required actions to specify a new fwtrace data file name and issue the fwtrace command again. If the problem continues, contact the service representative.

ICAM0009e Component name *component_name* **in configuration file is not valid**

Explanation: A component name read from the fwtrace configuration file is not a valid Firewall component.

System Action: The fwtrace command exits with a failure return code.

User Response: The fwtrace configuration file was corrupted. Contact the service representative.

ICAM0010i Status of IBM Firewall for OS/390 trace facility:
Trace output file = *trace_filename*
Component = *component_name* **Status =** *trace_status*

Explanation: An fwtrace query command was issued to check the status of the fwtrace components. The following data will include the current trace data file and the status of each of the Firewall components.

System Action: The fwtrace command displays the tracing status of each of the Firewall components and returns.

ICAM0013e Unable to open trace output file *filename* **-** *error_text*

Explanation: The fwtrace command is not able to open the currently active trace data file.

System Action: The fwtrace command exits with a failure return code.

User Response: Verify that the file name is correct, the file does exist, and the file is accessible in read/write mode by the Firewall administrator. If the problem continues, contact the service representative.

ICAM0014e Unable to open trace configuration file *filename*

Explanation: The fwtrace command is not able to open the trace configuration file.

System Action: The fwtrace command exits with a failure return code.

User Response: Verify that the file does exist and is accessible in read/write mode by the Firewall administrator. If the problem continues, contact the service representative.

ICAM0015e Unable to write to trace configuration file *filename*

Explanation: The fwtrace command is not able to write to the trace configuration file.

System Action: The fwtrace command exits with a failure return code.

User Response: Verify that the file does exist and is accessible in read/write mode by the Firewall administrator. If the problem continues, contact the service representative.

ICAM0019e Insufficient memory for tcpimage buffer

Explanation: Cannot allocate memory for stack buffer.

System Action: Command aborts (system unchanged).

User Response: Check memory usage and issue the command again.

ICAM0020e Cannot open TCP/IP stack name file, *FileName*: *Error*

Explanation: Cannot open or find TCP/IP stack name file.

System Action: Command aborts (system unchanged).

User Response: Check file permissions and issue the command again.

ICAM0021e Cannot link to named TCP/IP stack, *StackName*: *Error*

Explanation: Name listed in TCP/IP stack name file not valid or stack not active.

System Action: Command stops (system unchanged).

User Response: Use the **fwstack** command to verify or define firewall stacks and issue the command again.

ICAM0022i Trace data path does not exist; trace configuration reset

Explanation: The name of the trace data file specified in the trace configuration file does not exist or the path to the file does not exist. The contents of the configuration file are restored to an inactive state with all components deactivated and the trace data file name set to NULL.

System Action: Processing of the command continues.

ICAM0100e Cannot open or read file *FileName*: *Error*

Explanation: Cannot open, read or find the file.

System Action: Command ends (system unchanged).

User Response: Check the file permissions and issue the command again.

ICAM0101e Configuration file '*filename***' is empty.**

Explanation: The specified configuration file was either empty or not found. The most likely reason is that the configuration was not done for the related command.

System Action: Command processing ends. The system status remains the same.

User Response: Verify the state of the configuration file. If the file exists, the configuration command expects this file to contain data. See the OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835 for additional information.

ICAM0102e Search string not found in fwconfig.map: *search string*

Explanation: The search string specified was not found in the fwconfig.map file. This prevented being able to map the configuration files to their full path name.

System Action: Command processing ends. The system status remains the same.

User Response: Verify the state of the fwconfig.map file.

ICAM1000i Firewall daemon *daemon_name* **has stopped.**

Explanation: The specified Firewall daemon has stopped. This message is issued if the daemon is stopped using the STOP command, or if the daemon stops unexpectedly. If the daemon stops unexpectedly, FWKERN attempts to start it again.

System Action: The program continues.

ICAM1001i Firewall daemon *daemon_name* **status is** *status_string* **and process id is** *daemon_pid*.

Explanation: This message is issued when a daemon is queried using the QUERY command. The system returns one of the following status values:

DOWN
The daemon is not running.

STOPPING
FWKERN is attempting to end the daemon.

INITIALIZING
The daemon is running but has not completed initialization.

READY
The daemon is running, initialized, and ready to receive requests.

DISABLED
The daemon cannot be started because the OS/390 Security Server was not been purchased.

UNKNOWN
The daemon has started, but FWKERN cannot determine if the daemon has completed initialization.

System Action: The program continues.

ICAM1002i FWKERN is attempting to start daemon *daemon_name*.

Explanation: FWKERN detects the unexpected ending of the daemon and is attempting to automatically start it.

System Action: FWKERN attempts to automatically start the daemon.

ICAM1003i FWKERN initialization complete.

Explanation: FWKERN successfully completed initialization.

System Action: The program continues.

ICAM1004i FWKERN has received STOP command.

Explanation: FWKERN has received the STOP command.

System Action: FWKERN ends all Firewall daemons and ends.

ICAM1005i FWKERN detects missing parameters for MODIFY command.

Explanation: FWKERN received a **MODIFY** command that was missing required parameters. As the result, FWKERN ignored the command.

System Action: The FWKERN ignores the command and continues.

Operator Response: Ensure that the command syntax is correct, then try the request again.

ICAM1006e MODIFY command syntax is too long.

Explanation: The **MODIFY** command directed to FWKERN contains syntax that is greater than the maximum length permitted.

System Action: FWKERN ignores the command and continues.

Operator Response: Do one of the following, then try the request again:

- Ensure the command syntax is correct.
- Use abbreviations for command parameters to reduce the length of the command syntax.

ICAM1007e FWKERN detects internal error. *parameter_name*: *value*.

Explanation: A call to an internal FWKERN routine specifies a parameter value that is not valid. This error should not occur during normal processing.

System Action: The program continues.

ICAM1008e FWKERN cannot retrieve MODIFY command data.

Explanation: FWKERN encounters a severe error while attempting to retrieve the **MODIFY** command data from the communications area.

System Action: The program continues with reduced function. **MODIFY** commands are no longer processed.

ICAM1009e MODIFY command contains no verb.

Explanation: The required verb parameter is not specified on the **MODIFY** command.

System Action: The request fails.

Operator Response: Ensure that the command syntax is correct, then try the request again.

ICAM1010e FWKERN does not recognize verb *verb_name*.

Explanation: The verb parameter supplied on the **MODIFY** command is not valid.

System Action: The request fails.

Operator Response: Ensure that the command syntax is correct, then try the request again.

ICAM1011e Daemon name parameter required for *command_name* command.

Explanation: The **MODIFY** command does not specify the required daemon name parameter.

System Action: The request fails.

Operator Response: Ensure that the command syntax is correct, then try the request again.

ICAM1012e Daemon name *daemon_name* specified for *command_name* command is not valid.

Explanation: The daemon name parameter supplied on the **MODIFY** command is not valid.

System Action: The request fails.

Operator Response: Ensure that the command syntax is correct, then try the request again.

ICAM1013e Status parameter required for *command_name* command.

Explanation: The **MODIFY** command does not specify the required status parameter.

System Action: The request fails.

Operator Response: Ensure that the command syntax is correct, then try the request again.

ICAM1014e Status *daemon_status*, specified for *command_name* command is not valid.

Explanation: The status parameter specified for the **MODIFY** command is not valid.

System Action: The request fails.

Operator Response: Ensure that the command syntax is correct, then try the request again.

ICAM1015e User *user_ID* is not authorized to issue *command_name* command.

Explanation: The user does not have the required authority to issue the specified command.

System Action: The program continues.

System Programmer Response: Determine if the user should be authorized to use the command. Information about authorizing users for Firewall commands is found in OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAM1016e FWKERN cannot determine user ID of requestor. *error_text*

Explanation: A user has requested an operation that requires authorization. FWKERN is not able to determine the user ID of the requestor to verify if the requestor has the correct authority. Additional information about the error is given by the C library error text. This error should not occur during normal processing.

System Action: The program continues.

System Programmer Response: Correct the specified problem, then try the request again.

ICAM1017e START command cannot start daemon *daemon_name* because it is not configured.

Explanation: The START command cannot start the specified daemon because it is not configured.

System Action: The program continues.

Operator Response: Do the following, then try the request again:

- Ensure that the command syntax is correct.
- Ensure that the daemon specified is the daemon actually intended for starting.

System Programmer Response: If the daemon specified is the daemon actually intended for starting then follow the instructions in OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835 to configure the daemon.

ICAM1018e Daemon *daemon* cannot be started because it has status of *daemon_status*.

Explanation: The daemon has a status other than DOWN. A START request is only honored if the daemon has a status of DOWN. If the status is READY, the daemon is already running.

System Action: The program continues.

Operator Response: Do the following, then try the request again:

- Ensure that the daemon specified is the one actually intended for starting.
- If the status is either INITIALIZING or STOPPING, then wait for the status to change to READY or DOWN.
- If the status is DISABLED, the daemon cannot be started because the OS/390 Security Server was not purchased.
- If the status is UNKNOWN, contact the Firewall administrator.

ICAM1019e Daemon *daemon* cannot be stopped because it has status of *daemon_status*.

Explanation: The daemon has a status other than UNKNOWN or READY. A stop request is only honored if the daemon has a status of UNKNOWN or READY. If the daemon status is DOWN, the daemon is already stopped.

System Action: The program continues.

Operator Response: Do the following, then try the request again:

- Ensure that the daemon specified is the one actually intended for stopping.

- If the status is either INITIALIZING or STOPPING, then wait for the status to change to READY, DOWN, or UNKNOWN.

ICAM1020e Daemon *daemon_name* initialization not complete.

Explanation: FWKERN attempts to start a daemon, but the daemon does not report that initialization is complete.

System Action: FWKERN sets the status of the daemon to UNKNOWN and continues.

Operator Response: Do the following:

- Query the status of the daemon using the QUERY command. If the status shown for the daemon is READY, the initialization completed after the time out which generated this message. If the status is not READY, proceed with the following steps.
- If the daemon has a process ID other than zero, then do the following:
  - Issue a STOP command for the daemon.
  - Wait for a message indicating that the daemon has stopped.
  - Try the START command again.
- If the process id is zero try the START command again.
- Check the console log for any messages or dumps that occur prior to this message, and notify the system programmer.

System Programmer Response: Use the diagnostic data collected by the operator to determine the probable cause of the failure.

ICAM1021e FWKERN cannot start daemon *daemon_name* because the daemon did not remain active through the restart time limit.

Explanation: FWKERN cannot automatically start the daemon because the previous instance of the daemon did not run long enough to demonstrate the stability of the next instance.

System Action: The program continues.

ICAM1022e No entry exists for daemon *daemon_name* in daemon configuration file.

Explanation: The daemon configuration file (/etc/security/fwdaemon.cfg) does not contain an entry for the specified daemon.

System Action: The program continues.

System Programmer Response: Do the following:

- Ensure that the specified entry was not removed from the daemon configuration file.
- Ensure that all daemon names are spelled correctly.

Valid syntax for the daemon configuration file is found in OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAM1023e FWKERN detects unknown daemon name *daemon_name* in daemon configuration file.

Explanation: The daemon configuration file (/etc/security/fwdaemon.cfg) contains a daemon name that is not known to FWKERN.

System Action: The program continues.

System Programmer Response: Do the following:

- Examine the entry in the daemon configuration file that contains the specified daemon name.
- Ensure that all daemon names are spelled correctly.
- Ensure that no other syntax errors exist in the file.

Valid syntax for the daemon configuration file is found in OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAM1024e Entry is missing parameter *parameter_name* in daemon configuration file.

Explanation: FWKERN encounters an entry in the daemon configuration file (/etc/security/fwdaemon.cfg) that is missing the specified parameter.

System Action: The program continues.

System Programmer Response: Do the following:

- Examine the entry in the daemon configuration file and ensure that all entries contain the parameter.
- Ensure that the parameter is spelled correctly and that the value is valid.
- Ensure that no other syntax errors exist in the file.

Valid syntax for the daemon configuration file is found in OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAM1025e FWKERN detects error in value *value_string* in daemon configuration file.

Explanation: FWKERN encounters an error in a parameter in the daemon configuration file (/etc/security/fwdaemon.cfg). Either the parameter or the value specified in the error message is not valid.

System Action: The program continues.

System Programmer Response: Do the following:

- Examine the entry in the daemon configuration file and ensure that the value contains valid characters.

- Ensure that no other syntax errors exist in the file.

Valid syntax for the daemon configuration file is found in OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAM1026e FWKERN detects missing quotation mark in value *parameter_value* **in daemonconfiguration file.**

Explanation: A closed quotation mark is missing from an entry in the daemon configuration file (/etc/security.fwdaemon.cfg).

System Action: The program continues.

System Programmer Response: Examine the entry in the daemon configuration file and ensure that all quotation characters are matched. Valid syntax for the daemon configuration file is found in OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAM1027e FWKERN cannot create child process. System return code: *return_code-reason_code***.**

Explanation: FWKERN fails to create a child process. This error occurs when a START command is processing or when an automatic start attempt is made for a daemon.

System Action: The program continues.

ICAM1028e FWKERN cannot end process *process_id***.** *error_text*

Explanation: FWKERN fails to end a child process. This error occurs when a STOP command is processing. The error indicates that the process ID is not valid. Additional information about the error is given by the C library error text which is returned from the **kill()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1029e FWKERN cannot determine process ID for daemon *daemon_name***.**

Explanation: FWKERN has no record of a process ID for the specified daemon.

System Action: The program continues.

ICAM1030e FWKERN cannot create local socket. *error_text*

Explanation: FWKERN attempts to create a local socket, but the OS/390 UNIX System Services kernel returns an error. Additional information about the error is given by the C library error text, which is returned from the **socket()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1031e FWKERN cannot bind to local socket. *error_text*

Explanation: FWKERN attempts to bind to a local socket, but the OS/390 UNIX System Services kernel returns an error. Additional information about the error is given by the C library error text, which is returned from the **bind()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1032e FWKERN cannot listen on local socket. *error_text*

Explanation: FWKERN attempts to listen on a local socket, but the OS/390 UNIX System Services kernel returns an error. Additional information about the error is given by the C library error text, which is returned from the **listen()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1033e FWKERN cannot accept local socket request. *error_text*

Explanation: FWKERN attempts to accept a request on a local socket, but the OS/390 UNIX System Services kernel returns an error. Additional information about the error is given by the C library error text, which is returned from the **accept()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1034e FWKERN cannot write to local socket. *error_text*

Explanation: FWKERN attempts to write to a local socket, but the OS/390 UNIX System Services kernel returns an error. Additional information about the error is given by the C library error text, which is returned from the **write()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1035e FWKERN cannot read from local socket. *error_text*

Explanation: FWKERN attempts to read from a local socket, but the OS/390 UNIX System Services kernel returns an error. Additional information about the error is given by the C library error text, which is returned from the **read()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1036e FWKERN cannot open the file *file_name***.**

Explanation: FWKERN cannot open the specified file. This error occurs when a START command is processing or when an automatic start attempt is made for a daemon.

System Action: The program continues. The START request fails.

System Programmer Response: Do the following:

- Ensure that there is a valid DD statement for the file in the FWKERN PROC.
- Ensure that FWKERN has READ authority for the file.
- Ensure that the file exists and is not allocated by any other process.

ICAM1037e FWKERN detects file error for file *file_name***.**

Explanation: FWKERN detects a file error for the specified file. This error occurs when a START command is processing or when an automatic start attempt is made for a daemon.

System Action: The program continues. The START request fails.

System Programmer Response: Do the following:

- Ensure that the file is not empty.
- Ensure that the file contents are not corrupt.

ICAM1038e FWKERN cannot allocate memory.

Explanation: FWKERN is not able to allocate memory. This error should not occur during normal processing.

System Action: The program continues.

System Programmer Response: Increase region size for FWKERN and try the operation again.

ICAM1039e FWKERN cannot lock mutex. *error_text*

Explanation: The FWKERN cannot lock a mutex. Additional information about the error is given by the C library error text, which is returned from the **pthread_mutex_lock()** function. This error should not occur during normal processing.

System Action: The program continues with reduced function.

ICAM1040e FWKERN cannot unlock mutex. *error_text*

Explanation: FWKERN cannot unlock a mutex. Additional information about the error is given by the C library error text, which is returned from the **pthread_mutex_unlock()** function. This error should not occur during normal processing.

System Action: The program continues with reduced function.

ICAM1041e FWKERN cannot create thread. *error_text*

Explanation: FWKERN is not able to create a thread. Additional information about the error is given by the C library error text, which is returned from the **pthread_create()** function. The most likely reason is that the system limit for the maximum number of threads for each process is exceeded. This error should not occur during normal processing.

System Action: The program continues.

System Programmer Response: Do 1 of the following:

- Wait for at least one request in progress to complete, then try the request again.
- Ensure that the MAXTHREADTASKS and MAXTHREADS values in the OS/390 UNIX System Services parmlib member are set to sufficient levels to accommodate the system load. For more information on setting MAXTHREADTASKS and MAXTHREADS see OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAM1042e FWKERN cannot detach thread. *error_text*

Explanation: FWKERN is not able to detach a thread. Additional information about the error is given by the C library error text, which is returned from the **pthread_detach()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1043e FWKERN cannot join to thread. *error_text*

Explanation: FWKERN is not able to join to a thread. Additional information about the error is given by the C library error text, which is returned from the **pthread_join()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1044e FWKERN cannot start all Firewall daemons.

Explanation: FWKERN cannot start all of the Firewall daemons because an error occurs during initialization of one of the daemons.

System Action: The program continues.

Operator Response: Check the console log for any messages or dumps that occur prior to this message, and notify the system programmer.

System Programmer Response: Use the diagnostic data collected by the operator to determine the probable cause of the failure.

ICAM1045e *function* **failed - cannot obtain status of Firewall daemon.** *error_text*

Explanation: FWKERN is not able to determine the status of a Firewall daemon, using either the **waitpid()** function or the **kill()** function with signal 0, as displayed in the message. Additional information about the error is given by the C library error text, which is returned from the **waitpid()** or **kill()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1046e FWKERN cannot retrieve signal. *error_text*

Explanation: FWKERN is not able to retrieve a signal from the pending signals queue. Additional information about the error is given by the C library error text, which is returned from the **sigwait()** function. This error should not occur during normal processing.

System Action: The program continues.

ICAM1047e FWKERN cannot create required thread.

Explanation: FWKERN is not able to create a required thread. The most likely reason is that the system limit for the maximum number of threads for each process has been reached.

System Action: The program ends.

System Programmer Response: Ensure that the MAXTHREADTASKS and MAXTHREADS values in the OS/390 UNIX System Services parmlib member are set to sufficient levels to accommodate the system load. For more information on setting MAXTHREADTASKS and MAXTHREADS see OS/390 SecureWay Security Server Firewall Technologies Guide and Reference, SC24-5835.

ICAM1048e FWKERN cannot initialize mutex. *error_text*

Explanation: FWKERN cannot initialize a mutex. Additional information about the error is given by the C library error text, which is returned from the **pthread_mutex_init()** function. This error should not occur during normal processing.

System Action: The program ends.

ICAM1049e FWKERN cannot block signals. *error_text*

Explanation: FWKERN is not able to block signals. Additional information about the error is given by the C library error text, which is returned from the **sigprocmask()** function. This error should not occur during normal processing.

System Action: The program ends.

ICAM1050e FWKERN cannot initialize MODIFY command queue.

Explanation: FWKERN initialization of the **MODIFY** command queue is not successful.

System Action: The program continues with reduced function. The **MODIFY** command service is not available.

ICAM1051e FWKERN initialization failure.

Explanation: FWKERN fails to initialize.

System Action: The program ends.

Operator Response: Do the following:

- Check for previous messages that indicate the reason for the failure.
- If the problem can be corrected, start the FWKERN again.
- If the problem continues, collect the error information and notify the system programmer.

ICAM1052e FWKERN cannot determine user authorization.

Explanation: The user issued a command that requires an authorization check. The authorization check encounters a warning condition. In most cases, the reasons for the failure are:

- The **FWKERN.START.REQUEST** resource profile is not defined to your External Security Manager product.
- Your External Security Manager product is not active.
- The **FACILITY** class is not defined to your External Security Manager product.
- The **FACILITY** class is not active.

System Action: The program continues.

System Programmer Response: Do the following:

- Ensure that your External Security Manager product is installed and active.
- Ensure that the **FACILITY** class is defined to your External Security Manager product and is active.
- Ensure that the **FWKERN.START.REQUEST** resource profile is defined to your External Security Manager product and is active.

ICAM1053i START ALL command has completed.

Explanation: FWKERN has completed processing of the START ALL command issued by the operator.

System Action: The program continues.

ICAM1054i FWKERN receives a *request_name* **request from TSO user** *user_ID***.**

Explanation: FWKERN has received the specified request from the specified TSO user.

System Action: FWKERN processes the request and continues.

ICAM1055e FWKERN encounters error during Daemon *daemon_name* **STOP request.** *error_text*

Explanation: FWKERN cannot perform the stop request synchronously. Additional information about the error is provided by the C library error text, which is returned from the **pthread_cond_timedwait()** function. If the request is to stop all processes, they are stopped asynchronously. This error should not occur during normal processing.

System Action: The program continues with reduced function.

ICAM1056e FWKERN encounters error during STOP request. *error_text*

Explanation: FWKERN encounters an error while stopping. Additional information about the error is provided by the C library error text, which is returned from the **pthread_kill()** function. The STOP FWKERN processing continues. This error should not occur during normal processing.

System Action: The program continues with reduced function.

ICAM1057i Release *ver.rel.mod* **Service Level** *level***. Created on** *date***.**

Explanation: This message is issued when the Firewall kernel is started, or when the Firewall level is queried using the QUERY LEVEL command. The system returns the version, release, modification level, service level and creation date.

System Action: The program continues.

ICAM1059e Cannot create a system level token, *token_name***. System return code:** *return_code***.**

Explanation: The FWKERN control task or a daemon control task cannot create a system level named token. The specified value is returned from the **ieantcr()** callable service. The name of the token being created is also displayed. This error should not occur during normal processing.

System Action: If the token is for the FWKERN control task, processing continues without the token. If the token is for a daemon control task, the daemon does not start and additional daemons that start after this daemon are also not started.

ICAM1060e Cannot retrieve a system level token, *token_name***. System return code:** *return_code***.**

Explanation: The FWKERN control task or a daemon control task cannot retrieve a system level named token. The return code is returned from the **ieantrt()** callable service. The name of the token being retrieved is also displayed. This error should not occur during normal processing.

System Action: The FWKERN control task continues. The daemon may not start successfully.

ICAM1061e FWKERN cannot start because it is already running.

Explanation: There can only be one active FWKERN control task. A second FWKERN control task cannot be started when one is already running.

System Action: The program ends.

Operator Response: Use the **STOP** command to stop the active FWKERN control task. Try the request again.

ICAM1062e Daemon *daemon_name* **cannot start because it is already running.**

Explanation: There can only be one active instance of a daemon. A second daemon of the same type cannot be started when one is already running. This problem can occur if the daemon is started without using the **modify fwkern,start** command.

System Action: The daemon is not started.

Operator Response: Use the **stop** command to stop the active daemon. Try the request again. Start daemons only using the **modify fwkern,start** command.

ICAM1063e Cannot initialize OS/390 PAUSE service. System return code: *return_code***, system reason code:** *reason_code***.**

Explanation: The FWKERN control task cannot identify the event control blocks used by FWKERN when starting a daemon. The return code and reason code values are returned from the **BPX1MPI()** callable service. This error should not occur during normal processing.

System Action: The FWKERN control task continues, but the daemon is not started. Additional daemons that start after this daemon are also not started.

ICAM1064e Cannot start the OS/390 PAUSE service. System return code : *return_code***, system reason code:** *reason_code***.**

Explanation: The FWKERN control task cannot wait on the event control blocks used by FWKERN when starting a daemon. The return code and reason code values are returned from the **BPX1MP()** callable service. This error should not occur during normal processing.

System Action: The FWKERN control task continues, but the daemon is not started. Additional daemons that start after this daemon are also not started.

ICAM1065e Daemon *daemon_name* **cannot be started in a separate address space.**

Explanation: The daemon specified in the message is on the list of daemons to be started by the FWKERN in a separate address space. However, a previous problem with event control block initialization prevents FWKERN from doing this.

System Action: The FWKERN control task continues, but the daemon is not started. Additional daemons that start after this daemon are also not started.

Operator Response: Check the console log for any messages or dumps that occur prior to this message, and notify the system programmer.

System Programmer Response: Use the diagnostic data collected by the operator to determine the probable cause of the failure.

ICAM1066e START parameters for Daemon *daemon_name* **are too long.**

Explanation: FWKERN cannot start the daemon specified in the message because the start options specified on the **modify fwkern,start** command are too long. The maximum length for start options is 95 for a daemon that runs in its own address space.

System Action: The FWKERN control task continues, but the daemon is not started. Additional daemons that start after this daemon are also not started.

ICAM1067e Incorrect arguments to start Daemon *daemon_name***.**

Explanation: Attempting to start the daemon specified in the message fails because of incorrect arguments passed to the start procedure. This can occur if the procedure to start the daemon is invoked directly or if the procedure is modified incorrectly.

System Action: The FWKERN control task continues, but the daemon is not started. Additional daemons that start after this daemon are also not started.

Operator Response: Start all daemons through FWKERN, using the **modify fwkern,start** command.

ICAM1068e FWKERN cannot initialize condition variable. *error_text*

Explanation: FWKERN cannot initialize a condition variable. Additional information about the error is given by the C library error text, which is returned from the **pthread_cond_init()** function. This error should not occur during normal processing.

System Action: The program ends.

ICAM1069i Daemon *daemon_name* **has been started.**

Explanation: FWKERN has successfully started the specified daemon.

ICAM1070i Cannot update status. Reason code: *reason_code***.**

Explanation: A daemon cannot inform FWKERN of its status. This error should not occur during normal processing.

System Action: The daemon ends.

ICAM1071e FWKERN cannot initialize thread attribute object. *error_text*

Explanation: FWKERN cannot initialize a thread attribute object. Additional information about the error is given by the C library error text, which is returned from the **pthread_attr_init()** function. This error should not occur during normal processing.

System Action: The program ends.

ICAM1072e FWKERN cannot set detach state of thread attribute object. *error_text*

Explanation: FWKERN cannot set detach state of a thread attribute object. Additional information about the error is given by the C library error text, which is returned from the **pthread_attr_setdetachstate()** function. This error should not occur during normal processing.

System Action: The program ends.

ICAM1073e FWKERN cannot set weight of thread attribute object. *error_text*

Explanation: FWKERN cannot set the weight of a thread attribute object. Additional information about the error is given by the C library error text, which is returned from the **pthread_attr_setweight_np()** function. This error should not occur during normal processing.

System Action: The program ends.

ICAM1074e FWKERN START parameters (*parms*) are not valid.

Explanation: An attempt to start the FWKERN procedure via the **start fwkern,parms='start_parameters'** command specified start up parameters that are not valid.

System Action: The FWKERN procedure ends and the Firewall kernel is not started.

User Response: Check the start up parameters specified on the command and issue the start command again.

ICAM1075i All configured FWKERN daemons are already started.

Explanation: An attempt to start the Firewall daemons failed because all of the configured Firewall daemons are already active.

System Action: The status of the currently running daemons remains unchanged.

User Response: None

ICAM1078e Parameters cannot be specified for *daemon* daemon.

Explanation: An attempt to start the specified Firewall daemon via the **modify fwkern,start daemon_name** command failed because the Firewall daemons cannot be passed parameters on the **modify** command.

System Action: The modify commands ends and the daemon is not started.

User Response: Remove the start up parameters from the command and issue the **modify** command to start the daemon.

ICAM1079e Daemon *daemon* cannot be refreshed because it has status of *daemon_status*.

Explanation: The daemon status is not READY. A refresh request is only honored if the daemon has a status of READY.

System Action: The program continues.

Operator Response: Do the following, then try the request again:

- Ensure that the daemon specified is the one actually intended for refreshing.
- If the status is INITIALIZING, then wait for the status to change to READY.

ICAM1080e Unable to complete IP/SEC initialization for *stack_name*.

Explanation: FWSTACKD was unable to complete the initialization of the IP/SEC services on the specified stack. More information on the specific failure can be found in the SYSLOG daemon output or in the output for the started task (ICAPSTAK).

System Action: FWSTACKD continues without completing IP/SEC initialization on the specified stack.

ICAM1081i IP/SEC initialization complete for *stack_name*.

Explanation: FWSTACKD has completed the initialization of the IP/SEC services on the specified stack.

System Action: The program continues.

ICAM1082i Firewall daemon *daemon_name* is not configured.

Explanation: The specified daemon is not running and has not been configured to start.

System Action: The program continues.

ICAM1090i FWKERN waiting for TCP/IP initialization to complete.

Explanation: FWKERN initialization cannot complete until TCP/IP initialization has completed.

System Action: FWKERN pauses to allow TCP/IP to initialization to complete.

ICAM1091e Timeout waiting for TCP/IP initialization to complete.

Explanation: TCP/IP initialization did not complete. Firewall function might be limited.

System Action: The program continues.

ICAM1095e *function* **failed.** *error_text*

Explanation: FWKERN encounters an error executing the function displayed in the message. Additional information about the error is given by the error text.

System Action: The program ends.

ICAM1099e See the CTOUT data definition for more information.

Explanation: The CTOUT data definition, defined by the FWKERN procedure, will contain more information about the error.

System Action: The program continues. The request fails.

User Response: Use the information in CTOUT to correct the error, then reactivate the definitions that failed.

ICAM1100i Current child process status for *daemon_name***:**

Explanation: FWKERN has issued a query status request to one of the multi-threaded daemons under its control. The main task for this daemon has polled each of its subtasks for status on the number of active threads and sent this data back to FWKERN for display in this message. There will be one message for each subtask started by the multi-threaded daemon to which FWKERN had sent the initial request.

System Action: The program continues.

ICAM1101e *function_name* **failed - cannot obtain status of daemon.** *error_text*

System Action: The program continues.

ICAM1103e Cannot obtain status of daemon. Timeout waiting for response.

System Action: The program continues.

Explanation: FWKERN has issued a query status request to one of the multi-threaded daemons under its control. The main task for this daemon has polled each of its subtasks for status on the number of active threads and sent this data back to FWKERN for display in this message. There will be one message for each subtask started by the multi-threaded daemon to which FWKERN had sent the initial request.

System Action: The program continues.

ICAM1108e FWKERN cannot start daemon *daemon_name***; failure during initialization**

Explanation: FWKERN attempted to start the specified daemon and the daemon encountered an error during initialization. Refer to the file or dataset specified by the **outputfile** field in the daemon configuration for more information about the error. Daemon configuration can be obtained by issuing **fwdaemon cmd=list format=long**.

System Action: The FWKERN control task continues, but the daemon is not started. Additional daemons that start after this daemon are also not started.

User Response: Correct the error and start the daemon again. If the problem continues, contact the service representative.

ICAM1150e Internal error: RC = *return_code*

Explanation: An internal error has been encountered while processing the **fwdaemon** command.

User Response: Note the specified return code and contact your service representative.

System Action: The command ends abnormally.

ICAM1151e Daemon *daemon_name* **is not configured**

Explanation: The **fwdaemon** command attempted to start the specified daemon and the daemon cannot be started because it is not currently configured in the daemon configuration file.

User Response: Verify that the specified daemon is correct. If so, use the **fwdaemon** command to configure the daemon.

System Action: The program continues.

ICAM1152e Daemon *daemon_name* **can not be started; status is not DOWN**

Explanation: The **fwdaemon** command attempted to start the specified daemon and the daemon cannot be started because it is not currently in the DOWN state.

User Response: Issue the **fwdaemon** to query the state of the daemon.

- If the status is READY, the daemon is already running.
- If the status is either INITIALIZING or STOPPING or UNKNOWN, wait for the status to change to READY or DOWN. Issue the command again if the status changes to DOWN.
- If the status is DISABLED, the daemon cannot be started because the OS/390 Security Server was not purchased.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1153e Daemon *daemon_name* **can not be stopped; status is not READY**

Explanation: The **fwdaemon** command attempted to stop the specified daemon and the daemon cannot be stopped because it is not currently in the READY state.

User Response: Issue the **fwdaemon** to query the state of the daemon.

- If the status is DOWN, the daemon is not running.
- If the status is either INITIALIZING or STOPPING or UNKNOWN, wait for the status to change to READY or DOWN. Issue the command again if the status changes to READY.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1154e Daemon *daemon_name* **did not start within the specified TIMEOUT limit**

Explanation: The **fwdaemon** command attempted to start the specified daemon and the daemon did not start within the time out limit specified in the daemon configuration file.

User Response: Issue the command again. If the problem continues, contact the service representative.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1156e Error in daemon configuration entry for *daemon_name*

Explanation: The **fwdaemon** command attempted to start the specified daemon and the daemon cannot be started because the entry for the daemon in the daemon configuration file contains an error.

User Response: Check the entry for the specified daemon using the **fwdaemon cmd=list** command. Correct any problems through the **fwdaemon cmd=change** command.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1157e Daemon *daemon_name* **can not be started; can not create a process**

Explanation: The **fwdaemon** command attempted to start the specified daemon and the daemon cannot be started because a process cannot be created for the daemon.

User Response: Issue the command again. If the problem continues, contact the service representative.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1158e Daemon *daemon_name* **can not be stopped; can not kill process**

Explanation: The **fwdaemon** command attempted to stop the specified daemon and the daemon cannot be stopped because the process cannot be killed.

User Response: Issue the command again. If the problem continues, contact the service representative.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1159e Daemon *daemon_name* **can not be stopped; can not identify process**

Explanation: The **fwdaemon** command attempted to stop the specified daemon and the daemon cannot be stopped because the process cannot be located.

User Response: Issue the command again. If the problem continues, contact the service representative.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1160e Daemon *daemon_name* **can not be stopped; time out waiting for daemon to end**

Explanation: The **fwdaemon** command attempted to stop the specified daemon and the daemon cannot be stopped because a time out occurred waiting for the daemon to end.

User Response: Issue the command again. If the problem continues, contact the service representative.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1161e Daemon *daemon_name* **does not support refresh option**

Explanation: The **fwdaemon** command attempted to refresh the specified daemon and the daemon does not support the refresh option.

User Response: None.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1163e Daemon *daemon_name* **can not be started; OS/390 Security Server is not enabled**

Explanation: The **fwdaemon** command attempted to start the specified daemon and the daemon cannot be started because the OS/390 Security Server was not enabled.

User Response: None.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1164e Issuer has insufficient authority to START or STOP daemons

Explanation: The **fwdaemon** command attempted to start one or more daemons and the issuer of the command was not authorized to start or stop daemons. The issuer must be authorized to the **FWKERN.START.REQUEST** facility.

User Response: Contact system administrator for authority and issue the command again.

System Action: The **fwdaemon** command ends and the state of the daemons are unchanged.

ICAM1165e Daemon *daemon_name* **can not be started; failure during initialization**

Explanation: The **fwdaemon** command attempted to start the specified daemon and the daemon encountered an error during initialization. Refer to the file or dataset specified by the **outputfile** field in the daemon configuration for more information about the error. Daemon configuration can be obtained by issuing **fwdaemon cmd=list format=long**.

User Response: Correct the error and issue the command again. If the problem continues, contact the service representative.

System Action: The **fwdaemon** command ends and the state of the daemon is unchanged.

ICAM1166e FWKERN communication error encountered. RC = *return_code*

Explanation: The **fwdaemon** command attempted to communicate with **fwkern** and encountered a communication error. The most likely reason for the failure is that **fwkern** has not been started.

User Response: Verify **fwkern** has been started. If not, start it and issue the command again. If the problem continues, contact the service representative.

System Action: The **fwdaemon** command ends and the state of the daemon(s) are unchanged.

ICAM2501e Required parameter, *parameter***, missing on command invocation**

Explanation: The **fwlogtxt** command was issued without the required parameters. The usage statement for the command follows this message.

System Action: The **fwlogtxt** command exits.

User Response: Issue the command again with the required parameters.

ICAM2502e Unable to open file *filename* **-** *error_text*

Explanation: The **fwlogtxt** command attempted to open the specified file and the open failed. The reason for the failure is documented in the error information at the end of the message.

System Action: The **fwlogtxt** command exits.

User Response: Using the error information presented in the message, correct the specified problem and issue the command again. If the problem continues, contact the service representative.

ICAM2503e Unable to read file *filename* **-** *error_text*

Explanation: The **fwlogtxt** command attempted to read the specified file and the read failed. The reason for the failure is documented in the error information at the end of the message.

System Action: The **fwlogtxt**command exits.

User Response: Using the error information presented in the message, correct the specified problem and issue the command again. If the problem continues, contact the service representative.

ICAM5000e Specified tunnel encryption algorithm is not supported.

Explanation: A user has attempted to define a tunnel with an unsupported algorithm.

System Action: The command exits.

User Response: Check the tunnel definition and issue the command again.

ICAM5001e Error resolving file *file***:** *error*

Explanation: The system was unable to resolve the long name of the specified file.

System Action: The command exits.

User Response: Check the file name and issue the command again.

ICAM5002e stackname needed with -S option.

Explanation: Stack option selected without stack name.

System Action: The command exits.

User Response: Check the parameters and issue the command again.

ICAM5003e Error on socket open for stack (*stackname*) : *error*

Explanation: The system was unable to successfully open a socket on the specified stack.

System Action: The error message is returned which may cause a command exit.

User Response: The user should verify the status of the specified stack and reissue the command.

ICAM5004e Error on ioctl call (*ioctl*) : *error*

Explanation: The system was unable to successfully perform an ioctl call.

System Action: The error message is returned which may cause a command exit.

User Response: The user should verify the status of the stack and reissue the command.

ICAM5005e Data supplied on ioctl not valid.

Explanation: One or more fields on an ioctl structure were not correctly set.

System Action: The error message is returned which may cause a command exit.

User Response: The user should verify the status of the stack and reissue the command.

ICAM5006e Stack memory allocation failure.

Explanation: A memory allocation call in the TCP/IP Stack code failed.

System Action: The error message is returned which may cause a command exit.

User Response: The user should verify the status of the stack and reissue the command.

ICAM5007e Memory allocation failure : *error*

Explanation: A memory allocation call in the code failed.

System Action: The error message is returned which may cause a command exit.

User Response: The user should verify the amount of available memory and reissue the command.

ICAM5008e Specified manual tunnel (*tunnel_name*) not found.

Explanation: The TCP/IP Stack tunnel cache did not contain the tunnel name.

System Action: The error message is returned which may cause a command exit.

User Response: The user should verify the active tunnels and reissue the command.

ICAM5009e Queued tunnels not found on subsequent get.

Explanation: The TCP/IP Stack queue did not contained a set of tunnels for retrieval.

System Action: The error message is returned which may cause a command exit.

User Response: The user should verify the status of the stack and reissue the command.

ICAM5010e Stack return code not valid: *return_code*.

Explanation: The TCP/IP Stack retunred an unknown error code.

System Action: The error message is returned which may cause a command exit.

User Response: The user should verify the status of the stack and reissue the command.

ICAM5011e Specified dynamic tunnel (*conn_id*:*src_id*:*dst_id*:*service_id*:*frule_id*:*dyntun_id*:*dynconn_id*) not found.

Explanation: The TCP/IP Stack tunnel cache did not contain the tunnel name.

System Action: The error message is returned which may cause a command exit.

User Response: The user should verify the active tunnels and reissue the command.

ICAM5013e Specified dynamic tunnel (*conn_id*:*src_id*:*dst_id*:*service_id*:*frule_id*:*dyntun_id*:*dynconn_id*) already active.

Explanation: Tunnel ID previously activated or the name and SPI are the same as a currently active tunnel.

System Action: The command exits.

User Response: Check the active tunnel definitions and issue the command again.

ICAM5014e Internal tunnel algorithm conversion failed.

Explanation: Tunnel activation failed when initializing an internal stack data structure.

System Action: The command exits.

User Response: Contact the service representative.

ICAM5102e -S needed with -c, -u, -i, -d, -m and -p options

Explanation: Stack based option selected without a specified stack.

System Action: The command exits.

User Response: Check parameters and issue the command again.

ICAM5103e -S needed with -c, -u, -i, and -d options

Explanation: Stack based option selected without a specified stack.

System Action: The command exits.

User Response: Check parameters and issue the command again.

ICAM5110i IP stack default filter rules in effect for firewall TCP/IP stack *stackname***.**

Explanation: A request to query the status of the IP filter rules for the indicated firewall TCP/IP stack was issued. The query request detected that the system (no stack name specified) or a particular firewall TCP/IP stack was being protected by default filter rules. Default filter rules permit local IP traffic from the secure and nonsecure interfaces and deny all routed traffic.

System Action: The command completes.

User Response: None

ICAM5210e *file*: **Line** *lineno*: **my_addr not valid:** *value***.**

Explanation: Value encountered in the **my_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5211e *file*: **Line** *lineno*: **no his_addr value.**

Explanation: A user has defined a policy rule without a **his_addr** value.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5212e *file*: **Line** *lineno*: **his_addr not valid:** *value***.**

Explanation: Value encountered in the **his_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5213e *file*: **Line** *lineno*: **no context_id value.**

Explanation: A user has defined a policy rule without a **context_id** value.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5214e *file*: **Line** *lineno*: **context_id not valid:** *value***.**

Explanation: Value encountered in the **context_id** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5215e *file*: **Line** *lineno*: **no sender enc/mac value.**

Explanation: A user has defined a policy rule without a **sender enc/mac** value.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5216e *file*: **Line** *lineno*: **sender enc/mac not valid:** *value***.**

Explanation: Value encountered in the **sender enc/mac** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5217e *file*: **Line** *lineno*: **no enc/mac transport value.**

Explanation: A user has defined a policy rule without a **enc/mac transport** value.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5218e *file*: **Line** *lineno*: **no sender mac_first value.**

Explanation: A user has defined a policy rule without a **sender mac_first** value.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5219e *file*: **Line** *lineno*: **sender mac_first not valid:** *value*.

Explanation: Value encountered in the **sender mac_first** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5220e *file*: **Line** *lineno*: **no key value.**

Explanation: A user has defined a policy rule without a **key** value.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5221e *file*: **Line** *lineno*: **no receiver enc/mac value.**

Explanation: A user has defined a policy rule without a **receiver enc/mac** value.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5222e *file*: **Line** *lineno*: **receiver enc/mac not valid:** *value*.

Explanation: Value encountered in the **receiver enc/mac** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5223e *file*: **Line** *lineno*: **no receiver mac_first value.**

Explanation: A user has defined a policy rule without a **receiver mac_first** value.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5224e *file*: **Line** *lineno*: **receiver mac_first not valid:** *value*.

Explanation: Value encountered in the **receiver mac_first** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5226e *prog*: **Error encountered while parsing policy rules file:** *file*.

Explanation: The parse procedure detected an error in the policy rules file.

System Action: The command exits.

User Response: Correct the input file and issue the command again.

ICAM5227e *command*: **No matching policy definition found in** *policyfile* **for tunnel (***tunnel_name***) found in** *contextfile*.

Explanation: A user has attempted to define a context rule without a matching policy rule.

System Action: The command exits.

User Response: Correct the input files and issue the command again.

ICAM5228e Tunnel defined with unsupported encryption algorithm.

Explanation: A user has attempted to define a tunnel with an unsupported encryption algorithm.

System Action: The command exits.

User Response: Check the tunnel definitions and issue the command again.

ICAM5229e *file*: **Line** *lineno*: **s_enc_mac_alg format not valid:** *line*.

Explanation: Value encountered in the **s_enc_mac_alg** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5230e *file*: **Line** *lineno*: **s_enc_mac_key_len format not valid:** *line*.

Explanation: Value encountered in the **s_enc_mac_key_len** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5231e *file*: **Line** *lineno*: **s_enc_mac_key format not valid:** *line*.

Explanation: Value encountered in the **s_enc_mac_key** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5232e *file*: **Line** *lineno*: **r_enc_mac_alg format not valid:** *line*.

Explanation: Value encountered in the **r_enc_mac_alg** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5233e *file*: **Line** *lineno*: **r_enc_mac_key_len format not valid:** *line*.

Explanation: Value encountered in the **r_enc_mac_key_len** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5234e *file*: **Line** *lineno*: **r_enc_mac_key format not valid:** *line*.

Explanation: Value encountered in the **r_enc_mac_key** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5235e *file*: **Line** *lineno*: **s_enc_fmt format not valid:** *line*.

Explanation: Value encountered in the **s_enc_fmt** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5236e *file*: **Line** *lineno*: **r_enc_fmt format not valid:** *line*.

Explanation: Value encountered in the **r_enc_fmt** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5237e *file*: **Line** *lineno*: **s_mac_fmt format not valid:** *line*.

Explanation: Value encountered in the **s_mac_fmt** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5238e *file*: **Line** *lineno*: **r_mac_fmt format not valid:** *line*.

Explanation: Value encountered in the **r_mac_fmt** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5239e *file*: **Line** *lineno*: **s_enc_replay format not valid:** *line*.

Explanation: Value encountered in the **s_enc_replay** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5240e *file*: **Line** *lineno*: **r_enc_replay format not valid:** *line*.

Explanation: Value encountered in the **r_enc_replay** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5241e *file*: **Line** *lineno*: **s_mac_replay format not valid:** *line*.

Explanation: Value encountered in the **s_mac_replay** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5242e *file*: **Line** *lineno*: **r_mac_replay format not valid:** *line*.

Explanation: Value encountered in the **r_mac_replay** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5243e *file*: **Line** *lineno*: **no timeout value.**

Explanation: A user has attempted to define a policy rule without a **timeout** value.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5244e *file*: **Line** *lineno*: **enc/mac transport not valid :** *value*.

Explanation: Value encountered in the **enc/mac transport** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5245e *prog*: **Policy rules file (***file***) is empty.**

Explanation: The policy rules file has no definitions.

System Action: The command exits.

User Response: Correct the input file and issue the command again.

ICAM5246e *prog*: **Error encountered while parsing context rules file:** *file*.

Explanation: The parse procedure detected an error in the context rules file.

System Action: The command exits.

User Response: Correct the input file and issue the command again.

ICAM5247e *prog*: **Context rules file (***file***) is empty.**

Explanation: The context rules file has no definitions.

System Action: The command exits.

User Response: Correct the input file and issue the command again.

ICAM5248e *prog*: **Final context rule missing lines in** *file*.

Explanation: One or more lines of the final context rule are missing.

System Action: The command exits.

User Response: Correct the input file and issue the command again.

ICAM5249e *file*: **Replay prevention not supported by header format in tunnel (***tunnel_name***).**

Explanation: Specified tunnel has replay prevention selected without draft level header chosen.

System Action: The command exits.

User Response: Correct the input file and issue the command again.

ICAM5250e *file*: **Authentication not supported by header format in tunnel (***tunnel_name***).**

Explanation: Authentication field value is not valid with selected header format.

System Action: The command exits.

User Response: Correct the input file and issue the command again.

ICAM5251e *file*: **Encryption not supported by header format in tunnel (***tunnel_name***).**

Explanation: Encryption field value is not valid with selected header format.

System Action: The command exits.

User Response: Correct the input file and issue the command again.

ICAM5252e *file*: **Encryption choice in tunnel (***tunnel_name***) requires authentication.**

Explanation: Header format chosen requires encryption and/or authentication.

System Action: The command exits.

User Response: Correct the input file and issue the command again.

ICAM5301e *file*: **Line** *lineno*: **address format (his_addr) not valid:** *line*.

Explanation: Internet address encountered in the **his_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5302e *file*: **Line** *lineno*: **named address (his_addr) not valid:** *line*.

Explanation: Named address encountered in the **his_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5303e *file*: **Line** *lineno*: **address format (my_addr):** *line* **not valid.**

Explanation: Internet address encountered in the **my_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5304e *file*: **Line** *lineno*: **named address (my_addr) not valid:** *line*.

Explanation: Named address encountered in the **my_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5305e *file*: **Line** *lineno*: **context_id format not valid:** *line*.

Explanation: Value encountered in the **context_id** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5306e *file*: **Line** *lineno*: **his_esp_said format not valid:** *line*.

Explanation: Value encountered in the **his_esp_said** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5307e *file*: **Line** *lineno*: **his_ah_said format not valid:** *line*.

Explanation: Value encountered in the **his_ah_said** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5308e *file*: **Line** *lineno*: **my_esp_said format not valid:** *line*.

Explanation: Value encountered in the **my_esp_said** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5309e *file*: **Line** *lineno*: **my_ah_said format not valid:** *line*.

Explanation: Value encountered in the **my_ah_said** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5310e *file*: **Line** *lineno*: **s_enc_alg format not valid:** *line*.

Explanation: Value encountered in the **s_enc_alg** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5311e *file*: **Line** *lineno*: **s_enc_key_len format not valid:** *line*.

Explanation: Value encountered in the **s_enc_key_len** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5312e *file*: **Line** *lineno*: **s_enc_key format not valid:** *line*.

Explanation: Value encountered in the **s_enc_key** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5313e *file*: **Line** *lineno*: **r_enc_alg format not valid:** *line*.

Explanation: Value encountered in the **r_enc_alg** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5314e *file*: **Line** *lineno*: **r_enc_key_len format not valid:** *line*.

Explanation: Value encountered in the **r_enc_key_len** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5315e *file*: **Line** *lineno*: **r_enc_key format not valid:** *line*.

Explanation: Value encountered in the **r_enc_key** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5316e *file*: **Line** *lineno*: **s_mac_alg format not valid:** *line*.

Explanation: Value encountered in the **s_mac_alg** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5317e *file*: **Line** *lineno*: **s_mac_key_len format not valid:** *line*.

Explanation: Value encountered in the **s_mac_key_len** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5318e *file*: **Line** *lineno*: **s_mac_key format not valid:** *line*.

Explanation: Value encountered in the **s_mac_key** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5319e *file*: **Line** *lineno*: **r_mac_alg format not valid:** *line*.

Explanation: Value encountered in the **r_mac_alg** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5320e *file*: **Line** *lineno*: **r_mac_key_len format not valid:** *line*.

Explanation: Value encountered in the **r_mac_key_len** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5321e *file*: **Line** *lineno*: **r_mac_key format not valid:** *line*.

Explanation: Value encountered in the **r_mac_key** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5322e *file*: **Line** *lineno*: **start time format not valid:** *line*.

Explanation: Value encountered in the **start** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5323e *file*: **Line** *lineno*: **end time format not valid:** *line*.

Explanation: Value encountered in the **end** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5324e *file*: **start time (***start***) after end time (***end***).**

Explanation: The start time already exceeds the end time.\n

System Action: The command exits.

User Response: Verify the start and end times.

ICAM5326e *prog*: **Error on ioctl call (SIOCSCTXRUL):** *error*

Explanation: The system was unable to successfully cache a context rule.

System Action: The command exits.

ICAM5328e *prog* **: Line** *linenum* **: Wrong @command** *line*.

Explanation: A user has specified an @command directive (must be @del) that is not valid.

System Action: The command exits.

User Response: Check the usage and issue the command again.

ICAM5329e *file*: **Line** *lineno*: **address format (source_addr) not valid:** *line*.

Explanation: Internet address encountered in the **source_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5330e *file*: **Line** *lineno*: **named address (source_addr) not valid:** *line*.

Explanation: Named address encountered in the **source_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5331e *file*: **Line** *lineno*: **no dest_addr:** *line*.

Explanation: Input line does not contain a **dest_addr** field.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5332e *file*: **Line** *lineno*: **address format (dest_addr) not valid:** *line*.

Explanation: Internet address encountered in the **dest_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5333e *file*: **Line** *lineno*: **named address (dest_addr) not valid:** *line*.

Explanation: Named address encountered in the **dest_addr** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5334e *file*: **Line** *lineno*: **context_id format not valid:** *line*.

Explanation: Value encountered in the **context_id** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5335e *file*: **Line** *lineno*: **SAID format not valid:** *line*.

Explanation: Value encountered in the **SAID** field is not valid.

System Action: The command exits.

User Response: Correct the field and issue the command again.

ICAM5336e Specified manual tunnel (*tunnel_id***) already active.**

Explanation: Tunnel ID previously activated or the name and SPI are the same as a currently active tunnel.

System Action: The command exits.

User Response: Check the active tunnel definitions and issue the command again.

ICAM5400e Error opening file *file*: *error*

Explanation: The system was unable to open a specified NAT file.

System Action: The command exits.

User Response: Check the filename and issue the command again.

ICAM5401e Error on ioctl call (SIOCGNATTBL) : *error*

Explanation: The system was unable to retrieve the current NAT rules.

System Action: The command exits.

User Response: Check the TCP/IP stack status and issue the command again.

ICAM5402e -i option found but not -u

Explanation: The command was started with a -i without a -u.

System Action: The command exits.

User Response: Check parameters and issue the command again.

ICAM5403i: NAT support deactivated at *time* **on** *date*.

Explanation: NAT now is disabled.

ICAM5404i: NAT configuration updated at *time* **on** *date*.

Explanation: NAT configuration was updated.

ICAM5405i: NAT support (level *version.release***) initialized at** *time* **on** *date*.

Explanation: Firewall NAT support has been initialized.

ICAM5406e: Error on ioctl call (SIOCSNATCLK): *error*

Explanation: unable to refresh NAT rules.

System Action: The command exits.

User Response: Check the TCP/IP stack status and issue the command again.

ICAM5408e -S needed with -c, -u, -i, and -d options

Explanation: Stack based option selected without a specified stack.

System Action: The command exits.

User Response: Check the parameters and issue the command again.

ICAM5409e NAT support verification failed.

Explanation: Socket to specified TCP/IP stack could not be opened.

System Action: The command exits.

User Response: Check the stack and issue the command again.

ICAM5500w: File system containing *filename* **is** *nnn***% full.**

Explanation: The syslog daemon has detected that the file system containing the specified file has reached the specified percentage full threshold. If no actions are taken, and the file system continues to fill up, the syslog daemon will eventually fail to record data that it receives.

Operator Response: Notify the system administrator to alleviate the storage problems.

ICAM5501e: File system containing *filename* **is full; data not recorded.**

Explanation: The syslog daemon has detected that the file system containing the specified file has become full. As a result, the syslog daemon will no longer be able to record data into the specified file.

Operator Response: Notify the system administrator to alleviate the storage problems.

ICAM9998e: *command_name source target* **failed.**

Explanation: The install logic attempted to issue the specified command with the specified source and target operands and the command failed. Additional information about the error is given by the error information, which is returned from the specified command.

System Action: The program continues.

System Programmer Response: Correct the specified problem, then try the request again. If the problem continues, contact the service representative.

ICAM9999e: *command* **failed.**

Explanation: The install logic attempted to issue the specified command and the command failed. Additional information about the error is given by the error information, which is returned from the specified command.

System Action: The program continues.

System Programmer Response: Correct the specified problem, then try the request again. If the problem continues, contact the service representative.

Appendix A. Firewall Configuration Example

The following example can be used as a template to configure a firewall. In this example, Company A has 3 private LANs and wants to ping all of them from the secure network to any IP address on the internet. It also wants to permit hosts on 2 of the LANs specific access to the internet. The company wants to allow users on these 2 private LANs to:

- FTP from the private secure LAN to the Internet using the firewall's FTP proxy
- use `socksified` HTTP through the firewall's socks server

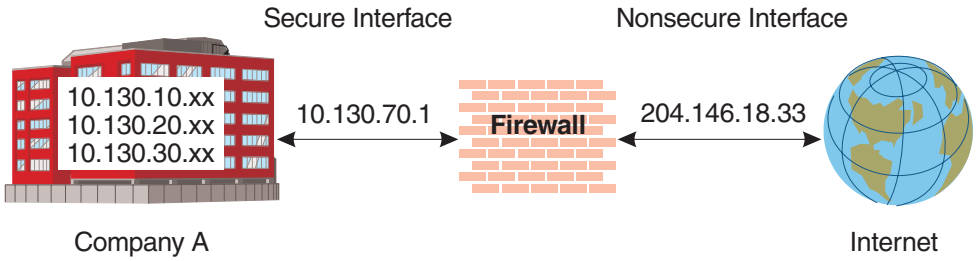


Figure 76. Internet Access of Company A via 2 LANs

In the following example, "The World" is the object and the services list is 18, which is predefined as the ID in service list.

The configuration needed to allow the above services uses a combination of default network objects, rules, user-defined rules and objects.

Note: The firewall is installed with many commonly used services, including filter and socks rules as well as the network object defined to contain all objects ("The World").

The following sections outline the necessary steps to achieve this configuration.

© Copyright IBM Corp. 1997, 2000

Part A - Defining Network Objects

The first step is to define network objects for the 3 private LANs. Since 2 of the private LANs will be given the same access capability, a network object group will be created to contain the 2 LANs as members. Another group object will be created to contain all hosts in Company A's private LANs.

Note: The network objects and groups we define will be used throughout this example along with the predefined network object "The World". "The World" network object encompasses all possible IP addresses.

Company A

Network Addresses	Firewall Addresses
Private LAN 1: 10.130.10.xx	10.130.70.1 (secure)
Private LAN 2: 10.130.20.xx	204.146.18.33 (non-secure)
Private LAN 3: 10.130.30.xx	

Define Network objects for Company A's 3 private LANs and a group network object to include the two network objects representing the hosts on 10.130.10.xx and 10.130.20.xx LANs.

```
name = LAN_1A        10.130.10.xx network
name = LAN_2A        10.130.20.xx network
name = LAN_3A        10.130.30.xx network

name = CompanyA_dept1     Group consisting of LAN_1A and
                                 LAN_2A members
name = CompanyA_dept2     Group consisting of LAN_1A and
                                 LAN_2A and LAN_3A members
```

User:

```
fwnwobj cmd=add name=LAN_1A type=Network desc="1 LAN"
                addr=10.130.10.0 mask=255.255.255.0

fwnwobj cmd=add name=LAN_2A type=Network desc="2 LAN"
                addr=10.130.20.0 mask=255.255.255.0

fwnwobj cmd=add name=LAN_3A type=Network desc="3 LAN"
                addr=10.130.30.0 mask=255.255.255.0

fwnwobj cmd=add name=Secure_FW_A type=Interface desc="Secure Firewall Interface"
                addr=10.130.70.1 mask=255.255.255.255

fwnwobj cmd=add name=NonSecure_FW_A type=Interface desc="Non-Secure Firewall Interface"
                addr=204.146.18.33 mask=255.255.255.255

fwnwgrp cmd=create name=CompanyA_dept1 desc="Company A dept1 members"
                   namelist="LAN_1A|LAN_2A"

fwnwgrp cmd=create name=CompanyA_dept2 desc="All hosts in Company A"
                   namelist="LAN_1A|LAN_2A|LAN_3A"
```

Part B - Allow LANs in Company A to Ping Internet Addresses

Allow hosts on all 3 private LANs to ping Internet IP addresses from the secure network. In order to allow the ping requests to flow from the secure to non-secure networks, predefined service 18 will be utilized.

- Service rule 18 allows ping to flow from the secure to non-secure network and the ping response to flow back from the non-secure network.

The following figure shows the ping service rule and the commands used to display the data.

```
   User:     fwservice cmd=list id=18
System:
                        id = 18
                      name = Ping
                      desc = Permit Ping outbound secure network to anywhere
                   rulelist = 13/f,12/b
                       log =
                   fragment =
                     tunnel =
                       time =
                      month =
                        day =
                    weekday =
                  timefilter =
   User:     fwfrule cmd=list id=13
System:
                        id = 13
                      type = permit
                      name = Ping
                      desc = ICMP port 8
                  protocol = icmp
                 srcopcode = eq
                   srcport = 8
                destopcode = eq
                  destport = 0
                 interface = both
                   routing = both
                 direction = both
                       log = no
                    tunnel =
                  fragment = yes
   User:     fwfrule cmd=list id=12
System:
                        id = 12
                      type = permit
                      name = Ping Response
                      desc = ICMP port 0
                  protocol = icmp
                 srcopcode = eq
                   srcport = 0
                destopcode = eq
                  destport = 0
                 interface = both
                   routing = both
                 direction = both
                       log = no
                    tunnel =
                  fragment = yes
```

Figure 77. Predefined Ping Service 18

Step 1 - Create a Connection

Using service rule 18, the next step is to create a connection that will use the ping service to control the traffic between Company A's 3 private LANs and the internet.

```
User:   fwconns cmd=create name="Allow Internet Pings"
               desc="Allow Pings from Company A to the Internet"
               source=CompanyA_dept2 destination="The World"
               servicelist=18
```

Step 2 - Update Filters to Activate Connection

In order to activate the newly defined connection, the filters must be updated. To update the filter information enter:

```
User:   fwfilter cmd=update
```

Once the **fwfilter** command completes, the firewall will now permit pings from any host within Company A to the internet. To display the filter rules that control this communication, enter **fwfilter cmd=list**. See Figure 78 for the displayed filter data.

```
User:   fwfilter cmd=list
System: # This file is generated by Firewall configuration.
        # DO NOT EDIT THIS FILE MANUALLY.
        #
        #       Between LAN_1A and The World
        #        Service : Ping
        # Description : Permit Ping outbound secure network to anywhere
        permit 10.130.10.0 255.255.255.0 0.0.0.0 0.0.0.0 icmp eq 8 eq 0 both both both l=n f=y s=m d=m
        permit 0.0.0.0 0.0.0.0 10.130.10.0 255.255.255.0 icmp eq 0 eq 0 both both both l=n f=y s=m d=m
        #       Between LAN_2A and The World
        #        Service : Ping
        # Description : Permit Ping outbound secure network to anywhere
        permit 10.130.20.0 255.255.255.0 0.0.0.0 0.0.0.0 icmp eq 8 eq 0 both both both l=n f=y s=m d=m
        permit 0.0.0.0 0.0.0.0 10.130.20.0 255.255.255.0 icmp eq 0 eq 0 both both both l=n f=y s=m d=m
        #       Between LAN_3A and The World
        #        Service : Ping
        # Description : Permit Ping outbound secure network to anywhere
        permit 10.130.30.0 255.255.255.0 0.0.0.0 0.0.0.0 icmp eq 8 eq 0 both both both l=n f=y s=m d=m
        permit 0.0.0.0 0.0.0.0 10.130.30.0 255.255.255.0 icmp eq 0 eq 0 both both both l=n f=y s=m d=m
```

Figure 78. Completed Ping Filter Rules

Part C - Configure FTP

Allow hosts on the 2 private LANs to FTP from the secure network to the non-secure network through the firewall FTP proxy. In order to allow the FTP to flow from the secure to non-secure networks, predefined service rules 13 and 14 will be utilized.

- Service rule 13 allows FTP traffic to flow from the secure network to the firewall and responses to flow from the firewall back to the secure network.
- Service rule 14 consists of the filter rules to allow FTP from the firewall to the non-secure network and responses to flow from the non-secure network back to the firewall.

The following figures contain a display of the FTP service rules and the commands used to display the data.

```
   User:     fwservice cmd=list id=13 format=long
System:
                        id = 13
                      name = FTP proxy out 1/2
                      desc = Permit FTP outbound from secure network to firewall
                  rulelist = 58/f,20/b,51/b,40/f,79/f,45/b
                       log =
                  fragment =
                    tunnel =
                      time =
                     month =
                       day =
                   weekday =
                 timefilter =

   User:     fwfrule cmd=list id=58 format=long
System:
                        id = 58
                      type = permit
                      name = Proxy FTP Control in secure 1/2
                      desc = TCP port 21
                  protocol = tcp
                 srcopcode = gt
                   srcport = 1023
                destopcode = eq
                  destport = 21
                 interface = secure
                   routing = local
                 direction = inbound
                       log = no
                    tunnel =
                  fragment = yes

                         .
                         .
                         .
```

Figure 79. Predefined FTP Service 13

To see the content of the other filter rules included in service 13, repeat the above **fwfrule** command replacing the ID value with the ID value of the filter rule you want to display.

```
User:     fwservice cmd=list id=14 format=long
System:
                    id = 14
                  name = FTP proxy out 2/2
                  desc = Permit FTP outbound from firewall to non-secure network
              rulelist = 56/f,16/b,48/b,39/f,77/f,42/b
                   log =
              fragment =
                tunnel =
                  time =
                 month =
                   day =
               weekday =
            timefilter =
```

Figure 80. Predefined FTP Services 14

To see the contents of service rule 14 and the filter rules included service 14, enter the **fwservice** command with an ID of 14, and enter the **fwfrule** command with the ID of the filter rule you want displayed.

Step 1 - Create Connections

Using service rules 13 and 14, the next step is to create a set of connections that will use the FTP service to control the traffic between the 2 private LANs and the internet.

- Allow traffic between CompanyA_dept1 and the secure interface on the Firewall

```
User:     fwconns cmd=create name="Allow FTPs from secure network to firewall"
          desc="FTPs from CompanyA to the Firewall"
          source=CompanyA_dept1 destination=Secure_FW_A
          servicelist=13
```

- Allow traffic between the non-secure interface on the Firewall and the internet

```
User:     fwconns cmd=create name="Allow FTPs out"
          desc="FTPs from Firewall to the Internet"
          source=NonSecure_FW_A destination="The World"
          servicelist=14
```

Step 2 - Update Filters to Activate Connection

In order to activate the newly defined connection, the filters must be updated. To update the filter information enter:

```
User:     fwfilter cmd=update
```

Once the **fwfilter** command completes, the firewall will now permit FTPs from any host within Company A to the internet. To display the filter rules that control this communication, enter **fwfilter cmd=list**. See Figure 81 on page 427 for the displayed filter data.

```
User:    fwfilter cmd=list
System:  # This file is generated by Firewall configuration.
         # DO NOT EDIT THIS FILE MANUALLY.
         #
         #       Between NonSecure_FW_A and The World
         #        Service : FTP proxy out 2/2
         # Description : Permit FTP outbound from firewall to non-secure network
         permit 10.130.1.70 255.255.255.255 0.0.0.0 0.0.0.0 tcp gt 1023 eq 21 non-secure local outbound l=n f=y s=m d=m
         permit 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp/ack eq 21 gt 1023 non-secure local inbound l=n f=y s=m d=m
         permit 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp eq 20 gt 1023 non-secure local inbound l=n f=y s=m d=m
         permit 10.130.1.70 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack gt 1023 eq 20 non-secure local outbound l=n f=y s=m d=m
         permit 10.130.1.70 255.255.255.255 0.0.0.0 0.0.0.0 tcp gt 1023 gt 1023 non-secure local outbound l=n f=y s=m d=m
         permit 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp/ack gt 1023 gt 1023 non-secure local inbound l=n f=y s=m d=m
         #       Between LAN_1A and Secure_FW_A
         #        Service : FTP proxy out 1/2
         # Description : Permit FTP outbound from secure network to firewall
         permit 10.130.10.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 eq 21 secure local inbound l=n f=y s=m d=m
         permit 10.130.70.1 255.255.255.255 10.130.10.0 255.255.255.0 tcp/ack eq 21 gt 1023 secure local outbound l=n f=y s=m d=m
         permit 10.130.70.1 255.255.255.255 10.130.10.0 255.255.255.0 tcp eq 20 gt 1023 secure local outbound l=n f=y s=m d=m
         permit 10.130.10.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp/ack gt 1023 eq 20 secure local inbound l=n f=y s=m d=m
         permit 10.130.10.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 gt 1023 secure local inbound l=n f=y s=m d=m
         permit 10.130.70.1 255.255.255.255 10.130.10.0 255.255.255.0 tcp/ack gt 1023 gt 1023 secure outbound l=n f=y s=m d=m
         #       Between LAN_2A and Secure_FW_A
         #        Service : FTP proxy out 1/2
         # Description : Permit FTP outbound from secure network to firewall
         permit 10.130.20.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 eq 21 secure local inbound l=n f=y s=m d=m
         permit 10.130.70.1 255.255.255.255 10.130.20.0 255.255.255.0 tcp/ack eq 21 gt 1023 secure local outbound l=n f=y s=m d=m
         permit 10.130.70.1 255.255.255.255 10.130.20.0 255.255.255.0 tcp eq 20 gt 1023 secure local outbound l=n f=y s=m d=m
         permit 10.130.20.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp/ack gt 1023 eq 20 secure local inbound l=n f=y s=m d=m
         permit 10.130.20.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 gt 1023 secure local inbound l=n f=y s=m d=m
         permit 10.130.70.1 255.255.255.255 10.130.20.0 255.255.255.0 tcp/ack gt 1023 gt 1023 secure outbound l=n f=y s=m d=m
         #       Between LAN_1A and The World
         #        Service : Ping
         # Description : Permit Ping outbound secure network to anywhere
         permit 10.130.10.0 255.255.255.0 0.0.0.0 0.0.0.0 icmp eq 8 eq 0 both both both l=n f=y s=m d=m
         permit 0.0.0.0 0.0.0.0 10.130.10.0 255.255.255.0 icmp eq 0 eq 0 both both both l=n f=y s=m d=m
         #       Between LAN_2A and The World
         #        Service : Ping
         # Description : Permit Ping outbound secure network to anywhere
         permit 10.130.20.0 255.255.255.0 0.0.0.0 0.0.0.0 icmp eq 8 eq 0 both both both l=n f=y s=m d=m
         permit 0.0.0.0 0.0.0.0 10.130.20.0 255.255.255.0 icmp eq 0 eq 0 both both both l=n f=y s=m d=m
         #       Between LAN_3A and The World
         #        Service : Ping
         # Description : Permit Ping outbound secure network to anywhere
         permit 10.130.30.0 255.255.255.0 0.0.0.0 0.0.0.0 icmp eq 8 eq 0 both both both l=n f=y s=m d=m
         permit 0.0.0.0 0.0.0.0 10.130.30.0 255.255.255.0 icmp eq 0 eq 0 both both both l=n f=y s=m d=m
```

Figure 81. Completed Ping and FTP Filter Rules

Part D - Configure HTTP

Allow hosts on the 2 private LANs to use Company A's HTTP requests to access data on the Internet. In order to allow the HTTP to go from the secure to nonsecure networks, a combination of predefined and user defined services and objects must be used.

- The objects/groups defined in Part 1 and the predefined object "The World".
- A predefined sock rule that allows `socksified` HTTP request (socks rule 501).
- The predefined services to use are:
  - Service Rule 2 - permits use of the socks server from the secure network to the firewall.
  - Service Rule 34 - denies access to the socks server from the internet (non-secure adapters).
- A user-defined service to permit HTTP traffic to flow between the firewall and the internet. Predefined filter rules 34 and 73 support this communication.

Once the necessary services are defined, the communication traffic must be put together with the source and target objects through a connection and the communication activated.

```
  User:   fwservice cmd=list id=2 format=long
System:
                    id = 2
                  name = Socks 1/2
                  desc = Permit use of Socks from secure network to the firewall
              rulelist = 53/f,14/b
                tunnel =
              fragment =
                tunnel =
                  time =
                 month =
                   day =
               weekday =
            timefilter =

  User:   fwfrule cmd=list id=53
System:
                    id = 53
                  type = permit
                  name = Socks TCP
                  desc = TCP in port 1080 secure
              protocol = tcp
             srcopcode = gt
               srcport = 1023
            destopcode = eq
              destport = 1080
             interface = secure
               routing = local
             direction = inbound
                   log = no
                tunnel =
              fragment = yes

  User:   fwfrule cmd=list id=14
System:
                    id = 14
                  type = permit
                  name = Socks TCP Ack
                  desc = TCP/ACK out port 1080 secure
              protocol = tcp/ack
             srcopcode = eq
               srcport = 1080
            destopcode = gt
              destport = 1023
             interface = secure
               routing = local
             direction = outbound
                   log = no
                tunnel =
              fragment = yes
```

Figure 82. Predefined HTTP Service 2

To see the content of the filter rules included in service 14, enter the **fwfrule** command with the id of the filter rule you want displayed.

Step 1 - Create a Service

Create a service that includes the filter rules to permit HTTP traffic to flow between the firewall and the internet. This is done through the **fwservice** command.

```
User:     fwservice cmd=create name="Allow HTTP to Internet"
                    desc="Allow HTTP traffic to flow from firewall to internet"
                    rulelist=34/b,73/f
```

Note: The use of /b and /f in the rulelist parameter will be discussed later when this service rule is used in a connection.

Step 2 - Create Connections

Using the predefined/user defined service rules and the predefined socks rule, the next step is to connect these rules with the network objects to control the traffic through the socks proxy from the 2 private LANs and the internet. First, display the ID of the service created in the previous step:

```
User:     fwservice cmd=list name="Allow HTTP to Internet" format=short
System:   501   Allow HTTP to Internet     Allow HTTP traffic to flow from firewall to internet
```

Using service rules 2, 34 and 501, the next step is to create a set of connections that will use the socks server and HTTP to control traffic between the 2 private LANs and the Internet.

- Allow HTTP traffic between CompanyA_dept1 and the secure interface on the firewall using the socks server

```
User:     fwconns cmd=create name="Allow HTTP out"
                  desc="HTTP requests from CompanyA to the Firewall socks server"
                  source=CompanyA_dept1 destination=Secure_FW_A
                  servicelist=2
```

- Deny traffic from the internet to the non-secure interface on the firewall and the socks server

```
User:     fwconns cmd=create name="Deny Socks In"
                  desc="Deny Socks request from the Internet"
                  source="The World" destination=NonSecure_FW_A
                  servicelist=34
```

- Allow HTTP traffic between the firewall and the Internet

```
User:     fwconns cmd=create name="Allow HTTP to Internet"
                  desc="Allow HTTP traffic from firewall to Internet"
                  source=NonSecure_FW_A destination="The World"
                  servicelist=501
```

  When using our user-defined service in “Step 1 - Create a Service” on page 429, the service was created to use two filter rules (34 and 73). When we were creating this service we had to assign a backward or forward identifier to the rule. This identifier is used in combination with the source and destination values in the connection. In our case, the connection is defined with a source value of **NonSecure_FW_A_** and a destination value of "The World". When the filter rules are activated, rule 34 will police traffic flowing from the NonSecure_FW_A to the "The World" and rule 73 will police traffic flowing from "The World" to the NonSecure_FW_A. See “fwservice” on page 298 for information on the parameter values.

- Allow the socks server to accept request from CompanyA_dept1 to the internet

```
User:   fwconns cmd=create name="Allow secure Socks"
                desc="Allow CompanyA to use the socks daemon"
                source=CompanyA_dept1 destination="The World"
                sockslist=501
```

Step 3 - Update Filters to Activate Connections

In order to activate the newly defined connections, the filter must be updated. To add our newly-defined filter rule to the existing rules to allow FTP, enter:

```
User:   fwfilter cmd=update
```

Once the fwfilter command completes, the firewall will now permit FTPs and socksified HTTPs from any host within Company A to the internet. To display the filter rules that control this set of communication, enter **fwfilter cmd=list**. See Figure 83 for the displayed filter data.

```
User:   fwfilter cmd=list
System: # This file is generated by Firewall configuration.
        # DO NOT EDIT THIS FILE MANUALLY.
        #
        #       Between NonSecure_FW_A and The World
        #        Service : Allow HTTP to Internet
        # Description : Allow HTTP traffic to flow from firewall to internet
        permit 0.0.0.0 0.0.0.0 10.130.1.70  255.255.255.255 tcp/ack eq 80 gt 1023 non-secure local inbound l=n f=y s=m d=m
        permit 10.130.1.70  255.255.255.255 0.0.0.0 0.0.0.0 tcp gt 1023 eq 80 non-secure local inbound l=n f=y s=m d=m
        #       Between The World and NonSecure_FW_A
        #        Service : Socks deny non-secure
        # Description : Deny Socks from non-secure Adapters
        deny 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp any 0 eq 1080 non-secure both both l=y f=h   s=m d=m
        #       Between LAN_1A and Secure_FW_A
        #        Service : Socks 1/2
        # Description : Permit use of Socks from secure network to the firewall
        permit 10.130.10.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 eq 1080 secure local inbound l=n f=y s=m d=m
        permit 10.130.70.1 255.255.255.255 10.130.10.0 255.255.255.0 tcp/ack eq 1080 gt1023 secure local outbound l=n f=y s=m d=m
        #       Between LAN_2A and Secure_FW_A
        #        Service : Socks 1/2
        # Description : Permit use of Socks from secure network to the firewall
        permit 10.130.20.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 eq 1080 secure local inbound l=n f=y s=m d=m
        permit 10.130.70.1 255.255.255.255 10.130.20.0 255.255.255.0 tcp/ack eq 1080 gt 1023 secure local inbound l=n f=y s=m d=m
```

Figure 83 (Part 1 of 3). Completed Ping, FTP and HTTP Filter Rules

```
#       Between NonSecure_FW_A and The World
#        Service : FTP proxy out 2/2
# Description : Permit FTP outbound from firewall to non-secure network
permit 10.130.1.70 255.255.255.255 0.0.0.0 0.0.0.0 tcp gt 1023 eq 21 non-secure local outbound l=n f=y s=m d=m
permit 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp/ack eq 21 gt 1023 non-secure local inbound l=n f=y s=m d=m
permit 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp eq 20 gt 1023 non-secure local inbound l=n f=y s=m d=m
permit 10.130.1.70 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack gt 1023 eq 20 non-secure local outbound l=n f=y s=m d=m
permit 10.130.1.70 255.255.255.255 0.0.0.0 0.0.0.0 tcp gt 1023 gt 1023 non-secure local outbound l=n f=y s=m d=m
permit 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp/ack gt 1023 gt 1023 non-secure local inbound l=n f=y s=m d=m
#       Between LAN_1A and Secure_FW_A
#        Service : FTP proxy out 1/2
# Description : Permit FTP outbound from secure network to firewall
permit 10.130.10.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 eq 21 secure local inbound l=n f=y s=m d=m
permit 10.130.70.1 255.255.255.255 10.130.10.0 255.255.255.0 tcp/ack eq 21 gt 1023 secure local outbound l=n f=y s=m d=m
permit 10.130.70.1 255.255.255.255 10.130.10.0 255.255.255.0 tcp eq 20 gt 1023 secure local outbound l=n f=y s=m d=m
permit 10.130.10.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp/ack gt 1023 eq 20 secure local inbound l=n f=y s=m d=m
permit 10.130.10.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 gt 1023 secure local inbound l=n f=y s=m d=m
permit 10.130.70.1 255.255.255.255 10.130.10.0 255.255.255.0 tcp/ack gt 1023 gt 1023 secure outbound l=n f=y s=m d=m
#       Between LAN_2A and Secure_FW_A
#        Service : FTP proxy out 1/2
# Description : Permit FTP outbound from secure network to firewall
permit 10.130.20.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 eq 21 secure local inbound l=n f=y s=m d=m
permit 10.130.70.1 255.255.255.255 10.130.20.0 255.255.255.0 tcp/ack eq 21 gt 1023 secure local outbound l=n f=y s=m d=m
permit 10.130.70.1 255.255.255.255 10.130.20.0 255.255.255.0 tcp eq 20 gt 1023 secure local outbound l=n f=y s=m d=m
permit 10.130.20.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp/ack gt 1023 eq 20 secure local inbound l=n f=y s=m d=m
permit 10.130.20.0 255.255.255.0 10.130.70.1 255.255.255.255 tcp gt 1023 gt 1023 secure local inbound l=n f=y s=m d=m
permit 10.130.70.1 255.255.255.255 10.130.20.0 255.255.255.0 tcp/ack gt 1023 gt 1023 secure outbound l=n f=y s=m d=m
#       Between NonSecure_FW_A and The World
#        Service : FTP proxy out 2/2
# Description : Permit FTP outbound from firewall to non-secure network
permit 10.130.1.70 255.255.255.255 0.0.0.0 0.0.0.0 tcp gt 1023 eq 21 non-secure local outbound l=n f=y s=m d=m
permit 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp/ack eq 21 gt 1023 non-secure local inbound l=n f=y s=m d=m
permit 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp eq 20 gt 1023 non-secure local inbound l=n f=y s=m d=m
permit 10.130.1.70 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack gt 1023 eq 20 non-secure local outbound l=n f=y s=m d=m
permit 10.130.1.70 255.255.255.255 0.0.0.0 0.0.0.0 tcp gt 1023 gt 1023 non-secure local outbound l=n f=y s=m d=m
permit 0.0.0.0 0.0.0.0 10.130.1.70 255.255.255.255 tcp/ack gt 1023 gt 1023 non-secure local inbound l=n f=y s=m d=m
#       Between LAN_1A and The World
#        Service : Ping
# Description : Permit Ping outbound secure network to anywhere
permit 10.130.10.0 255.255.255.0 0.0.0.0 0.0.0.0 icmp eq 8 eq 0 both both both l=n f=y s=m d=m
permit 0.0.0.0 0.0.0.0 10.130.10.0 255.255.255.0 icmp eq 0 eq 0 both both both l=n f=y s=m d=m
```

Figure 83 (Part 2 of 3). Completed Ping, FTP and HTTP Filter Rules

```
#       Between LAN_2A and The World
#        Service : Ping
# Description : Permit Ping outbound secure network to anywhere
permit 10.130.20.0 255.255.255.0 0.0.0.0 0.0.0.0 icmp eq 8 eq 0 both both both l=n f=y s=m d=m
permit 0.0.0.0 0.0.0.0 10.130.20.0 255.255.255.0 icmp eq 0 eq 0 both both both l=n f=y s=m d=m
#       Between LAN_3A and The World
#        Service : Ping
# Description : Permit Ping outbound secure network to anywhere
permit 10.130.30.0 255.255.255.0 0.0.0.0 0.0.0.0 icmp eq 8 eq 0 both both both l=n f=y s=m d=m
permit 0.0.0.0 0.0.0.0 10.130.30.0 255.255.255.0 icmp eq 0 eq 0 both both both l=n f=y s=m d=m
```

Figure 83 (Part 3 of 3). Completed Ping, FTP and HTTP Filter Rules

Appendix B. IP Security Example (Using Manual Tunnels)

The following example shows how the IPSec support in the OS/390 Firewall Technologies can be used to establish a security association between two communicating systems. In this scenario, we are assuming that there are two systems that want to communicate securely using IPSec.

- The local system is an OS/390 system containing the OS/390 Firewall Technologies support.
- The remote system can be any system that supports the IPSec protocol standards, described in Request for Comments (RFC) 2401 through 2406 and 2410.

In this example, the S/390 host (tunnel owner) has one network interface at address x.x.x.x and the remote host (tunnel partner) has an address of y.y.y.y, and:

- The tunnel will operate in transport mode
- The data sent in the tunnel will be encrypted using the triple DES algorithm
- All traffic that flows between the two systems will be sent via the tunnel

1. First, use the **fwnwobj** command to add a host object to represent the local S/390 system:

```
fwnwobj cmd=add name=localhost desc="S/390 Local Host"
        type=host addr=x.x.x.x mask=255.255.255.255
```

2. Next, use the **fwnwobj** command to add a host object to represent the remote system that will be connected through the tunnel:

```
fwnwobj cmd=add name=remotehost desc="Remote Host"
        type=host addr=y.y.y.y mask=255.255.255.255
```

3. Next, use the **fwtunnl** command to add a tunnel object to specify all the operational characteristics of the tunnel. This tunnel ID is arbitrarily chosen as 10:

```
fwtunnl cmd=add tunnel=10 type=manual addr=x.x.x.x remaddr=y.y.y.y
        policy=encr encrypthow=3DES_CBC spi=500 mode=transport
```

4. Next, use the **fwfrule** command to add a filter rule to request that all traffic flow through tunnel 10:

```
fwfrule cmd=add name=tunneltraffic desc="Route all traffic through tunnel 10"
        type=permit protocol=all srcopcode=any srcport=0 destopcode=any
        destport=0 interface=nonsecure routing=local direction=both
        tunnel=10
```

5. Next, use the **fwfrule** command to add a filter rule to allow the Encapsulating Security Payload (ESP) protocol to flow between the two systems. This is the protocol used by an encrypted tunnel:

```
fwfrule cmd=add name=ESPtraffic desc="Allow ESP traffic"
        type=permit protocol=ESP srcopcode=any srcport=0 destopcode=any
        destport=0 interface=nonsecure routing=local direction=both
```

6. Next, use the **fwfrule** command to obtain the IDs assigned to the two filter rules you just added:

```
fwfrule cmd=list name=tunneltraffic
fwfrule cmd=list name=ESPtraffic

ASSUME:
   tunneltraffic id returned = 501
   ESPtraffic id returned = 502
```

7. Next, use the **fwservice** command to combine the filters rules you just added into a service. This service describes all of the traffic that is permitted into and out of the local system.

```
fwservice cmd=create name=alltrafficservice desc="All traffic"
          rulelist=502/f,502/b,501/f,501/b
```

© Copyright IBM Corp. 1997, 2000

Note: The order of the filter rules in the rule list is important. Filter rules associated with the IPSec protocols are always placed at the beginning of the list. In our example, this is rule 502.

8. Next, use the **fwservice** command to obtain the ID assigned to the service you just created:

```
fwservice cmd=list name=alltrafficservice

ASSUME:
   alltrafficservice id returned = 503
```

9. Next, use the **fwconns** command to create a connection to associate the local and remote host objects with the filter rule service:

```
fwconns cmd=create name=alltrafficconnection
        desc="Connect all traffic service with local and remote hosts"
        source=localhost destination=remotehost servicelist=503
```

10. Next, use the **fwtunnl** command to export the tunnel definition into files that contain information needed by the remote host:

```
fwtunnl cmd=export tunnel=10 directory=/tmp
```

    - Two files are created by the export process: **fwexpmctx.manual** and **fwexppolicy**. A detailed explanation of the file formats is contained in Appendix F, "FWTUNNL Export File Formats" on page 457.
    - If the remote host is running the OS/390 Firewall Technologies code, then these files can simply be imported using the **fwtunnl** command. For any other IPSec product, the data from the exported files must be converted to the appropriate format. Refer to Appendix F, "FWTUNNL Export File Formats" on page 457 for information to convert the files.

11. Next, use the **fwtunnl** command to activate the tunnel:

```
fwtunnl cmd=activate tunnel=10
```

12. Finally, use the **fwfilter** command to activate the filter rules that you created:

```
fwfilter cmd=update
```

Once the remote host completes its filter and tunnel activation, all the data flowing between the two hosts is encrypted using the triple DES algorithm.

Appendix C. Dynamic Virtual Private Networking (VPN) Tunnel (ISAKMP) Examples

Branch Office Network Using Pre-Shared Key Authentication

The following example shows how the OS/390 Firewall Technologies Dynamic Virtual Private Networking support can be used to establish a security association between two communicating systems. In this scenario, a user on a subnet in a branch office of a company wants to communicate with a user on a different subnet at the company's corporate headquarters.

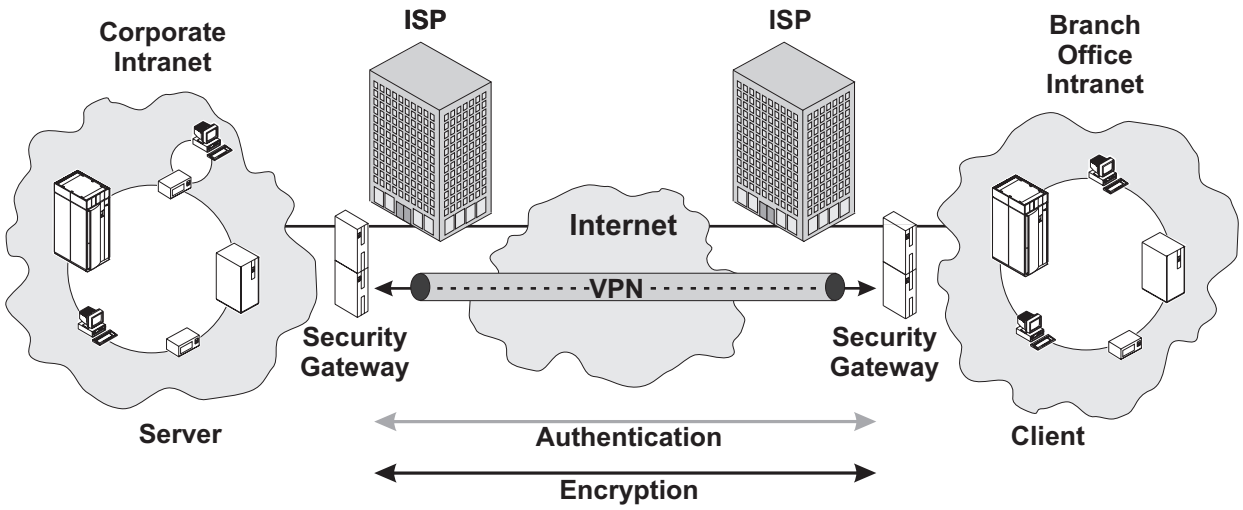


Figure 84. Branch Office Connection Network

In this example, the branch office user (tunnel owner) has a network interface at address 10.20.30.40 and the corporate user (tunnel partner) network interface has an address of 50.60.70.80. Other tunnel set up includes:

- The tunnel operates in tunnel mode.
- Key management negotiation will use either SHA/3DES_CBC or MD5/DES_CBC_8 algorithms depending on the key transform selected during negotiation by the key servers.
- Key management negotiation will use a pre-shared key to authenticate the key servers.
- The data proposal offered during data management negotiation includes HMAC_MD5 for authentication and DES_CBC_8 for encryption.
- Either end of the tunnel may initiate the connection.
- The connection will expire after 1440 minutes.
- All traffic that flows between the two systems will be sent through the tunnel.

This Virtual Private Network requires configuration information on both ends of the connection. In this example, the key policy, data policy, and dynamic tunnel configuration information is the same on both ends of the connection.

Configuration objects must be defined before they can be referenced in other objects. The user can perform configuration steps in whatever order they desire as long as they adhere to this requirement.

© Copyright IBM Corp. 1997, 2000

Configuration may be done through the command interface or by using the Configuration Client Graphical User Interface.

Key Policy: Two key transforms are defined in this example. One of the key transforms used was shipped as a pre-defined key transform in the product. The pre-defined key transform that will be used is:

```
9       SharedKey/MD5/DES_CBC_8/GROUP1/default
```

The second key transform used in this example must be defined. Use the **fwkeytran** command to define and then list this key transform.

```
User:     fwkeytran cmd=add name=BOCorpKeyTran \
          desc="Branch Office/Corporate Key Transform" prot=IKE authmeth=sharedkey \
          hashalg=SHA encralg=3DES_CBC dhgrp=GROUP1 itime=480 isize=0 rtime=60-480 \
          rsize=0

User:     fwkeytran cmd=list keytran=BOCorpKeyTran
System:                 id = 501
                      name = BOCorpKeyTran
                      desc = Branch Office/Corporate Key Transform
                      prot = IKE
                  authmeth = SHAREDKEY
                   hashalg = SHA
                   encralg = 3DES_CBC
                     dhgrp = GROUP1
                     itime = 480
                     isize = 0
                     rtime = 60-480
                     rsize = 0
```

The key transforms are grouped together in a key proposal using the **fwkeyprop** command. The order in which the key transforms are listed determines how they will be proposed during negotiation. The first key transform which both key servers accept will be used.

```
User:     fwkeyprop cmd=create name=BOCorpKeyProp \
          desc="Branch Office/Corporate Key Proposal" \
          keytranlist=BOCorpKeyTran,SharedKey/MD5/DES_CBC_8/GROUP1/default
```

The key proposal is specified in a key policy. Also specified in the key policy is whether main mode or aggressive mode will be used by the key servers during key negotiation. This is done through the **fwkeypol** command.

```
User:     fwkeypol cmd=add name=BOCorpKeyPol \
          desc="Branch Office/Corporate Key Policy" \
          imode=main rmode=main keyprop=BOCorpKeyProp
```

Data Policy: Two transforms are defined in this example; one is an AH transform and one is an ESP transform. Use the **fwahtran** command to create the AH transform.

```
User:     fwahtran cmd=add name=BOCorpAHTran \
          desc="Branch Office/Corporate AH Transform" \
          mode=tunnel authalg=HMAC_MD5 itime=480 isize=0 rtime=60-480 rsize=0
```

Use the **fwesptran** command to define the ESP transform.

```
User:     fwesptran cmd=add name=BOCorpESPTran \
          desc="Branch Office/Corporate ESP Transform" mode=tunnel encralg=DES_CBC_8 \
          itime=240 isize=0 rtime=60-240 rsize=0
```

The AH and ESP transforms are grouped together in a data proposal using the **fwdataprop** command.

```
User:     fwdataprop cmd=create name=BOCorpDataProp \
          desc="Branch Office/Corporate Data Proposal" ahtranlist=BOCorpAHTran \
          esptranlist=BOCorpESPTran
```

The data proposal is combined with information relative to use of Perfect Forward Secrecy in generating keying material to create a data policy. This is done through the **fwdatapol** command.

```
User:     fwdatapol cmd=create name=BOCorpDataPol \
          desc="Branch Office/Corporate Data Policy" pfs=GROUP1 \
          dataproplist=BOCorpDataProp
```

Dynamic Tunnel Policy: The Dynamic Tunnel policy is used to associate specific information pertaining to the tunnel with the anchor rules that govern traffic through the tunnel. The dynamic tunnel defines the data policy, initiation role, and lifetime of the connections. Use the **fwdyntun** command to define a Dynamic Tunnel object.

```
User:     fwdyntun cmd=add name=BOCorpDynTun \
          desc="Branch Office/Corporate Dynamic Tunnel" datapol=BOCorpDataPol init=either \
          connlifetime=1440
```

Configure Key Servers: After the key policy is defined, it needs to be associated with the Key Servers (ISAKMP daemons). Key servers are defined through the **fwkeysrv** command. A group of remote key servers is associated with the local key server and a key policy through the **fwkeysrvgrp** command. As part of Virtual Private Networking, the key servers are required to authenticate themselves. The information required to permit this authentication to occur in this example is provided through the **fwauthinfo** command.

In this example, the local key server is shown as the Branch Office Gateway and the remote key server is shown as the Corporate Gateway.

Key Servers: Two key servers will be defined in this example: the local key server and the remote key server. Use the **fwkeysrv** command to define the key servers.

```
User:     fwkeysrv cmd=add name=BOGatewayKS desc="Branch Office Gateway Key Server" \
          idtype=IPV4 authid=10.20.30.40 ipaddr=10.20.30.40

User:     fwkeysrv cmd=add name=CorpGatewayKS desc="Corporate Gateway Key Server" \
          idtype=FQDN authid=gateway.mybiz.com ipaddr=50.60.70.80
```

Next, a key server group must be defined that associates the Branch Office Gateway Key Server and the Corporate Gateway Key Server with the key policy that will be used between them. Use the **fwkeysrvgrp** command to create the key server group.

```
User:     fwkeysrvgrp cmd=create name=BOKSG desc="Branch Office Key Server Group" \
          keypol=BOCorpKeyPol lockeysrv=BOGatewayKS remkeysrvlist=CorpGatewayKS
```

The key server group on the Corporate machine would use the following definition.

```
User:     fwkeysrvgrp cmd=create name=CorpKSG desc="Corporate Key Server Group" \
          keypol=BOCorpKeyPol lockeysrv=CorpGatewayKS remkeysrvlist=BOGatewayKS
```

Authentication Data: Because the key transforms defined in the key policy were set up for pre-shared key authentication, additional authentication information is required. An authentication information object must be defined which includes specific information relative to the shared key.

To define the authentication information, use the **fwauthinfo** command.

```
User:   fwauthinfo cmd=add name=BOAuthInfo \
        desc="Branch Office Authentication Information" remkeysrv=CorpGatewayKS \
        shkey=4142434445464748
```

The authentication information on the Corporate machine would also need to be defined.

```
User:   fwauthinfo cmd=add name=CorpAuthInfo \
        desc="Corporate Authentication Information" remkeysrv=BOGatewayKS \
        shkey=4142434445464748
```

Define Anchor Rules: After the policy and key server information is defined, it needs to be associated with objects that allow dynamic connections to be created. These objects are anchor filter rules. The anchor filter rule is a special type of filter rule that serves as a placeholder for the dynamic filter rules in the ordered list of all manual and anchor filter rules. To set up the dynamic filter rules, the **fwfrule**, **fwservice**, **fwnwobj**, and **fwconns** commands will be used.

First the anchor rule will be set up using the **fwfrule** command. This rule would be defined on both the Branch Office and the Corporate machines.

```
User:   fwfrule cmd=add name=BOCorpAnchor \
        desc="Branch Office/Corporate Anchor Rule" \
        type=anchor protocol=all srcopcode=any srcport=0 destopcode=any destport=0 \
        interface=nonsecure routing=local log=no tunnel=BOCorpDynTun

User:   fwfrule cmd=list name=BOCorpAnchor
System:                id = 501
                     type = anchor
                     name = BOCorpAnchor
                     desc = Branch Office/Corporate Anchor Rule
                 protocol = all
                srcopcode = any
                  srcport = 0
               destopcode = any
                 destport = 0
                interface = nonsecure
                  routing = local
                      log = no
                   tunnel = 501

User:   fwfrule cmd=add name=BOCorpSecure \
        desc="Branch Office/Corporate Rule for Secure Interface" \
        type=permit protocol=all srcopcode=any srcport=0 destopcode=any destport=0 interface=secure routing=route direction=both \

User:   fwfrule cmd=list name=BOCorpSecure

System:                id = 502
                     type = permit
                     name = BOCorpSecure
                     desc = Branch Office/Corporate Rule for Secure Interface
                 protocol = all
                srcopcode = any
                  srcport = 0
               destopcode = any
                 destport = 0
                interface = secure
                  routine = route
                direction = both
                      log =
                   tunnel =
                 fragment =
```

Several other filter rules that are required are shipped as pre-defined rules with OS/390 Firewall Technologies. Issue **fwfrule cmd=list** to see the rules that are pre-defined. The rules that will be used include:

```
136   Permit ISAKMPD UDP Non-Secure
137   Permit ISAKMPD UDP Secure
5     VPN Authentication
11    VPN Encryption
```

Next, two services will be defined using the **fwservice** command. One service establishes the anchor rule between the connection endpoints (BOCorpAnchorSvc). The other service establishes the permit rules for the key server (ISAKMP server) traffic (BOCorp ISAKMPSvc).

These rules would be defined on both the Branch Office and the Corporate machines.

```
User:     fwservice cmd=create name=BOCorpSvc \
          desc="Branch Office/Corporate Service" \
          rulelist=501/f,501/b,502/f,502/b

User:     fwservice cmd=create name=BOCorpISAKMPSvc \
          desc="Branch Office/Corporate ISAKMPD Service" \
          rulelist=136/f,136/b,5/f,5/b,11/f,11/b
```

The objects which define the tunnel endpoints, data connection endpoints, and users on the subnet will be defined using the **fwnwobj** command.

These rules would be defined on both the Branch Office and the Corporate machines.

```
User:     fwnwobj cmd=add name=BOGateway desc="Branch Office Gateway" type=VPN \
          addr=10.20.30.40 mask=255.255.255.255

User:     fwnwobj cmd=add name=CorpGateway desc="Corporate Gateway" type=VPN \
          addr=50.60.70.80 mask=255.255.255.255

User:     fwnwobj cmd=add name=BOSubnet desc="Branch Office Subnet" type=Network \
          addr=1.2.3.0 mask=255.255.255.0

User:     fwnwobj cmd=add name=CorpSubnet desc="Corporate Subnet" type=Network \
          addr=5.6.7.0 mask=255.255.255.0

User:     fwnwobj cmd=add name=BOUser desc="Branch Office User" type=Host \
          addr=1.2.3.4 mask=255.255.255.255

User:     fwnwobj cmd=add name=CorpUser desc="Corporate User" type=Host \
          addr=5.6.7.8 mask=255.255.255.255
```

Next, two connections will be defined which set up the anchor filter rules for the two connection subnets and key servers. The source and destination would be in the reverse order on the corporate machine. The source is usually the machine or it's clients.

```
User:     fwconns cmd=create name=BOCorpISAKMP desc="Branch Office/Corporate ISAKMP" \
          source=BOGateway destination=CorpGateway servicelist=502

User:     fwconns cmd=create name=BOCorpAnchor desc="Branch Office/Corporate Anchor" \
          source=BOSubnet destination=CorpSubnet servicelist=501
```

These connections must be ordered so that the BOCorpISAKMP connection is before the BOCorpAnchor connection. Use the **fwconns** command to determine the order of the connections.

```
   User:    fwconns cmd=list
System:     502   BOCorpAnchor              Branch Office/Corporate Anchor
            501   BOCorpISAKMP              Branch Office/Corporate ISAKMP

   User:    fwconns cmd=move connection=BOCorpAnchor after=BOCorpISAKMP

   User:    fwconns cmd=list
System:     501   BOCorpISAKMP              Branch Office/Corporate ISAKMP
            502   BOCorpAnchor              Branch Office/Corporate Anchor
```

These connections are generated using the **fwfilter** command on the Branch Office machine. This result in activated manual and anchor filter rules.

```
   User:    fwfilter cmd=update
System:     ICAC1577i Processing firewall TCP/IP stack CS390IPB:
            Filter support (level 2.10.0) initialized at 09:32:50 on May-25-2000
```

The following connections would be defined on the Corporate machine and would also need to be ordered and activated as above.

```
   User:    fwconns cmd=create name=CorpBOAnchor desc="Corporate/Branch Office Anchor" \
            source=CorpSubnet destination=BOSubnet servicelist=501

   User:    fwconns cmd=create name=CorpBOISAKMP desc="Corporate/Branch Office ISAKMP" \
            source=CorpGateway destination=BOGateway servicelist=502
```

Dynamic Connection: Once the anchor filters have been generated and the appropriate key server group has been defined, the locally activated dynamic connections can be defined and activated. If a connection will be remotely activated, no further configuration steps are required.

In this example, the connection may be activated at either the local or remote endpoints. The dynamic connection definition will be shown such that the Branch Office is the point where the dynamic connections will be activated. The **fwdynconns** command is used.

The dynamic connection is defined using the **fwdynconns** command.

```
   User:    fwdynconns cmd=add name=BOCorpDynCon \
            desc="Branch Office/Corporate Dynamic Connection" src=BOUser dest=CorpUser \
            srcport=0 destport=0 prot=all remkeysrv=CorpGatewayKS keysrvgrp=BOKSG
```

To activate this dynamic connection, issue:

```
   User:    fwdynconns cmd=activate dynconnlist=BOCorpDynCon
System:     ICAC1756i Processing dynconn: BOCorpDynCon
```

After a brief period of time, the active connection may be listed by using the **fwdynconns** command.

```
   User:    fwdynconns cmd=listactive
System:     3          10.20.30.40    50.60.70.80     active  user
```

Additional information may also be viewed.

```
   User:     fwdynconns cmd=listactive tunlist=3 format=long
System:              TunnelID = 3
                       TunSrc = 10.20.30.40
                      TunDest = 50.60.70.80
                        State = active
                    Activator = user
                     SrcAddr1 = 1.2.3.4
                     SrcAddr2 = 255.255.255.255
                    DestAddr1 = 5.6.7.8
                    DestAddr2 = 255.255.255.255
            Tunnel expires at = Tue May 25 13:33:18 1999 EDT
            Bytes till expire = 0
                AH Encap Mode = tunnel
               ESP Encap Mode = tunnel
                  AH Auth Alg = HMAC_MD5
                  Encrypt Alg = DES_CBC_8
                 ESP Auth Alg =
                       ConnID = 502
                    SrcObjID = 503
                   DestObjID = 504
                       SvcID = 501
                     FruleID = 501
                    DynTunID = 501
```

To deactivate the connection, issue:

```
   User:     fwdynconns cmd=deactivate tunlist=3
```

Branch Office Network Using RSA Signature Authentication

The following example shows how the OS/390 Firewall Technologies Dynamic Virtual Private Networking support can be used to establish a security association between two communicating systems. In this scenario, a user on a subnet in a branch office of a company wants to communicate with a user on a different subnet at the company's corporate headquarters.

Tunnel set up includes:

- The tunnel operates in tunnel mode.
- Key management negotiation will use either SHA/3DES_CBC or MD5/DES_CBC_8 algorithms depending on the key transform selected during negotiation by the key servers.
- Key management negotiation will use RSA Signature mode to authenticate the key servers.
- The data proposal offered during data management negotiation includes HMAC_MD5 for authentication and DES_CBC_8 for encryption.
- Either end of the tunnel may initiate the connection.
- The connection will expire after 1440 minutes.
- All traffic that flows between the two systems will be sent through the tunnel.

This Virtual Private Network requires configuration information on both ends of the connection. In this example, the key policy, data policy, and dynamic tunnel configuration information is the same on both ends of the connection.

This example sets up a VPN connection between a user in a Branch Office and a user in the Corporate intranet. The ordering of the commands used here is the recommended order to configure the VPN. Configuration objects must be defined before they can be referenced in other objects. However, the configuration steps can be done in whatever order is desired as long as they adhere to the above requirement.

Key Policy: Two key transforms are defined in this example. One of the key transforms used was shipped as a predefined key transform in the product. Use the **fwkeytran** command to list the shipped transforms.

The predefined key transform desired is RSASIG/MD5/DES_CBC_8/GROUP1/default which is referred to as key transform 1 in the predefined list.

The second key transform used in this example must be defined. Use the **fwkeytran** command to define this key transform. List the defined key transform.

```
User:   fwkeytran cmd=add name=BOCorpKeyTran \
        desc="Branch Office/Corporate Key Transform" prot=IKE authmeth=rsasig \
        hashalg=SHA encralg=3DES_CBC dhgrp=GROUP1 itime=480 isize=0 rtime=60-480 \
        rsize=0

User:   fwkeytran cmd=list keytran=BOCorpKeyTran
System:
                id = 501
              name = BOCorpKeyTran
              desc = Branch Office/Corporate Key Transform
              prot = IKE
          authmeth = RSASIG
           hashalg = SHA
           encralg = 3DES_CBC
             dhgrp = GROUP1
             itime = 480
             isize = 0
             rtime = 60-480
             rsize = 0
```

The key transforms are grouped together in a key proposal using the **fwkeyprop** command. The order in which the key transforms are listed determines how they will be proposed during negotiation. The first key transform which both key servers accept will be used.

```
User:   fwkeyprop cmd=create name=BOCorpKeyProp \
        desc="Branch Office/Corporate Key Proposal" \
        keytranlist=BOCorpKeyTran,RSASIG/MD5/DES_CBC_8/GROUP1/default
        RSA
```

The key proposal is specified in a key policy. Also specified in the key policy is whether main mode or aggressive mode will be used by the key servers during key negotiation. This is done through the **fwkeypol** command.

```
User:   fwkeypol cmd=add name=BOCorpKeyPol desc="Branch Office/Corporate Key Policy" \
        imode=main rmode=main keyprop=BOCorpKeyProp
```

Data Policy: Two transforms are defined in this example; one is an AH transform and one is an ESP transform. Use the **fwahtran** command to create the AH transform.

```
User:   fwahtran cmd=add name=BOCorpAHTran desc="Branch Office/Corporate AH Transform" \
        mode=tunnel authalg=HMAC_MD5 itime=480 isize=0 rtime=60-480 rsize=0
```

Use the **fwesptran** command to define the ESP transform.

```
User:   fwesptran cmd=add name=BOCorpESPTran \
        desc="Branch Office/Corporate ESP Transform" mode=tunnel encralg=DES_CBC_8 \
        itime=240 isize=0 rtime=60-240 rsize=0
```

The AH and ESP transforms are grouped together in a data proposal using the **fwdataprop** command.

```
User:   fwdataprop cmd=create name=BOCorpDataProp \
        desc="Branch Office/Corporate Data Proposal" ahtranlist=BOCorpAHTran \
        esptranlist=BOCorpESPTran
```

The data proposal is combined with information relative to use of Perfect Forward Secrecy in generating keying material to create a data policy. This is done through the **fwdatapol** command.

```
User:   fwdatapol cmd=create name=BOCorpDataPol \
        desc="Branch Office/Corporate Data Policy" pfs=GROUP1 \
        dataproplist=BOCorpDataProp
```

Dynamic Tunnel Policy: The Dynamic Tunnel policy is used to associate specific information pertaining to the tunnel with the anchor filter rules that govern traffic through the tunnel. The dynamic tunnel defines the data policy, initiation role, and lifetime of the connections. Use the **fwdyntun** command to define a Dynamic Tunnel object.

```
User:   fwdyntun cmd=add name=BOCorpDynTun \
        desc="Branch Office/Corporate Dynamic Tunnel" datapol=BOCorpDataPol init=either \
        connlifetime=1440
```

Configure Key Servers: After the key policy is defined, it needs to be associated with the Key Servers (ISAKMP daemons). Key servers are defined through the **fwkeysrv** command. A group of remote key servers is associated with the local key server and a key policy through the **fwkeysrvgrp** command. As part of Virtual Private Networking, the key servers are required to authenticate themselves to their peer. The information required to permit this authentication to occur in this example is provided through the **fwauthinfo** command.

In this example, the local key server is shown as the Branch Office Gateway and the remote key server is shown as the Corporate Gateway.

Key Servers: Two key servers will be defined in this example: the local key server and the remote key server. Use the **fwkeysrv** command to define the key servers.

```
User:   fwkeysrv cmd=add name=BOGatewayKS desc="Branch Office Gateway Key Server" \
        idtype=IPV4 authid=10.20.30.40 ipaddr=10.20.30.40
User:   fwkeysrv cmd=add name=CorpGatewayKS desc="Corporate Gateway Key Server" \
        idtype=IPV4 authid=50.60.70.80 ipaddr=50.60.70.80
```

Next, a key server group must be defined that associates the Branch Office Gateway Key Server and the Corporate Gateway Key Server with the key policy that will be used between them. Use the **fwkeysrvgrp** command to create the key server group.

```
User:   fwkeysrvgrp cmd=create name=BOKSG desc="Branch Office Key Server Group" \
        keypol=BOCorpKeyPol lockeysrv=BOGatewayKS remkeysrvlist=CorpGatewayKS

        The key server group on the Corporate machine would use the
        following definition.

User:   fwkeysrvgrp cmd=create name=CorpKSG desc="Corporate Key Server Group" \
        keypol=BOCorpKeyPol lockeysrv=CorpGatewayKS remkeysrvlist=BOGatewayKS
```

Authentication Data: Since the key transforms defined in the key policy were set up for RSA Signature mode authentication, additional authentication information must be specified. An authentication information object should be defined which includes specific information relative to the certificate authority (CA) to be used by the key server's peer in selecting their certificate to use in the negotiation. This example will set up both the Branch Office and Corporate Gateways to use an Entrust CA.

Note: The local key server certificates and any CA certificates that are required to negotiate with the

remote key server must be located in RACF on the key ring specified by the **fwkeyring** command. The certificates are added to RACF using the RACF **RACDCERT** command. The certificates are also added to the key ring using the same command.

The two key servers will specify the certificate authority that will be used as their certificate issuers. The first step in doing this is to use the **fwcertauth** command.

```
User:    fwcertauth cmd=add name=EntrustCertAuth desc="Entrust Certificate Authority" \
         label="Entrust CA Cert"
```

Then define the authentication information object, to use this Certificate Authority. To support certificate authentication, both of the key servers must have certificates and certificate authority certificates added to RACF. When adding these certificates a label is used to reference each certificate. That label is used to tie the key server to that certificate authority's certificate in RACF. This is done using the **fwauthinfo** command.

```
User:    fwauthinfo cmd=add name=BOAuthInfo \
         desc="Branch Office Authentication Information" remkeysrv=CorpGatewayKS \
         certauth=EntrustCertAuth
```

The authentication information on the Corporate machine would also need to be defined.

```
User:    fwauthinfo cmd=add name=CorpAuthInfo \
         desc="Corporate Authentication Information" remkeysrv=BOGatewayKS \
         certauth=EntrustCertAuth
```

Define Anchor Rules: After the policy and key server information is defined, it needs to be associated with objects that allow dynamic connections to be created. These objects are anchor filter rules The anchor filter rule is a special type of filter rule that serves as a placeholder for the dynamic filter rules in the ordered list of all manual and anchor filter rules. To set up the dynamic filter rules, the **fwfrule**, **fwservice**, **fwnwobj**, and **fwconns** commands will be used.

First the anchor rule will be set up using the **fwfrule** command. This rule would be defined on both the Branch Office and the Corporate machines.

```
   User:     fwfrule cmd=add name=BOCorpAnchor desc="Branch Office/Corporate Anchor Rule" \
             type=anchor protocol=all srcopcode=any srcport=0 destopcode=any destport=0 \
             interface=both routing=both log=no tunnel=BOCorpDynTun
   User:     fwfrule cmd=list name=BOCorpAnchor
 System:
                        id = 501
                      type = anchor
                      name = BOCorpAnchor
                      desc = Branch Office/Corporate Anchor Rule
                  protocol = all
                 srcopcode = any
                   srcport = 0
                destopcode = any
                  destport = 0
                 interface = both
                   routing = both
                       log = no
                    tunnel = 503

   User:     fwfrule cmd=add name=BOCorpSecure \
             desc="Branch Office/Corporate Rule for Secure Interface" \
             type=permit protocol=all srcopcode=any srcport=0 destopcode=any destport=0 interface=secure routing=route direction=both \

   User:     fwfrule cmd=list name=BOCorpSecure

 System:                 id = 502
                       type = permit
                       name = BOCorpSecure
                       desc = Branch Office/Corporate Rule for Secure Interface
                   protocol = all
                  srcopcode = any
                    srcport = 0
                 destopcode = any
                   destport = 0
                  interface = secure
                    routine = route
                  direction = both
                        log =
                     tunnel =
                   fragment =
```

Several other filter rules that are required are shipped as pre-defined rules with OS/390 Firewall Technologies. Issue **fwfrule cmd=list** to see the rules that are pre-defined. The rules that will be used include:

```
136   Permit ISAKMPD UDP Non-Secure
5     VPN Authentication
11    VPN Encryption
```

Next, two services will be defined using the **fwservice** command. One service establishes the anchor filter rule between the connection endpoints (BOCorpAnchorSvc). The other service establishes the permit filter rules for the key server (ISAKMP daemon) traffic (BOCorpISAKMPSvc).

These rules would be defined on both the Branch Office and the Corporate machines.

```
   User:     fwservice cmd=create name=BOCorpAnchorSvc \
             desc="Branch Office/Corporate Anchor Service" rulelist=501/f
   User:     fwservice cmd=create name=BOCorpISAKMPSvc \
             desc="Branch Office/Corporate ISAKMPD Service" \
             rulelist=136/f,136/b,5/f,5/b,11/f,11/b
```

The objects which define the tunnel endpoints, data connection endpoints, and users on the subnet will be defined using the **fwnwobj** command.

These rules would be defined on both the Branch Office and the Corporate machines.

```
User:   fwnwobj cmd=add name=BOGateway desc="Branch Office Gateway" type=VPN \
        addr=10.20.30.40 mask=255.255.255.255

User:   fwnwobj cmd=add name=CorpGateway desc="Corporate Gateway" type=VPN \
        addr=50.60.70.80 mask=255.255.255.255

User:   fwnwobj cmd=add name=BOSubnet desc="Branch Office Subnet" type=Network \
        addr=1.2.3.0 mask=255.255.255.0

User:   fwnwobj cmd=add name=CorpSubnet desc="Corporate Subnet" type=Network \
        addr=5.6.7.0 mask=255.255.255.0

User:   fwnwobj cmd=add name=BOUser desc="Branch Office User" type=Host \
        addr=1.2.3.4 mask=255.255.255.255

User:   fwnwobj cmd=add name=CorpUser desc="Corporate User" type=Host \
        addr=5.6.7.8 mask=255.255.255.255
```

Next, two connections will be defined which set up the anchor filter rules for the two connection subnets and key servers.

```
User:   fwconns cmd=create name=BOCorpAnchor desc="Branch Office/Corporate ISAKMP" \
        source=BOSubnet destination=CorpSubnet servicelist=501

User:   fwconns cmd=create name=BOCorpISAKMP desc="Branch Office/Corporate ISAKMP" \
        source=BOGateway destination=CorpGateway servicelist=502
```

These connections must be ordered so that the BOCorpISAKMP connection is before the BOCorpAnchor connection. Use the **fwconns** command to determine the order of the connections.

```
  User:   fwconns cmd=list
System:
        502   BOCorpAnchor              Branch Office/Corporate Anchor
        501   BOCorpISAKMP              Branch Office/Corporate ISAKMP

  User:   fwconns cmd=move connection=BOCorpAnchor after=BOCorpISAKMP

  User:   fwconns cmd=list
System:
        501   BOCorpAnchor              Branch Office/Corporate Anchor
        502   BOCorpISAKMP              Branch Office/Corporate ISAKMP
```

These connections must be activated using the **fwfilter** command on the Branch Office machine. This results in activated manual and anchor filter rules.

```
  User:   fwfilter cmd=update
System:   ICAC1577i Processing firewall TCP/IP stack CS390IP:
          Filter support (level 2.10.0) initialized at 11:10:35 on Apr-18-2000
```

The following connections would be defined on the Corporate machine and would also need to be ordered and activated as above.

```
User:   fwconns cmd=create name=CorpBOISAKMP desc="Corporate/Branch Office ISAKMP" \
        source=CorpGateway destination=BOGateway servicelist=501
User:   fwconns cmd=create name=CorpBOAnchor desc="Corporate/Branch Office Anchor" \
        source=CorpSubnet destination=BOSubnet servicelist=502
```

Dynamic Connection: Once the anchor filters have been generated and the appropriate key server group has been defined, the locally activated dynamic connections can be defined and activated. If a connection will be remotely activated, no further configuration steps are required.

In this example, the connection may be activated at either the local or remote endpoints. The dynamic connection definition will be shown such that the Branch Office is the point where the dynamic connections will be activated. The **fwdynconns** command is used.

The dynamic connection is defined using the **fwdynconns** command.

```
  User:  fwdynconns cmd=add name=BOCorpDynCon \
         desc="Branch Office/Corporate Dynamic Connection" src=BOUser dest=CorpUser \
         srcport=0 destport=0 prot=all remkeysrv=CorpGatewayKS
```

To activate this dynamic connection, issue:

```
  User:  fwdynconns cmd=activate dynconnlist=BOCorpDynCon
System:  ICAC1756i Processing dynconn: BOCorpDynCon
```

After a brief period of time, the active connection may be listed by using the **fwdynconns** command.

```
  User:  fwdynconns cmd=listactive
System:  1          10.20.30.40     50.60.70.80     active  user
```

Additional information may also be viewed.

```
  User:  fwdynconns cmd=listactive tunlist=1 format=long
System:
                 TunnelID = 1
                   TunSrc = 10.20.30.40
                  TunDest = 50.60.70.80
                    State = active
                Activator = user
                 SrcAddr1 = 1.2.3.4
                 SrcAddr2 = 255.255.255.255
                DestAddr1 = 5.6.7.8
                DestAddr2 = 255.255.255.255
        Tunnel expires at = Wed May 26 15:11:17 1999 EDT
        Bytes till expire = 0
            AH Encap Mode = tunnel
           ESP Encap Mode = tunnel
              AH Auth Alg = HMAC_MD5
              Encrypt Alg = DES_CBC_8
             ESP Auth Alg =
                   ConnID = 502
                SrcObjID = 503
               DestObjID = 504
                   SvcID = 501
                 FruleID = 501
                DynTunID = 501
```

To deactivate the connection, issue:

```
  User:  fwdynconns cmd=deactivate tunlist=1
```

Appendix D. Alternative DNS Configuration

The following is an alternative to the recommended DNS configuration discussed in “The Recommended Configuration” on page 74. It can be accomplished by modifying the configuration files generated by the **fwdns** configuration command..

Configuration Without a Secure Name Server

If you do not have a name server in the secure network, set up your configuration as follows:

1. Specify the secure adapter using **fwdns secaddr=**:

   ```
   fwdns cmd=change
         secdom=secure.company.com
         secaddr=sa.sa.sa.sa
         remaddr=nnn.nnn.nnn.nnn
   ```

2. Enable the DNS to provide reverse name resolution for the secure adapter. In this example, the secure adapter is named **firewall.secure.company.com** at 199.2.3.27. To do this, follow these steps:

   a. Add a line to the boot file for the reverse file:

      ```
      primary 3.2.199.in-addr.arpa    /etc/fwnamed.rev
      ```

   b. Create the reverse file by copying **/etc/fwnamed.loc** to **/etc/fwnamed.rev** and making the following changes:

      1) Change

         ```
         0.0.127.in-addr.arpa.
         ```

         in the SOA statement to

         ```
         3.2.199.is-addr.arpa.
         ```

      2) Change

         ```
         1 IN PTR localhost.
         ```

         to

         ```
         27 IN PTR firewall
         ```

3. Configure the hosts in the secure network to use the secure adapter as the name server.

4. Ensure that the name and address information for the hosts in the secure network are kept in the **hosts** file on each host. This is because if the addresses were to be defined in the DNS on the firewall, then this information would be visible to users in the non-secure network.

© Copyright IBM Corp. 1997, 2000

Appendix E. Problem Determination

This section tells you how to troubleshoot some of the common problems encountered when setting up and configuring a firewall.

General Problems

FWKERN does not start

Ensure that:

- All of the External Security Manager configuration was completed, including the definition of **/u/fwkern**.
- The Firewall server start-up procedures (JCL) are in a library in the system search path.
- The Firewall daemon code is in an APF authorized data set that is in the system search path.
- **/etc/security/fwdaemon.cfg** exists and each server to be started specifies CONFIGURED=Y.
- **/etc/security/fwstack.cfg** exists and contains the name of each firewall TCP/IP stack.
- The firewall's TCP/IP stack was started with, minimally, IPCONFIG FIREWALL in the TCP/IP profile.
- Any required server configuration was completed (see the sections pertaining to each of the servers for more details).
- The following HFS filesets and any other data sets specified during logging configuration are not full:

  /dev

  /u/fwkern

  /var

- Check SYSOUT, CTOUT, DCTOUT data sets in ICAxxx or **fwkern** job output.

Abend code S069 from Firewall Server

When **fwkern** is cancelled at the operator console, any firewall servers currently active will also be terminated. The termination is not graceful and will result in an abend code of S069 from each server address space. The **fwkern** address space will terminate with an abend code of S222. Cancelling **fwkern** should only be done as a last resort. It is recommended that **fwkern** should be stopped which allows for graceful termination of all firewall server address spaces.

Routing Problems

Cannot ping from secure side to the nonsecure side.

- Ensure that the firewall's TCP/IP stack was started with IPCONFIG FIREWALL DATAGRAMFWD in the TCP/IP profile.
- Ensure that static routing is set properly on the firewall. To verify the static routes on the firewall, use the **netstat home** and **netstat gate** commands. **netstat home** shows the addresses for the links (adapters) defined to TCP/IP and **netstat gate** shows information about the static routing.

  Refer to the *OS/390 IBM Communications Server: IP Diagnosis* for more information.

  You should have an interface route for each interface and your default route should point to the router on the nonsecure side of the firewall.

© Copyright IBM Corp. 1997, 2000

- If you are able to ping the secure interface on the firewall and the nonsecure interface on the firewall but are unable to ping the external router, you are probably missing a return route in the external router. The return route in the external router is a static route to your secure network listing the firewall's nonsecure interface at the gateway. If you are using unregistered addresses in your secure network, you will not be able to setup a return route.
- Ensure that ASORTEDPARMS NOFWD is *not* specified in the TCP/IP profile.

DNS Problems

1. **Receive "ICAC1170e DNS has not been configured yet" error message**.
   - You have not configured the firewall's DNS facilities. Use the **fwdns** command. (See “fwdns” on page 216.)
2. **File permissions do not allow the specified action**.
   - Firewall traffic control is not permitting the DNS packets to flow. Your firewall security policy needs to be set to predefined service 20, “Permit DNS queries” using the **fwsecpolicy** command. (See “fwsecpolicy” on page 296.)
3. **onslookup www.ibm.com nns.nns.nns.nns fails**.
   - The non-secure name server is not using the indicated address or is not configured properly. Contact your DNS service provider for a valid name server address.
4. **onslookup www.ibm.com 127.0.0.1 fails**.
   - **named** may not be running. Issue one of the following commands from a superuser ID (UID=0):
     – If the UNIX services name server was started from a PROC:

     ```
     ps -e | grep NAMED
     ```

     – If the UNIX services name server was started from the UNIX initialization script **/etc/rc**:

     ```
     ps -e | grep named
     ```

     to find it. Start the UNIX services name server and check the **syslog** records.
5. **onslookup host.secure.company.com.sns.sns.sns.sns fails**
   - Your secure name server is down. Restart it.
6. **onslookup www.ibm.com.sns.sns.sns.sns fails**
   - Your secure name server is not configured properly to interact with OS/390 Firewall Technologies. See Chapter 12, “Configuring and Using the Domain Name Service (DNS)” on page 71 for information on how to do this.
7. **Cannot resolve external names**.
   - Make sure there is a forwarders entry and a cache entry in the boot file of the secure name server. Make sure that the cache file points to the firewall as a root name server.
   - Verify that `/etc/named.ca` is set up correctly on the firewall. To verify that the firewall is configured correctly enter:

     User:

     ```
     nslookup (or onslookup) www.ibm.com 127.0.0.1
     ```

     If the http://www.ibm.com cannot be resolved, the `/etc/named.ca` file on the firewall is not configured properly or the routing for the firewall is not configured properly. The cache file should have an NS record and an A record pointing to the name server provided by your Internet Service Provider (ISP).
8. **Cannot resolve internal names from the firewall**.

   Make sure `/etc/resolv.conf` is set up correctly on the firewall. It should have a domain statement specifying the secure domain and a `nameserver` statement specifying the inside name server as its resolver.

Socks Problems

Socks connections do not work.

- External names cannot be resolved. Verify that domain name services is configured properly.
- FTP data commands such as **ls**, **get** or **put** do not work.

  Make sure filter rules allow the data connection to occur.
- Network error. Connection refused.

  Verify that socks rules and connections are set up allow a connection for the source IP address, destination IP address, and port. See “fwconns” on page 195 and “fwsrule” on page 305 for details on how to define rules and connections.

  Make sure that the client points to the firewall as the socks server.

  Make sure filter rules are not denying the TCP connection to the socks server or remote host and that socks rules (**/etc/sockd.conf**) allow the connection from the client to the target.
- Check SYSOUT, CTOUT, SOCKDOUT data sets in ICADSOCK job output.

FTP Problems

Proxy ftp does not work.

- Remote host refused a connect operation.
  - Ensure that your secure and nonsecure adapters are set properly (see “fwadapter” on page 179.)
  - Ensure that the **/etc/security/fwdaemon.cfg** file has a correct entry for `pftpd`.
  - Verify that the user is not in the **/etc/fwftp.deniedusers** file.
  - Ensure that filter rules are not denying the TCP connection.
  - Check SYSOUT, DCTOUT, PFTPOUT data sets in ICADPFTP job output.

Appendix F. FWTUNNL Export File Formats

Use the **fwtunnl** command to export one or more tunnel definitions. The export action creates two files in the requested directory.

Export File format for OS/390 Version 2 Release 6 and Prior Releases

The following files are produced when **fwtunnl cmd=export version=V2R6** is issued. These files contain tunnel definition data representing an OS/390 Version 2 Release 6 and earlier tunnel.

- The first file is known as the context file and is named `fwexpmctx.manual`. This file contains 22 lines (per tunnel definition) and all data must begin in column 1. Unless stated otherwise, all values are decimal. An example of the format and typical contents of the `fwexpmctx.manual` file follows:

```
Typical Contents:                   File Format:
#Tunnel_partner Version = 1.0  Line 1: context line - must be: #Tunnel_partner Version = 1.0
11.130.10.7                         2: IP Address of Tunnel Owner
11.130.10.5                         3: IP Address of Tunnel Partner
1                                   4: Tunnel Id
1000                                5: ESP SPI of Tunnel Partner
1000                                6: AH SPI of Tunnel Partner
682262                              7: ESP SPI of Tunnel Owner
682262                              8: AH SPI of Tunnel Owner
CDMF                                9: Encryption Algorithm of Tunnel Partner (CDMF or DES_CBC_8 or DES_CBC_4)
8                                  10: Size (number of bytes) of Encryption Key of Tunnel Partner
0xce803234ce732f25                 11: Encryption Key of Tunnel Partner (hex value starting with '0x')
CDMF                               12: Encryption Algorithm of Tunnel Owner (CDMF or DES_CBC_8 or DES_CBC_4)
8                                  13: Size (number of bytes) of Encryption Key of Tunnel Owner
0xf2e943e6c8c24fa7                 14: Encryption Key of Tunnel Owner (hex value starting with '0x')
KEYED_MD5                          15: Authentication Algorithm of Tunnel Partner (KEYED_MD5)
16                                 16: Size (number of bytes) of Authentication Key of Tunnel Partner
0x3ba30d2a8278c94b29828ccef98081af 17: Authentication Key of Tunnel Partner (hex value starting with '0x')
KEYED_MD5                          18: Authentication Algorithm of Tunnel Owner (KEYED_MD5)
16                                 19: Size (number of bytes) of Authentication Key of Tunnel Owner
0xc1bbc0b0105bf88387f8708ca2585897 20: Authentication Key of Tunnel Owner (hex value starting with '0x')
0                                  21: Reserved - set to 0
28800                              22: Timeout in Seconds
```

Figure 85. Export fwexpmctx.manual File Example (V2 R6 and Prior)

Note: It is possible to export multiple tunnel definitions at one time. In this case, the subsequent tunnel definitions are simply appended to the bottom of the file.

© Copyright IBM Corp. 1997, 2000

- The second file created by the export request is known as the policy file and is named `fwexppolicy`. This file contains one line (per tunnel definition) and all data must begin in column 1. Each line contains eight fields, each separated by a blank. Unless stated otherwise, all values are decimal. An example of the format and typical contents of the `fwexppolicy` file follows:

Typical Contents in Fields 1 through 8

The content of a typical `fwexppolicy` is:

```
                          Fields
-----------------------------------------------------------------------
      1               2       3     4       5     6    7       8
-----------------------------------------------------------------------
 11.130.10.5     11.130.10.7  1   28800    n/y   n/y   y   key=manual
```

File Format of Fields 1 through 8:

```
 Field 1 - IP Address of Tunnel Partner
 Field 2 - IP Address of Tunnel Owner
 Field 3 - Tunnel ID
 Field 4 - Timeout in Seconds
 Field 5 - Encryption/Authentication Flag - valid combinations are:
           y/y - Encryption and Authentication
           y/n - Encryption Only
           n/y - Authentication Only
 Field 6 - Encryption Mode/Authentication Mode (y=transport, n=tunnel)
 Field 7 - Authentication First - y or n
           Determines the order of operations when
           both Encryption and Authentication are requested.
 Field 8 - Tunnel Type - must be: key=manual
```

Figure 86. Export fwexppolicy File Example (V2 R6 and Prior)

Note: It is possible to export multiple tunnel definitions at one time. In this case, the subsequent tunnel policy definitions are appended to the bottom of the file.

Export File format for OS/390 Version 2 Release 7 and Release 8

The following OS/390 Version 2 Release 7 definition files are produced when the **fwtunnl cmd=export version=new** or when **version** is not specified.

- The first file is known as the context file and is named `fwexpmctx.manual`. This file contains 36 lines (per tunnel definition) and all data must begin in column 1. Unless stated otherwise, all values are decimal. An example of the format and typical contents of the `fwexpmctx.manual` file follows:

```
Typical Contents:                   File Format:
#Tunnel_partner Version = 1.0  Line 1: context line - must be: #Tunnel_partner Version = 1.0
11.130.10.7                         2: IP Address of Tunnel Owner
11.130.10.5                         3: IP Address of Tunnel Partner
2                                   4: Tunnel Id
1000                                5: ESP SPI of Tunnel Partner
1000                                6: AH SPI of Tunnel Partner
682262                              7: ESP SPI of Tunnel Owner
682262                              8: AH SPI of Tunnel Owner
CDMF                                9: Encryption Algorithm of Tunnel Partner (CDMF or DES_CBC_8 or DES_CBC_4 or 3DES_CBC or ESP_NULL)
8                                  10: Size (number of bytes) of Encryption Key of Tunnel Partner
0xce803234ce732f25                 11: Encryption Key of Tunnel Partner (hex value starting with '0x')
CDMF                               12: Encryption Algorithm of Tunnel Owner (CDMF or DES_CBC_8 or DES_CBC_4 or 3DES_CBC or ESP_NULL)
8                                  13: Size (number of bytes) of Encryption Key of Tunnel Owner
0xf2e943e6c8c24fa7                 14: Encryption Key of Tunnel Owner (hex value starting with '0x')
KEYED_MD5                          15: Authentication Algorithm of Tunnel Partner (KEYED_MD5 or HMAC_MD5 or HMAC_SHA)
16                                 16: Size (number of bytes) of Authentication Key of Tunnel Partner
0x3ba30d2a8278c94b29828ccef98081af 17: Authentication Key of Tunnel Partner (hex value starting with '0x')
KEYED_MD5                          18: Authentication Algorithm of Tunnel Owner (KEYED_MD5 or HMAC_MD5 or HMAC_SHA)
16                                 19: Size (number of bytes) of Authentication Key of Tunnel Owner
0xc1bbc0b0105bf88387f8708ca2585897 20: Authentication Key of Tunnel Owner (hex value starting with '0x')
0                                  21: Reserved - set to 0
28800                              22: Timeout in Seconds
ESP_NOOP                           23: ESP Authentication Algorithm of Tunnel Partner (HMAC_MD5 or HMAC_SHA or ESP_NOOP)
0                                  24: Size (number of bytes) of ESP Authentication Key of Tunnel Partner
0x                                 25: ESP Authentication Key of Tunnel Partner (hex value starting with '0x')
ESP_NOOP                           26: ESP Authentication Algorithm of Tunnel Owner (HMAC_MD5 or HMAC_SHA or ESP_NOOP)
0                                  27: Size (number of bytes) of ESP Authentication Key of Tunnel Owner
0x                                 28: ESP Authentication Key of Tunnel Owner (hex value starting with '0x')
n                                  29: ESP replay prevention indicator for Tunnel Partner (n - no replay, y - replay)
n                                  30: ESP replay prevention indicator for Tunnel Owner (n - no replay, y - replay)
n                                  31: AH replay prevention indicator for Tunnel Partner (n - no replay, y - replay)
n                                  32: AH replay prevention indicator for Tunnel Owner (n - no replay, y - replay)
n                                  33: ESP new header indicator for Tunnel Partner (n - no new header, y - new header)
n                                  34: ESP new header indicator for Tunnel Owner (n - no new header, y - new header)
n                                  35: AH new header indicator for Tunnel Partner (n - no new header, y - new header)
n                                  36: AH new header indicator for Tunnel Owner (n - no new header, y - new header)
```

Figure 87. Export fwexpmctx.manual File Example (V2 R7)

Note: It is possible to export multiple tunnel definitions at one time. In this case, the subsequent tunnel definitions are simply appended to the bottom of the file.

- The second file created by the export request is known as the policy file and is named `fwexppolicy`. This file contains one line (per tunnel definition) and all data must begin in column 1. Each line contains 10 fields, each separated by a blank. Unless stated otherwise, all values are decimal. An example of the format and typical contents of the `fwexppolicy` file follows:

Typical Contents in Fields 1 through 10

The content of a typical `fwexppolicy` is:

```
                            Fields
-------------------------------------------------------------------------------------
      1               2       3     4       5     6    7       8          9     10
-------------------------------------------------------------------------------------
 11.130.10.5     11.130.10.7  1   28800    n/y   n/y   y   key=manual   n/y    y
```

File Format of Fields 1 through 10:

```
 Field 1 - IP Address of Tunnel Partner
 Field 2 - IP Address of Tunnel Owner
 Field 3 - Tunnel ID
 Field 4 - Timeout in Seconds
 Field 5 - Encryption/Authentication Flag of Tunnel Partner - valid combinations are:
             y/y - Encryption and Authentication
             y/n - Encryption Only
             n/y - Authentication Only
 Field 6 - Encryption Mode/Authentication Mode (y=transport, n=tunnel)
 Field 7 - Authentication First indicator for Tunnel Partner - y or n
           Determines the order of operations when
           both Encryption and Authentication are requested.
 Field 8 - Tunnel Type - must be: key=manual
 Field 9 - Encryption/Authentication Flag of Tunnel Owner - valid combinations are:
           y/y - Encryption and Authentication
           y/n - Encryption Only
           n/y - Authentication Only
 Field 10- Authentication First indicator for Tunnel Owner - y or n
           Determines the order of operations when
           both Encryption and Authentication are requested.
```

Figure 88. Export fwexppolicy File Example (V2 R7)

Note: It is possible to export multiple tunnel definitions at one time. In this case, the subsequent tunnel policy definitions are appended to the bottom of the file.

Appendix G. SMF Record Type 109

The SecureWay Communication Server syslog daemon can optionally send output to the OS/390 System Management Facilty (SMF). The following is the layout of the record type where the firewall logging data is kept in SMF.

Format

Figure 89. SMF Record 109 Layout

Offsets		Name	Length	Format	Description
00	00	SMF109LEN	2	binary	Record length (maximum size of 32,756). This field and the next field (total of four bytes) form the record descriptor word (RDW). The first two bytes (this field) must contain the logical record length including the RDW. The second two bytes (the following field) are used for variable block spanned records. If the record is not spanned, set these two bytes to hexadecimal zeroes. These fields must be filled in before writing the record to the SMF data set.
02	02	SMF109SEG	2	binary	Segment descriptor (see record length field above.)
04	04	SMF109FLG	1	binary	System indicator: **Bit** **Meaning When Set** **0-2** Reserved **3** MVS/SP Version 4 and above. Bits 3, 4, 5, and 6 are on.* **4** MVS/SP Version 3 and above. Bits 4, 5, and 6 are on. **5** MVS/SP Version 2 and above. Bits 5 and 6 are on. **6** VS2. Bit 6 is on. **7** Reserved. *IBM recommends that you use information located elsewhere in this record to determine the MVS product level.
05	05	SMF109RTY	1	binary	Record type: 109 (X'6D')
06	06	SMF109TME	4	binary	Time since midnight, in hundredths of a second, record was moved into the SMF buffer. In record types 2 and 3 this field indicates the time that the record was moved to the dump data set.
10	0A	SMF109DTE	4	packed	Date when the record was moved into the SMF buffer. in the form 00*yyddF* or 0*cyydddF* (where *c* is 0 for 19*xx* and 1 for 20*xx*, *yy* is the current year (0-99), *ddd* is the current day (1-366), and *F* is the sign). In record types 2 and 3, this field indicates the date that the record was moved into the dump data set.
14	0E	SMF109SID	4	EBCDIC	System identification (from the SID parameter).
18	12	SMF109LOG	4096	EBCDIC	**syslogd** messages.

© Copyright IBM Corp. 1997, 2000

Appendix H. Initializing Security Associations with ISAKMP and IKE

This section outlines how ISAKMP protocols initially establish security associations (SAs) and exchange keys between two systems that want to communicate securely.

The Internet Security Association and Key Management Protocol (ISAKMP) defines a framework for dynamically establishing security associations and cryptographic keys in an Internet environment. This framework defines a set of message flows and message formats. The message flows in ISAKMP are called exchanges. The message formats in ISAKMP are called payloads. ISAKMP defines a generic payload for key exchange information. This allows the ISAKMP protocol to manage cryptographic keys independent of the key exchange protocol that is used to generate them.

ISAKMP defers the interpretation of the key exchange payload to individual key exchange protocols. Internet Key Exchange (IKE) is such a protocol. IKE augments the ISAKMP protocol to facilitate the creation of authenticated keying material. IKE defines how keying material is generated. The exchanges defined by ISAKMP require authentication to take place, but they do not specify how authentication is to be performed. IKE defines how authentication is to be performed.

ISAKMP defines two phases of negotiation. Both of these phases are also applicable to the IKE protocol. The first phase is referred to as Phase 1. In Phase 1, two ISAKMP servers agree on how to protect traffic between themselves. This agreement results in the creation of an ISAKMP security association. The second phase is referred to as Phase 2. In Phase 2, security associations for other security protocols are established; for example, AH or ESP. Negotiations during each phase are accomplished using an ISAKMP defined exchange or by an exchange specific to a key exchange protocol.

Phase 1:

IKE supports two type of Phase 1 exchanges - main mode and aggressive mode. Both of these exchange modes are based on exchanges defined by ISAKMP. Main mode is an implementation of ISAKMP's Identity Protect Exchange. Aggressive mode is an implementation of ISAKMP's Aggressive exchange.

IKE defines four techniques for authentication of phase 1 exchanges. These techniques are:

- Pre-shared key
- Signature based
- Public key encryption
- Revised public key encryption

Note: Of these techniques pre-shared key and signature based using RSA signatures are supported by the OS/390 Firewall Technologies ISAKMPD server.

Main Mode: A main mode exchange is comprised of six messages as shown in Figure 90 on page 464

© Copyright IBM Corp. 1997, 2000

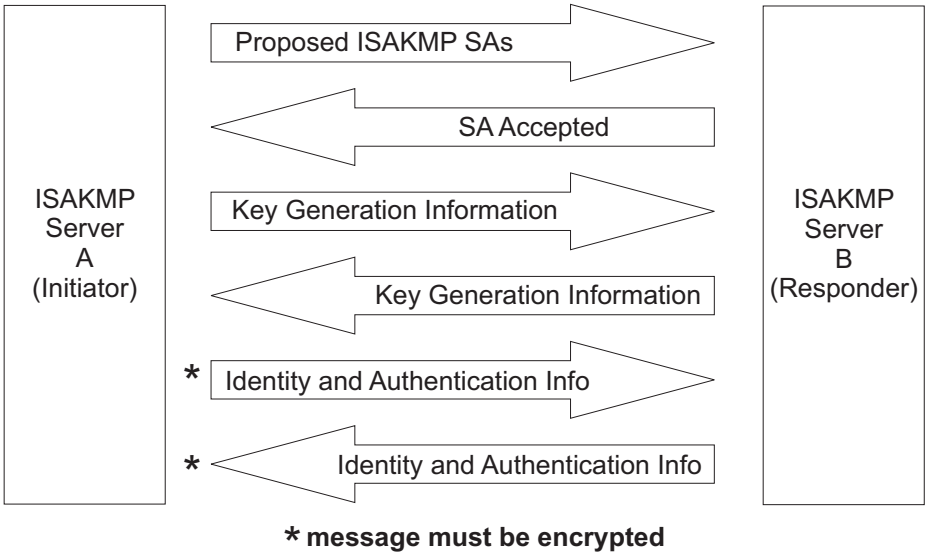


Figure 90. Security Policy

Messages 1 and 2 are used to agree on the negotiable attributes of the ISAKMP security association used to protect phase 2 negotiations established using this Phase 1. The initiator sends a list of acceptable security associations to the responder in message 1. Each security association defines an acceptable combination of attributes for the ISAKMP SA being negotiated. The responder picks a security association that is acceptable and returns the choice to the initiator in message 2.

The attributes that can be negotiated in Phase 1 are:

- authentication method (for example, pre-shared key or RSA signature)
- hash algorithm (for example, MD5 or SHA)
- encryption algorithm (for example DES_CBC_8 or 3DES_CBC)
- Diffie-Hellman group information (for example group 1)
- Life time and life size of the ISAKMP SA

Messages 3 and 4 are used to exchange information specific to the generation of a shared secret key. This information includes Diffie-Hellman public values and a randomly generated value called a nonce. The initiator sends his Diffie-Hellman public value (i.e. g**x mod n) and a nonce in message 3. The responder sends his Diffie-Hellman public value (i.e. g**y mod n) and a nonce in message 4. With this information, both the responder and initiator can independently generate the identical keying information. The calculations used to generate keying information varies depending on the authentication method that was agreed upon during messages 1 and 2. The keying information generated by both sides include:

- a key used for authenticating messages sent under the protection of this ISAKMP SA (e.g. phase 2 messages),
- a key used to encrypt messages sent under the protection of this ISAKMP SA (e.g. phase 2 messages),
- keying material that will be used to derive keys established for phase 2 SAs.

Messages 5 and 6 are used to exchange identity information and authentication information. The authentication information varies depending on the authentication method that was agreed upon during messages 1 and 2. For pre-shared key authentication, public key encryption authentication, and revised public key encryption authentication the information takes the form of an encrypted hash. For signature based authentication, this information takes the form of a signature. The initiator includes his identity and authentication information in message 5. The responder includes his identity and authentication information in message 6.

Main mode provides a mechanism to exchange certificates when signature based authentication is used. This mechanism is not shown in Figure 90, but works as follows. In messages 5 the initiating ISAKMP server may include the certificate it used to create its signature. In messages 6 the responding ISAKMP server may include the certificate it used to create its signature. Inclusion of the certificate is optional unless the ISAKMP server's peer explicitly requests that the certificate be sent.

Aggressive Mode

An aggressive mode exchange is comprised of 3 messages as shown in Figure 91

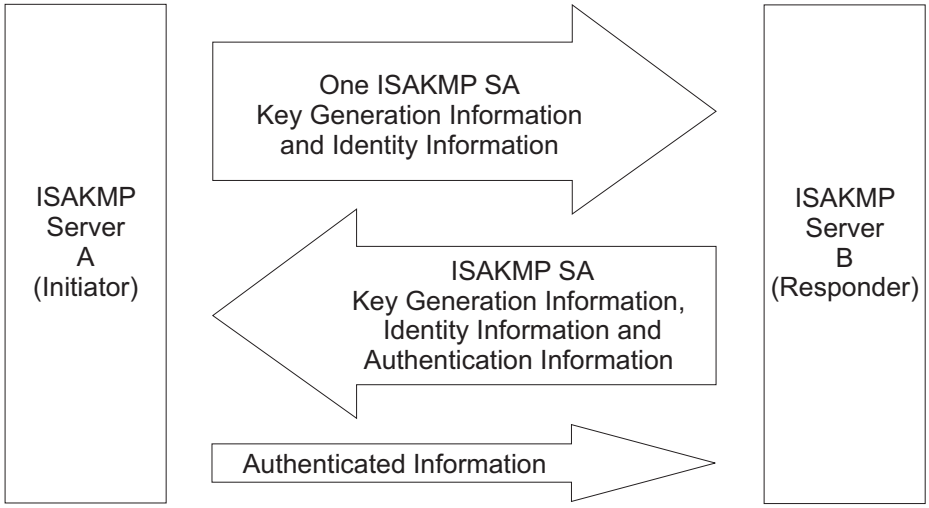


Figure 91. Security Policy

Aggressive mode exchanges the same information as main mode differing in the following:

- In aggressive mode the initiator may send only one proposal. In main mode, the initiator may send a list of proposals.
- In aggressive mode only 3 messages are exchanged instead of 6 message as in main mode.
  - Message 1 of aggressive mode contains all the information that was contained in messages 1 and 3 of main mode, plus the identity information sent in message 5 of main mode.
  - Message 2 of aggressive mode contains all the information sent in messages 2, 4, and 6 of main mode.
  - Message 3 of aggressive mode contains the authentication information that was contained in message 5 of main mode.
- In aggressive mode, no messages are required to be encrypted. Message 3 can be sent encrypted, but doing so provides little additional protection. In main mode messages 5 and 6 are required to be encrypted. The ISAKMP servers send their identity in message 5 or 6 of main mode. The result is that main mode protects the identity of the ISAKMP servers while aggressive mode does not.

Aggressive mode provides a mechanism to exchange certificates when signature based authentication is used. This mechanism is not shown in Figure 91 but works as follows.

In messages 2 the initiating ISAKMP server may include the certificate it used to create its signature. In messages 3 the responding ISAKMP server may include the certificate it used to create its signature. Inclusion of the certificates is optional unless the ISAKMP server's peer explicitly requests that the certificate be sent.

Phase 2:

IKE supports one type of phase 2 exchange - quick mode. Quick mode is an IKE specific exchange. It is not based on an ISAKMP defined exchange.

Quick mode exchanges are bound to a specific Phase 1 exchange. This is accomplished by encrypting a hash of each quick mode message with a cryptographic key derived during the Phase 1 exchange. No explicit authentication of the identities involved in a Phase 2 exchange is performed.

Quick mode: A quick mode exchange is comprised of 3 messages as shown in Figure 92

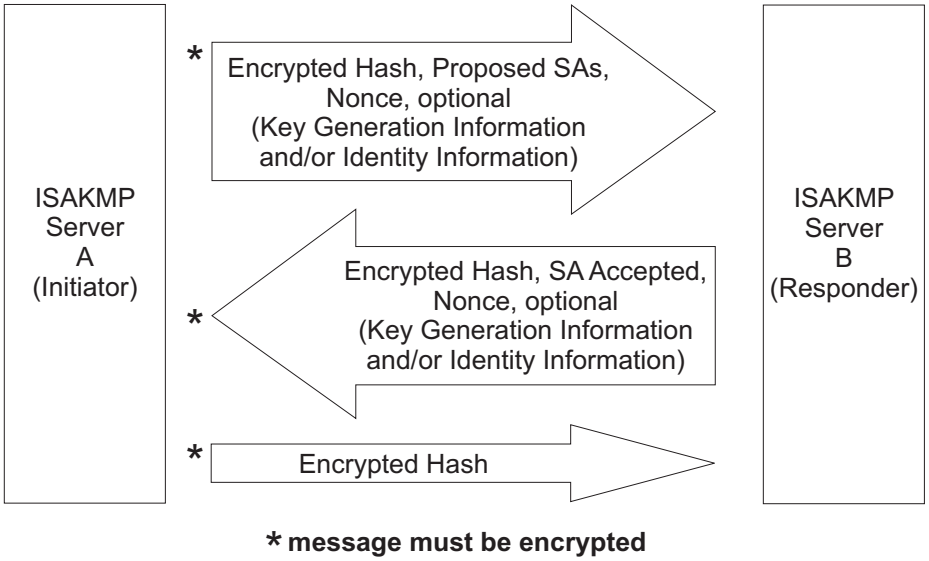


Figure 92. Security Policy

In quick mode each message contains an encrypted hash. This hash authenticates the source of the message (for example, verifies it is bound to an ISAKMP SA), authenticates the integrity of the message, and proves liveliness. In message 1 the initiator sends a list of acceptable proposals to the responder. Each proposal defines an acceptable combination of attributes for the non-ISAKMP SA being negotiated (AH or ESP SA). The responder picks a proposal that is acceptable and returns the choice to the initiator in message 2.

The attributes that can be negotiated in quick mode include:

- protocol (AH, ESP, or both AH and ESP)
- authentication algorithm (HMAC-MD5 or HMAC-SHA)
- encapsulation mode (tunnel or transport)
- encryption algorithm (DES_CBC_8 or 3DES_CBC)
- Diffie-Hellman group information (group 1)
- Life time and life size of the ISAKMP SA

Quick mode allows an optional Diffie-Hellman exchange to occur. When the Diffie-Hellman exchange is to take place the initiator includes his Diffie-Hellman public value (for example, g**x mod n) in message 1 and the responder includes his Diffie-Hellman public value (for exmaple, g**y mod n) in message 2. The key generated from this Diffie-Hellman exchange will be used in the calculation that generates the keying material for the non-ISAKMP SA. The Diffie-Hellman exchange provides Perfect Forward Secrecy (PFS). PFS is the notion that a compromise of a single key will only permit access to data encrypted with that key. Without the Diffie-Hellman exchange the keying material generated during quick mode does not have the PFS characteristics.

The keying information generated in quick mode is dependent on security association chosen by the responder. The keying information generated for an ESP security association might include:

- a key for generating the HMAC-MD5 value of transmitted packets
- a key for validating the HMAC-MD5 value of received packets
- a key for encrypting transmitted packets
- a key for decrypting received packets

Quick mode allows the identities of the source and destination of the IP traffic to be optionally included in message 1 and 2. If no identities are included then the source and destination of the traffic is assumed to be the IP addresses of the two devices that were identified during the phase 1 SA exchange. If the ISAKMP server is negotiating non-ISAKMP SAs on behalf of other devices (for example, the ISAKMP server is functioning as a gateway), the identities of these devices must be included.

Message 3 of quick mode indicates that the initiator received and successfully processed message 2. This message tells the responder that the newly negotiated SAs may now be used.

Appendix I. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

© Copyright IBM Corp. 1997, 2000

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
U.S.A

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

IBM is required to include the following statements in order to distribute portions of this document and the software described herein to which contributions have been made by the University of California, NEC Systems Laboratory, and JScape Corporation.

This product includes software developed by the University of California, Berkeley and its contributors.

Portions Copyright © 1993, 1994 by NEC Systems Laboratory.

This application is powered by JScape Components © 1995, 1996, 1997 JScape Corporation, Scottsdale, Arizona. All rights reserved.

This product contains code licensed from RSA Data Security Incorporated.



Trademarks

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

AIX	BookManager	SecureWay
IBM	OpenEdition	MVS/SP
OS/390	RACF	S/390

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States or other countries.

Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

Bibliography

This bibliography is a list of publication for OS/390 Firewall Technologies and other products. The complete title and order number is given for each publication.

IBM Product Libraries

OS/390 IBM Communications Server IP Library:

- *OS/390 IBM Communications Server: IP Configuration Reference*, SC31-8726
- *OS/390 IBM Communications Server: IP Configuration Guide*, SC31-8725
- *TCP/IP for MVS: Messages and Codes*, SC31-7132
- *OS/390 UNIX System Services User's Guide*, SC28-1891
- *OS/390 IBM Communications Server: IP Programmer's Reference*, SC31-8515
- *OS/390 IBM Communications Server: IP Diagnosis*, SC31-8521

OS/390 UNIX Library:

- *OS/390 UNIX System Services Planning*, SC28-1890
- *OS/390 UNIX System Services Command Reference*, SC28-1892

OS/390 Cryptographic Services

- *OS/390 Open Cryptographic Services Facility Application Developer's Guide and Reference*, SC24-5875
- *OS/390 SecureWay Security Server Open Cryptographic Enhanced Plug-ins Guide and Reference*, SA22-7429

OS/390 Security Server

- *OS/390 SecureWay Security Server RACF Command Language Reference*, SC28-1919

MVS/ESA Library:

- *OS/390 MVS Initialization and Tuning Reference*, SC28-1752

ICSF/MVS Library:

- *OS/390 SecureWay Integrated Cryptographic Service Facility Administrator's Guide*, SC23-3975

Information on the Internet

- Socks information: **http://www.socks.nec.com**
- RealAudio information: **http://www.real.com**
- OS/390 information: **http://www.ibm.com/s390/firewall** and **http://www.s390.ibm.com/firewall**
- WinZip information: **http://www.winzip.com**

Redbooks

- *Stay Cool on OS/390: Installing Firewall Technology*, SG24-2046
- *SecureWay Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements*, SG24-5631

Other References

- Cheswick, William R. and Bellovin, Steven M., *Firewalls and Internet Security : Repelling the Wily Hacker* (Addison-Wesley Professional Computing Series).

© Copyright IBM Corp. 1997, 2000

Index

Special Characters

-a parameter
of fwlogmgmt command 272
-c parameter
of fwtrace command 314
-l parameter
of fwlogmgmt command 272
of fwtrace command 314
/etc/security/fwstacks/<STACK>/fwnamed.boot file 75
/etc/security/fwstacks/<STACK>/fwnamed.loc file 75
/etc/security/fwstacks/<STACK>/fwresolv.conf file 75
/etc/security/fwstacks/<STACK>fwnamed.ca file 75

A

abend code S069 from firewall server 453
about this book xiii
activate parameter
of fwdynconns command 224
of fwtunnl command 322
activate sample configuration files 30
activating a LAN connection 424
activating FTP connection 426
activating system configuration changes 33
activation, connection from the configuration 126
adapter, secure, configuring 21
add ah parameter
of fwdataprop command 212
add esp parameter
of fwdataprop command 212
add parameter
of fwahtran command 180
of fwarchive command 184
of fwauthinfo command 188
of fwcertauth command 192
of fwdatapol command 207
of fwdynconns command 221
of fwdyntun command 228
of fwesptran command 231
of fwfrule command 240
of fwkeypol command 251
of fwkeyprop command 255
of fwkeysrv command 264
of fwkeysrvgrp command 259
of fwkeytran command 268
of fwnat command 280
of fwnwgrp command 286
of fwnwobj command 288
of fwondemand command 292
add parameter *(continued)*
of fwservice command 302
of fwsrule command 305
of fwstack command 311
of fwtunnl command 318
adding administrators to FWGRP 25
addremkeysrv parameter
of fwkeysrvgrp command 261
address translation and filters 4
addresses, entering IP xiv
addservice parameter
of fwconns command 197
addsocks parameter
of fwconns command 198
administrators, adding 25
AH transform 150
AIX configuration client (GUI) setup 36
allow LANs to ping internet addresses 423
anchor rules 164
anonymous FTP 67
audit log facility 128
authentication (tunnel security) 80
authentication information 161
authority, granting to users and groups 24
authorize the Firewall to external security manager 23
authorize the firewall to ICSF/MVS 27
automated migration 16

B

bibliography 473
blanket policies for firewall, set 111
books, list of Firewall and related 473
building connections 58, 125
building VPN connections 141

C

cancel parameter
of fwkern command 247
certificate authority 158
change an IP rule 119
change parameter
of fwadapter command 179
of fwahtran command 181
of fwarchive command 185
of fwconns command 196
of fwdaemon command 201
of fwdatapol command 206
of fwdataprop command 211
of fwdns command 216
of fwdynconns command 223

© Copyright IBM Corp. 1997, 2000

change parameter *(continued)*
of fwdyntun command 229
of fwesptran command 232
of fwfrule command 242
of fwkeypol command 252
of fwkeyprop command 255
of fwkeysrv command 266
of fwkeysrvgrp command 260
of fwkeytran command 269
of fwnat command 281
of fwnwgrp command 285
of fwnwobj command 289
of fwondemand command 294
of fwsecpolicy command 296
of fwservice command 300
of fwsrule command 307
of fwtunnl command 322
changes in this document xvii
changes to DNS settings 75
choices for your firewall 9
command common return codes 178
command syntax 177
command syntax notes 177
commands 177
configuration
adapters 179
address translation 279
connections 195
DNS 216
filter 238
filters 235
firewall kernel 247
log management 272
logging 274
migrating 275
network object groups 284
RealAudio 187
security policy 296
services 195, 298
single network objects 288
Socks rules 305
tracing 314
tunnels, IP 316
fwadapter 179
fwahtran 180
fwarchive 184
fwaudio 187
fwauthinfo 188
fwcertauth 192
fwconns 195
fwdaemon 201
fwdatapol 205
fwdataprop 210
fwdns 216
fwdynconns 220
fwdyntun 228
commands *(continued)*
fwesptran 231
fwfilter 235
fwfrule 238
fwkern 247
fwkeypol 251
fwkeyprop 254
fwkeyring 258
fwkeysrv 264
fwkeysrvgrp 259
fwkeytran 268
fwlogmgmt 272
fwlogtxt 274
fwmigrate 275
fwnat 279
fwnwgrp 284
fwnwobj 288
fwondemand 292
fwsecpolicy 296
fwservice 298
fwsrule 305
fwstack 311
fwtrace 314
fwtunnl 316
commands, common parameters 177
common parameters, commands 177
common return codes 178
configuration changes, activating 33
configuration client (GUI) 35
configuration client (GUI) setup on AIX 36
configuration client (GUI) setup on Windows 36
configuration client (GUI), logon 37
configuration client, configuring tunnels 130
configuration example 421
configuration file entry interactions, NAT 104
configuration server 35
configuration, default filter 114
configure DNS 129
configure firewall servers 31
configure TCP/IP on firewall host 28
configuring
adapter, secure 21
firewall, overview 21
configuring and activating a manual tunnel 79
configuring and using the DNS 71
configuring and using the socks server 61
configuring FTP 425
configuring HTTP 428
configuring logging using configuration client 128
configuring socks rules 58
configuring tunnels using the configuration client 130
connection activation 126
connection, building 58
connections
building 58

connections *(continued)*
re-ordering 59
connections setting, proxy FTP server 66
connections, building 125
connections, order 126
controlling servers 31
conventions used in this book xiv
correct command syntax 177
creating a connection 424
creating a connections 430
creating a service 429
creating connections 426
cryptographic support, hardware 27

D

data management policies 147
data policy 153
data proposal 152
deactivate parameter
of fwdynconns command 225
of fwtunnl command 322
default filter configuration 114
default set of services 53
define firewall logging 34
define the secure interface 31
defining FTP users 66
defining network objects 422
defining users and groups 23
delete parameter
of fwahtran command 181
of fwarchive command 185
of fwauthinfo command 189
of fwcertauth command 192
of fwconns command 196
of fwdatapol command 206
of fwdataprop command 211
of fwdynconns command 223
of fwdyntun command 229
of fwesptran command 232
of fwfrule command 242
of fwkeypol command 252
of fwkeyprop command 255
of fwkeysrv command 266
of fwkeytran command 269
of fwnat command 281
of fwnwgrp command 285
of fwnwobj command 289
of fwondemand command 294
of fwservice command 301
of fwsrule command 307
of fwtunnl command 322
offwkeysrvgrp command 260
delete rule templates 119
deleting a rule 57
description parameter
of fwkeypol command 251
determining proxy FTP server connections setting 66
DNS
See domain name server (DNS)
DNS problems 455
DNS Server objectives 8
DNS settings, changing 75
DNS, configure 129
document changes xvii
domain name server (DNS)
configuration
alternate 451
recommended 74
with no secure name server 451
on the firewall 73
parts of
name servers 72
resolver routines 72
resource records 72
request routing 73
dynamic tunnel policy 155
dynamic tunnels (ISAKMP) 79
dynamic VPN connection 166
Dynamic VPN examples 437

E

enabling firewall services and features 32
encryption (tunnel security) 80
enhancements xiv
entering IP addresses xiv
error messages and explanations 330
ESP transform 147
examples
of fwadapter command 179
of fwaudio command 187
of fwdaemon command 203
of fwdns command 219
of fwfilter command 236
of fwfrule command 244
of fwkern command 248
of fwlogmgmt command 273
of fwlogtxt command 274
of fwmigrate command 275
of fwnat command 282
of fwnwobj command 290
of fwsecpolicy command 297
of fwservice command 303
of fwsrule command 308
of fwstack command 312
of fwtrace command 315
of fwtunnl command 324
EXCLUDE 103

EXCLUDE Secured IP Addresses 104
export file formats 457
export parameter
of fwtunnl command 323
external security manager authorization 23
extracting log messages 47

F

file formats for export files 457
File Transfer Protocol (FTP) proxy
See FTP proxy
file, trace 49
filter configuration, default 114
filter rules 55, 115
activation states 60
adding 55
base, creating 55
base, managing 59
changing 57
default 55
filtering, packet
See filters, IP
filters and address translation 4
filters, IP
components of
connections 58
diagram 52
explanation 52
filter rules 55
network object groups 54
network objects 53
rules, filter 55
services 57
description of 51
firewall
concepts 3
introduction to 3
servers, network 5
firewall configuration example 421
firewall configuration files 15
firewall kernel 34
firewall log 40
Firewall logging 45
Firewall TCP/IP Stack 31
format of messages 329
format, trace file 49
FTP (proxy) server connections setting, determining 66
FTP problems 456
FTP proxy
process 67
FTP server 7
fwadapter command
examples 179
parameters
change 179
fwadapter command *(continued)*
parameters *(continued)*
list 179
fwahtran command
parameters
add 180
change 181
delete 181
list 181
fwarchive command
&I2@FWCAR.
add 184
change 185
delete 185
parameters
fwaudio command
examples 187
parameters
change 187
list 187
port 187
sessions 187
fwauthinfo command
change 189
delete 189
list 189
parameters
fwcertauth command
add 192
change 192
delete 192
list 193
parameters
fwconns command
parameters
addservice 197
addsocks 198
change 196
create 196
delete 196
list 196
move 197
moveservice 197
movesocks 198
removeservice 197
removesocks 198
fwdaemon command
examples 203
parameters
change 201
list 202
query 202
start 202
stop 202
fwdatapol command
parameters
add 207

fwdatapol command *(continued)*
parameters *(continued)*
change 206
create 205
delete 206
list 206
move 207
remove 207
fwdns command
examples 219
parameters
change 216
list 216
fwdynconns command
parameters
activate 224
add 221
change 223
deactivate 225
delete 223
list 223
listactivate 223
refreshn 225
shutdown 225
fwdyntun command
fwesptran command
parameters
add 231
change 232
delete 232
list 232
fwfilter command
examples 236
parameters
list 235
listsocks 235
shutdown 235
startlog 235
stoplog 235
update 235
verify 235
fwfrule command
examples 244
parameters
add 240
change 242
delete 242
list 242
FWGRP, adding administrators 25
fwkern address space 31
fwkern command
examples 248
parameters
cancel 247
modiFy 247
Start 247
stoP 247
fwkern does not start 453
abend code S069 from firewall server 453
fwkeypol command
parameters
add 251
add ah 212
add esp 212
change 211, 252
create 211
delete 211, 252
description 251
list 211, 252
move ah 212
move esp 213
name 251
remove ah 212
remove esp 213
fwkeyprop command
parameters
add 255
change 255
create 254
delete 255
list 255
move 256
remove 256
fwkeyring command
change 258
list 258
parameters
fwkeysrv command
parameters
add 264
change 266
delete 266
list 266
fwkeysrvgrp command
parameters
add 259
addremkeysrv 261
change 260
delete 260
list 260
move 260
removeremkeysrv 261
fwkeytran command
parameters
add 268
change 269
delete 269
list 269
fwlogmgmt command
examples 273
parameters
-a 272
-l 272

fwlogtxt command
examples 274
fwmigrate command
examples 275
fwnat command
entry types
exclude 281
map 280
reserve 280
translate 280
examples 282
parameters
add 280
change 281
delete 281
list 281
listlog 282
shutdown 281
startlog 281
stoplog 281
update 281
verify 281
fwnwgrp command
parameters
add parameter 286
change parameter 285
create parameter 284
delete parameter 285
list parameter 285
remove parameter 286
fwnwobj command
examples 290
parameters
add 288
change 289
delete 289
list 289
fwondemand command
change 294
delete 294
parameters
add 292
fwsecpolicy command
examples 297
parameters
change 296
list 297
fwservice command
examples 303
parameters
add 302
change 300
create 299
delete 301
list 301
move 302
remove 302
fwsrule command
examples 308
parameters
add 305
change 307
delete 307
list 308
fwstack command
parameters
add 311
delete 311
list 312
query 312
rename 311
fwstackd server 6
fwtimernat daemon 104
fwtrace command
examples 315
parameters
-c 314
-l 314
fwtunnl command 316
examples 324
parameters
activate 322
add 318
change 322
deactivate 322
delete 322
export 323
import 323
list 322
listactive 323
shutdown 323

G

general problems
fwkern does not start 453
general security policy 111
granting users and groups authority 24
graphical user interface 35
group definition, planning 23
group of network objects 114
group, network object 53
groups and users, granting authority 24
groups, defining 23
groups, network object
See network object groups
GUI client setup on AIX 36
GUI client setup on Windows 36
GUI logon 37

H

hardware cryptographic support 27
HFS name changes 16
HFS, logging to 46
how servers are controlled 31
how to use this book xiii
HTTP, activating 431
HTTP, configuring 428

I

ICSF/MVS 27
identify firewall TCP/IP stack 31
IKE 463
import parameter
of fwtunnl command 323
inter-operability considerations (IP security), manual tunnels 79
interface, graphic user 35
interfaces, network 14
Internet Key Exchange 463
Internet Protocol (IP) security 77
choosing tunnel types 80
dynamic tunnels (ISAKMP) 79
IP security architecture 77
manual tunnels 79
security associations 78
tunnel modes 80
tunnels and key management 78
using IP security 77
internet protocol security 8
introducing the OS/390 Firewall Technologies 3
IP addresses, entering xiv
IP filters 4
See also filters, IP
IP inter-operability considerations, manual tunnels 79
IP rule, modify 119
IP security
See Internet Protocol (IP) security
IP Security Example 435
IP tunnel 141
IPSec 8
ISAKMP 79
ISAKMP examples 437
ISAKMP protocols 463

K

key policy 145
key proposal 144
key ring 157
key server 159
key server group 162
key server information 157
key transform configuration entry 143

L

licensing agreement 470
list parameter
of fwadapter command 179
of fwahtran command 181
of fwauthinfo command 189
of fwcertauth command 193
of fwconns command 196
of fwdaemon command 202
of fwdatapol command 206
of fwdataprop command 211
of fwdns command 216
of fwdynconns command 223
of fwdyntun command 229
of fwesptran command 232
of fwfilter command 235
of fwfrule command 242
of fwkeypol command 252
of fwkeyprop command 255
of fwkeyring command 258
of fwkeysrv command 266
of fwkeysrvgrp command 260
of fwkeytran command 269
of fwnat command 281
of fwnwgrp command 285
of fwnwobj command 289
of fwsecpolicy command 297
of fwservice command 301
of fwsrule command 308
of fwstack command 312
of fwtunnl command 322
listactivate parameter
of fwdynconns command 223
listlog parameter
of fwnat command 282
listsocks parameter
of fwfilter command 235
local4 log facility 128
locally activated dynamic connections 165
log archive management 128
log messages, messages 47
log on to configuration client (GUI) 37
log viewer 39, 40
logging
overview
logging by the Firewall 45
logging servers, other
explanation of fwstackd server 6
logging to HFS 46
logging, defining 34
logging, using configuration client 128

M

making your firewall choice 9
manaual tunnels 79
manual tunnels,
configuring and activating 79
inter-operability considerations 79
MAP 103
MAP Secured IP Address 104
maxconns for proxy FTP 66
messages and explanations 330
messages format 329
migration considerations
for CFGSRV and fwstackd 18
for fwkern 17
for ISAKMPD 18
for PROCLIB 18
HFS name changes 16
removal of fwadm user ID 17
migration for CFGSRV and fwstackd 18
migration for fwkern 17
migration for fwtrace 16
migration for ISAKMPD 18
migration for PROCLIB 18
modify an IP rule 119
modiFy parameter
of fwkern command 247
move ah parameter
of fwdataprop command 212
move esp parameter
of fwdataprop command 213
move parameter
of fwconns command 197
of fwdatapol command 207
of fwkeysrvgrp command 260
of fwservice command 302
moveservice parameter
of fwconns command 197
movesocks parameter
of fwconns command 198

N

NAT 103, 104, 106
See also Network Address Translation (NAT)
EXCLUDE 103
MAP 103
NAT configuration file 103
NAT configuration file entry interactions 104
NAT Logging 106
RESERVE 103
TRANSLATE 103
NAT (Network Address Translation 45
NAT entry 169
navigation tree 38
network
interfaces, types 14
network address translation 169
Network Address Translation (NAT) 5, 45, 103
Network Addresses Translation (NAT) 103
EXCLUDE 103
MAP 103
RESERVE 103
TRANSLATE 103
network object groups
defining 54
network objects 112
defining 54
types 54
network servers 5
networks, virtual private 8
notes for command syntax 177
notices 467

O

Objectives of Network Address Translation (NAT) 5
objectives, DNS Server 8
objectives, proxy server 7
objectives, socks server 6
objects, network 112
See also network objects
order connections 126
ordering connections 59
overview of DNS 72

P

packet filtering
See filters, IP
password validation errors 67
planning for group definition 23
policy
security
IDs, valid 296
tunnel, values 318
pre-shared key authentication 437
predefined services 53
preface xiii
prerequisite knowledge xiv
private networks, virtual 8
problem determination 453
proxy FTP server connections setting, determining 66
proxy server 7
FTP
changing user passwords 67
configuring and starting 65
defining users 66
overview 65
server connections setting 66

proxy server objectives 7

R

RealAudio 5
RealAudio protocol 107
refresh parameter
of fwdynconns command 225
removal of FWADM iser ID 17
remove ah parameter
o fwdataprop command 212
remove esp parameter
of fwdataprop command 213
remove parameter
of fwdatapol command 207
of fwkeyprop command 256
removeremkeysrv parameter
of fwkeysrvgrp command 261
removeservice parameter
of fwconns command 197
removesocks parameter
of fwconns command 198
RESERVE 103
RESERVE registered address 103
resolver configuration file 75
routing problems 453
RSA Signature authentication 443
rule templates 115
rule templates, delete 119
rule, deleting 57
rules, filter
See filter rules

S

S069 abend code 453
sample configuration files, activating 30
secure adapter, configuring 21
secure and nonsecure domains, socks 62
secure interface, defining 31
secure name server 75
security
policy
IDs, valid 296
tunnel, values 318
Security (IP) Example 435
Security Associations (SAs) 463
security policy, general 111
security, IP 8
server
DNS
introduction to 7
fwstackd
explanation of fwstackd 6
name 72
proxy
introduction to 7
server *(continued)*
socks
introduction to 6
server connections setting, proxy FTP 66
server, domain name
See also domain name server (DNS)
explanation of 7
server, proxy
See proxy server
server, socks
See also socks server
explanation of 6
servers, configuring 31
services
filter
adding 302
changing 300
creating 299
deleting 301
listing 301
moving 302
removing 302
services, predefined 53
set blanket policies for firewall 111
set UNIX system parameters 22
setting proxy FTP server connections 66
setting up the configuration client (GUI) 35
setting up the configuration client (GUI) on AIX 36
setting up the configuration client (GUI) on Windows 36
setting up the configuration server 35
shutdown parameter
of fwfilter command 235
of fwnat command 281
of fwtunnl command 323
SMF record type 109 461
sockd 62
See also socks server
socks and secure and nonsecure domains 62
socks daemon
See socks server
socks problems 456
socks rules, configuring 58
socks server 6
clients 61
overview 61
rules
adding 305
changing 307
deleting 307
listing 308
sockd 61
using 61
socks server objectives 6
socks server, configuring and using 61

socks server, starting 62
Socks templates 122
socks with filters 58
softcopy publications xv
start parameter
of fwkern command 247
starting the firewall kernel 34
starting the socks server 62
starting the UNIX services name server 76
startlog parameter
of fwfilter command 235
of fwnat command 281
stoP parameter
of fwkern command 247
stoplog parameter
of fwfilter command 235
of fwnat command 281
summary of changes xvii
supporting the RealAudio protocol 107
syntax notes 177
syntax of a command 177
system configuration changes, activating 33
system parameters, setting 22

T

TCP/IP stack, identify 31
templates, rule 115
templates, Socks 122
tracing
files
format 49
name 49
specifying 314
firewall activity
overview 49
levels 314
trademarks 470
traffic
firewall
controlling 51
TRANSLATE 103
TRANSLATE Secured IP Addresses 104
Translating Network Addresses 103
EXCLUDE 103
MAP 103
RESERVE 103
TRANSLATE 103
tunnel modes 80
tunnel policy 140
tunnel types 80
tunnels 140
activating 322
adding 318
changing 322
deactivating 322
tunnels *(continued)*
deleting 322
listing 322, 323
shutting down 323
tunnels, configure using the configuration client 130
tunnels, IP 140
security
authentication 80
encryption 80
policy 80

U

UNIX name server, starting 76
update parameter
of fwfilter command 235
of fwnat command 281
updating filters to activate connection, FTP 426
updating filters to activate connection, LANS 424
updating filters to activate HTTP 431
user interface, graphical 35
users
FTP
defining users 66
passwords, changing 67
users and groups, granting authority 24
users, defining 23

V

V2R10 enhancements xiv
verify parameter
of fwfilter command 235
of fwnat command 281
virtual private network
benefits of 77
definition 8
VPN 8
VPN connections, building 141
VPN customer scenarios 81

W

who should use this book xiii
Windows configuration client (GUI) setup 36

Readers' Comments

OS/390®
SecureWay Security Server
Firewall Technologies
Guide and Reference

Publication No. SC24-5835-05

You may use this form to report errors, to suggest improvements, or to express your opinion on the appearance, organization, or completeness of this book.

Date: _____

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note

Report system problems to your IBM representative or the IBM branch office serving you.
U.S. customers can order publications by calling the IBM Software Manufacturing Solutions at **1-800-879-2755**.

In addition to using this postage-paid form, you may send your comments by:

FAX	1-607-752-2327	Internet	pubrcf@vnet.ibm.com
IBMLink	GDLVME(PUBRCF)		

Would you like a reply? ___YES ___ NO If yes, please tell us the type of response you prefer.

___ Electronic address: _____

___ FAX number: _____

___ Mail: (Please fill in your name and address below.)

Name _____ Address _____

Company or Organization _____

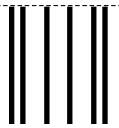
Phone No. _____

Readers' Comments
SC24-5835-05

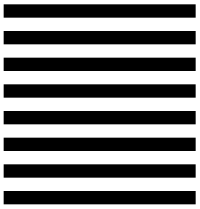
Cut or Fold
Along Line



Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Department G60
International Business Machines Corporation
Information Development
1701 North Street
ENDICOTT NY 13760-5553



Fold and Tape **Please do not staple** Fold and Tape

SC24-5835-05

Cut or Fold
Along Line



Program Number: 5647-A01



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC24-5835-05



Spine information:



OS/390 Firewall Technologies Guide and Reference