

CLI Book 2: Cisco Secure Firewall ASA Firewall CLI Configuration Guide, 9.23

First Published: 2025-01-31

Last Modified: 2026-07-17

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2026 Cisco Systems, Inc. All rights reserved.



CONTENTS

PREFACE

About This Guide xix

Document Objectives xix

Related Documentation xix

Document Conventions xix

Communications, Services, and Additional Information xxi

CHAPTER 1

Introduction to Secure Firewall ASA-Firewall Services 1

How to Implement Firewall Services 1

Basic Access Control 2

URL Filtering 2

Threat Protection 2

Firewall Services for Virtual Environments 3

Network Address Translation 3

Application Inspection 4

Use Case: Expose a Server to the Public 4

PART I

Access Control 7

CHAPTER 2

Objects for Access Control 9

Guidelines for Objects 9

Configure Objects 10

Configure Network Objects and Groups 10

Configure a Network Object 10

Configure a Network Object Group 11

Configure Service Objects and Service Groups 12

Configure a Service Object 12

Configure a Service Group 13
Configuring Network-Service Objects and Groups 15
Guidelines for Network-Service Objects 15
Configure Trusted DNS Servers 16
Configure Network-Service Objects 17
Configure Network-Service Object Groups 19
Configure Local User Groups 20
Configure Security Group Object Groups 21
Configure Time Ranges 22
Monitoring Objects 24
History for Objects 24

CHAPTER 3 **Access Control Lists** 27
About ACLs 27
ACL Types 27
ACL Names 29
Access Control Entry Order 29
Permit/Deny vs. Match/Do Not Match 29
Access Control Implicit Deny 29
IP Addresses Used for Extended ACLs When You Use NAT 30
Time-Based ACEs 31
Licensing for Access Control Lists 31
Guidelines for ACLs 31
Configure ACLs 32
Basic ACL Configuration and Management Options 32
Configure Extended ACLs 33
Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching 34
Add an Extended ACE for Port-Based Matching 35
Add an Extended ACE for ICMP-Based Matching 36
Add an Extended ACE for User-Based Matching (Identity Firewall) 37
Add an Extended ACE for Security Group-Based Matching (Cisco TrustSec) 38
Examples for Extended ACLs 39
Example of Converting Addresses to Objects for Extended ACLs 40
Configure Standard ACLs 41

Configure Webtype ACLs 41
Add a Webtype ACE for URL Matching 41
Add a Webtype ACE for IP Address Matching 43
Examples for Webtype ACLs 43
Configure EtherType ACLs 45
Examples for EtherType ACLs 46
Edit ACLs in an Isolated Configuration Session 46
Monitoring ACLs 48
History for ACLs 48

CHAPTER 4 **Access Rules** 51
Controlling Network Access 51
General Information About Rules 52
Interface Access Rules and Global Access Rules 52
Inbound and Outbound Rules 52
Rule Order 53
Implicit Permits 53
Implicit Deny 54
NAT and Access Rules 54
Same Security Level Interfaces and Access Rules 54
Extended Access Rules 55
Extended Access Rules for Returning Traffic 55
Allowing Broadcast and Multicast Traffic 55
Management Access Rules 56
EtherType Rules 56
Supported EtherTypes and Other Traffic 56
EtherType Rules for Returning Traffic 56
Allowing MPLS 57
Licensing for Access Rules 57
Guidelines for Access Control 57
Configure Access Control 59
Configure an Access Group 59
Configure ICMP Access Rules 60
Monitoring Access Rules 61

Evaluating Syslog Messages for Access Rules 62
Configuration Examples for Permitting or Denying Network Access 63
History for Access Rules 64

CHAPTER 5 ASA and Cisco TrustSec 67
About Cisco TrustSec 67
About SGT and SXP Support in Cisco TrustSec 68
Roles in the Cisco TrustSec Feature 68
Security Group Policy Enforcement 69
How the ASA Enforces Security Group-Based Policies 70
Effects of Changes to Security Groups on the ISE 71
Speaker and Listener Roles on the ASA 72
Register the ASA with the ISE 73
Create a Security Group on the ISE 74
Generate the PAC File 74
Guidelines for Cisco TrustSec 75
Configure the ASA to Integrate with Cisco Trustsec 77
Configure the AAA Server for Cisco TrustSec Integration 78
Import a PAC File 80
Configure the Security Exchange Protocol 81
Add an SXP Connection Peer 83
Refresh Environment Data 84
Configure the Security Policy 85
Configure Layer 2 Security Group Tagging Imposition 86
Usage Scenarios 87
Configure a Security Group Tag on an Interface 88
Configure IP-SGT Bindings Manually 90
Troubleshooting Tips 90
Example for Cisco TrustSec 91
Secure Client VPN Support for Cisco TrustSec 91
Add an SGT to Remote Access VPN Group Policies and Local Users 92
Monitoring Cisco TrustSec 93
History for Cisco TrustSec 94

CHAPTER 6 **Cisco Umbrella** 97

About Cisco Umbrella Connector 97

Cisco Umbrella Enterprise Security Policy 97

Cisco Umbrella Registration 98

Switching from Legacy API Token to API/Secret Keys 98

Licensing Requirements for Cisco Umbrella Connector 99

Guidelines and Limitations for Cisco Umbrella 99

Configure Cisco Umbrella Connector 101

Install the CA Certificate from the Cisco Umbrella Registration Server 102

Configure the Umbrella Connector Global Settings 103

Enable Umbrella in the DNS Inspection Policy Map 105

Verify the Umbrella Registration 106

Examples for the Umbrella Connector 107

Example: Enabling Umbrella on the Global DNS Inspection Policy 107

Example: Enabling Umbrella on an Interface with a Custom Inspection Policy 108

Example: Excluding Certain Hosts or Networks from Umbrella Globally 108

Monitoring the Umbrella Connector 109

Monitoring the Umbrella Service Policy Statistics 110

Monitoring Umbrella Syslog Messages 111

History for Cisco Umbrella Connector 112

PART II **Firewall Services for Virtual Environments** 115

CHAPTER 7 **Attribute-Based Access Control** 117

Guidelines for Attribute-Based Network Objects 117

Configure Attribute-Based Access Control 118

Configure Attributes for vCenter Virtual Machines 118

Configure a VM Attribute Agent 120

Configure Attribute-Based Network Objects 122

Configure Access Control Using Attribute-Based Network Objects 123

Monitoring Attribute-Based Network Objects 125

History for Attribute-Based Access Control 126

PART III Network Address Translation 127

CHAPTER 8 Network Address Translation (NAT) 129

Why use NAT? 129

NAT basics 130

NAT terminology 130

NAT types 130

Network object NAT and Twice NAT 131

Network object NAT 131

Twice NAT 131

Comparing Network object NAT and Twice NAT 132

NAT rule order 132

NAT interfaces 134

Guidelines for NAT 135

Firewall mode guidelines for NAT 135

IPv6 NAT guidelines 135

IPv6 NAT best practices 136

Additional guidelines for NAT 136

Network Object NAT Guidelines for Mapped Address Objects 139

Twice NAT Guidelines for Real and Mapped Address Objects 140

FQDN destination guidelines 141

Twice NAT Guidelines for Service Objects for Real and Mapped Ports 142

Dynamic NAT 142

About dynamic NAT 142

Dynamic NAT disadvantages and advantages 143

Configure Dynamic Network Object NAT 144

Configure Dynamic Twice NAT 146

Dynamic PAT 149

About dynamic PAT 149

Dynamic PAT disadvantages and advantages 150

PAT pool object guidelines 150

Configure Dynamic Network Object PAT 151

Configure Dynamic Twice PAT 153

Configure PAT with Port Block Allocation 157
Configure Per-Session PAT or Multi-Session PAT 159
Static NAT 160
About static NAT 161
Static NAT with port translation 161
One-to-many static NAT 162
Other mapping scenarios (not recommended) 163
Configure Static Network Object NAT or Static NAT-with-Port-Translation 164
Configure Static Twice NAT or Static NAT-with-Port-Translation 167
Identity NAT 170
Configure Identity Network Object NAT 170
Configure Identity Twice NAT 172
Monitoring NAT 174
History for NAT 175

CHAPTER 9 **NAT Examples and Reference** 181
Examples for Network Object NAT 181
Providing Access to an Inside Web Server (Static NAT) 181
NAT for Inside Hosts (Dynamic NAT) and NAT for an Outside Web Server (Static NAT) 182
Inside Load Balancer with Multiple Mapped Addresses (Static NAT, One-to-Many) 184
Single Address for FTP, HTTP, and SMTP (Static NAT-with-Port-Translation) 185
Examples for Twice NAT 186
Different Translation Depending on the Destination (Dynamic Twice PAT) 186
Different Translation Depending on the Destination Address and Port (Dynamic PAT) 188
NAT in routed and transparent mode 190
NAT in routed mode 190
NAT in transparent mode or within a bridge group 190
Routing NAT Packets 192
Mapped Addresses and Routing 192
Addresses on the same network as the mapped interface 192
Addresses on a unique network 192
The same address as the real address (identity NAT) 193
Transparent Mode Routing Requirements for Remote Networks 194
Determining the Egress Interface 194

NAT for VPN 195
NAT and Remote Access VPN 195
NAT and Site-to-Site VPN 197
NAT and VPN Management Access 200
Troubleshooting NAT and VPN 201
Translating IPv6 networks 201
NAT64/46: translating IPv6 addresses to IPv4 202
NAT64/46 Example: Inside IPv6 Network with Outside IPv4 Internet 202
NAT64/46 Example: Inside IPv6 Network with Outside IPv4 Internet and DNS Translation 203
NAT66: translating IPv6 addresses to different IPv6 addresses 205
NAT66 Example, Static Translation between Networks 205
NAT66 Example, Simple IPv6 Interface PAT 206
Rewriting DNS queries and responses using NAT 207
DNS Reply Modification, DNS Server on Outside 207
DNS Reply Modification, DNS Server, Host, and Server on Separate Networks 209
DNS Reply Modification, DNS Server on Host Network 209
DNS64 Reply Modification 210
PTR Modification, DNS Server on Host Network 212

CHAPTER 10 **Mapping Address and Port (MAP)** 213
About Mapping Address and Port (MAP) 213
About Mapping Address and Port Translation (MAP-T) 213
Guidelines for Mapping Address and Port (MAP) 214
Configure MAP-T Domains 216
Monitoring MAP 218
Verifying the MAP Domain Configuration 218
Monitoring MAP Syslog Messages 218
History for MAP 219

PART IV **Service Policies and Application Inspection** 221

CHAPTER 11 **Service Policy** 223
About Service Policies 223
The Components of a Service Policy 223

Features Configured with Service Policies 225
Feature Directionality 226
Feature Matching Within a Service Policy 226
Order in Which Multiple Feature Actions are Applied 227
Incompatibility of Certain Feature Actions 228
Feature Matching for Multiple Service Policies 229
Guidelines for Service Policies 229
Defaults for Service Policies 231
Default Service Policy Configuration 231
Default Class Maps (Traffic Classes) 231
Configure Service Policies 232
Identify Traffic (Layer 3/4 Class Maps) 234
Create a Layer 3/4 Class Map for Through Traffic 234
Create a Layer 3/4 Class Map for Management Traffic 237
Define Actions (Layer 3/4 Policy Map) 238
Apply Actions to an Interface (Service Policy) 239
Monitoring Service Policies 240
Examples for Service Policies (Modular Policy Framework) 240
Applying Inspection and QoS Policing to HTTP Traffic 240
Applying Inspection to HTTP Traffic Globally 241
Applying Inspection and Connection Limits to HTTP Traffic to Specific Servers 242
Applying Inspection to HTTP Traffic with NAT 243
History for Service Policies 243

CHAPTER 12 **Getting Started with Application Layer Protocol Inspection** 245
Application Layer Protocol Inspection 245
When to Use Application Protocol Inspection 245
Inspection Policy Maps 246
Replacing an In-Use Inspection Policy Map 246
How Multiple Traffic Classes are Handled 246
Guidelines for Application Inspection 247
Defaults for Application Inspection 249
Default Inspections and NAT Limitations 249
Default Inspection Policy Maps 253

Configure Application Layer Protocol Inspection 253
Choosing the Right Traffic Class for Inspection 258
Configure Regular Expressions 259
Create a Regular Expression 259
Create a Regular Expression Class Map 262
Monitoring Inspection Policies 263
History for Application Inspection 264

CHAPTER 13 **Inspection of Basic Internet Protocols** 265
DCERPC Inspection 265
DCERPC Overview 266
Configure a DCERPC Inspection Policy Map 266
DNS Inspection 268
Defaults for DNS Inspection 268
Configure DNS Inspection Policy Map 269
FTP Inspection 273
FTP Inspection Overview 273
Strict FTP 274
Configure an FTP Inspection Policy Map 275
HTTP Inspection 278
HTTP Inspection Overview 278
Configure an HTTP Inspection Policy Map 278
ICMP Inspection 282
ICMP Error Inspection 282
ILS Inspection 283
Instant Messaging Inspection 284
IP Options Inspection 286
Defaults for IP Options Inspection 287
Configure an IP Options Inspection Policy Map 287
IPsec Pass Through Inspection 289
IPsec Pass Through Inspection Overview 289
Configure an IPsec Pass Through Inspection Policy Map 289
IPv6 Inspection 290
Defaults for IPv6 Inspection 290

Configure an IPv6 Inspection Policy Map 291
NetBIOS Inspection 292
PPTP Inspection 293
RSH Inspection 294
SMTP and Extended SMTP Inspection 294
SMTP and ESMTP Inspection Overview 294
Defaults for ESMTP Inspection 295
Configure an ESMTP Inspection Policy Map 296
SNMP Inspection 298
SQL*Net Inspection 299
Sun RPC Inspection 300
Sun RPC Inspection Overview 300
Manage Sun RPC Services 300
TFTP Inspection 301
XDMCP Inspection 302
VXLAN Inspection 302
History for Basic Internet Protocol Inspection 302

CHAPTER 14 **Inspection for Voice and Video Protocols** 305
CTIQBE Inspection 305
Limitations for CTIQBE Inspection 305
H.323 Inspection 306
H.323 Inspection Overview 306
How H.323 Works 306
H.239 Support in H.245 Messages 307
Limitations for H.323 Inspection 308
Configure H.323 Inspection Policy Map 308
MGCP Inspection 311
MGCP Inspection Overview 311
Configure an MGCP Inspection Policy Map 312
RTSP Inspection 314
RTSP Inspection Overview 314
RealPlayer Configuration Requirements 314
Limitations for RSTP Inspection 314

Configure RTSP Inspection Policy Map 315
SIP Inspection 317
SIP Inspection Overview 317
Limitations for SIP Inspection 318
Default SIP Inspection 319
Configure SIP Inspection Policy Map 319
Skinny (SCCP) Inspection 323
SCCP Inspection Overview 323
Supporting Cisco IP Phones 323
Limitations for SCCP Inspection 324
Default SCCP Inspection 324
Configure a Skinny (SCCP) Inspection Policy Map 324
STUN Inspection 326
History for Voice and Video Protocol Inspection 326

CHAPTER 15 **Inspection for Mobile Networks** 329
Mobile Network Inspection Overview 329
GTP Inspection Overview 329
Tracking Location Changes for Mobile Stations 330
GTP Inspection Limitations 330
Stream Control Transmission Protocol (SCTP) Inspection and Access Control 330
SCTP Stateful Inspection 331
SCTP Access Control 332
SCTP NAT 332
SCTP Application Layer Inspection 332
SCTP Limitations 333
Diameter Inspection 333
M3UA Inspection 334
M3UA Protocol Conformance 334
M3UA Inspection Limitations 335
RADIUS Accounting Inspection Overview 335
Licensing for Mobile Network Protocol Inspection 336
Defaults for GTP Inspection 336
Configure Mobile Network Inspection 337

Configure a GTP Inspection Policy Map 337
Configure an SCTP Inspection Policy Map 342
Configure a Diameter Inspection Policy Map 343
Create a Custom Diameter Attribute-Value Pair (AVP) 347
Inspecting Encrypted Diameter Sessions 348
Configure Server Trust Relationship with Diameter Clients 350
Configure Full TLS Proxy with Static Client Certificate for Diameter Inspection 351
Configure Full TLS Proxy with Local Dynamic Certificates for Diameter Inspection 354
Configure TLS Proxy with TLS Offload for Diameter Inspection 358
Configure an M3UA Inspection Policy Map 359
Configure the Mobile Network Inspection Service Policy 363
Configure RADIUS Accounting Inspection 364
Configure a RADIUS Accounting Inspection Policy Map 365
Configure the RADIUS Accounting Inspection Service Policy 366
Monitoring Mobile Network Inspection 367
Monitoring GTP Inspection 367
Monitoring SCTP 369
Monitoring Diameter 370
Monitoring M3UA 370
History for Mobile Network Inspection 372

PART V **Connection Management and Threat Detection** 375

CHAPTER 16 **Connection Settings** 377
Connection settings 377
Configure Connection Settings 378
Configure Global Timeouts 379
Protect Servers from a SYN Flood DoS Attack (TCP Intercept) 381
Customize Abnormal TCP Packet Handling (TCP Maps, TCP Normalizer) 384
Bypass TCP state checks for asymmetrical routing (TCP state bypass) 387
The asymmetrical routing problem 388
Guidelines and limitations for TCP state bypass 389
Configure TCP State Bypass 389
Disable TCP Sequence Randomization 391

Offload Large Flows 392
Restrictions for flow offload implementation 393
Configure Flow Offload 394
IPsec flow offload 396
Limitations for IPsec flow offload 397
Configure IPsec Flow Offload 397
DTLS crypto acceleration 398
Configure DTLS Crypto Acceleration 398
Monitor DTLS crypto acceleration 399
Configure Connection Settings for Specific Traffic Classes (All Services) 400
Configure TCP Options 405
Monitoring Connections 406
History for Connection Settings 407

CHAPTER 17 **Quality of Service** 411
About QoS 411
Supported QoS Features 411
What is a Token Bucket? 412
Policing 412
Priority Queuing 412
How QoS Features Interact 413
DSCP (DiffServ) Preservation 413
Guidelines for QoS 413
Configure QoS 413
Determine the Queue and TX Ring Limits for a Priority Queue 414
Queue Limit Worksheet 414
TX Ring Limit Worksheet 414
Configure the Priority Queue for an Interface 415
Configure a Service Rule for Priority Queuing and Policing 416
Monitor QoS 419
QoS Police Statistics 419
QoS Priority Statistics 419
QoS Priority Queue Statistics 420
Configuration Examples for Priority Queuing and Policing 420

Class Map Examples for VPN Traffic 420
Priority and Policing Example 421
History for QoS 422

CHAPTER 18 **Threat Detection** 423
Detecting Threats 423
Basic Threat Detection Statistics 424
Advanced Threat Detection Statistics 425
Scanning Threat Detection 425
Guidelines for Threat Detection 426
Defaults for Threat Detection 426
Configure Threat Detection 427
Configure Basic Threat Detection Statistics 427
Configure Advanced Threat Detection Statistics 428
Configure Scanning Threat Detection 430
Configure Threat Detection for VPN Services 431
Monitoring Threat Detection 433
Monitoring Basic Threat Detection Statistics 433
Monitoring Advanced Threat Detection Statistics 434
Evaluating Host Threat Detection Statistics 435
Monitoring Scanning Threat Detection Shunned Hosts, Attackers, and Targets 438
Monitoring Threat Detection for VPN Services 438
Syslog Monitoring for Threat Detection VPN Services 438
Show Command Monitoring for Threat Detection for VPN Services 439
Removing Shuns Applied for VPN Service Violations 441
Examples for Threat Detection 441
History for Threat Detection 442



About This Guide

The following topics explain how to use this guide.

- Document Objectives, on page xix
- Related Documentation, on page xix
- Document Conventions, on page xix
- Communications, Services, and Additional Information, on page xxi

Document Objectives

The purpose of this guide is to help you configure the firewall features for the Secure Firewall ASA series using the command-line interface. This guide does not cover every feature, but describes only the most common configuration scenarios.

You can also configure and monitor the ASA by using the Adaptive Security Device Manager (ASDM), a web-based GUI application. ASDM includes configuration wizards to guide you through some common configuration scenarios, and online help for less common scenarios.

Throughout this guide, the term "ASA" applies generically to supported models, unless specified otherwise.

Related Documentation

For more information, see *Navigating the Cisco ASA Series Documentation* at http://www.cisco.com/go/asadocs.

Document Conventions

This document adheres to the following text, display, and alert conventions.

Text Conventions

Convention	Indication
boldface	Commands, keywords, button labels, field names, and user-entered text appear in **boldface**. For menu-based commands, the full path to the command is shown.

Convention	Indication
italic	Variables, for which you supply values, are presented in an *italic* typeface. Italic type is also used for document titles, and for general emphasis.
`monospace`	Terminal sessions and information that the system displays appear in `monospace` type.
{x | y | z}	Required alternative keywords are grouped in braces and separated by vertical bars.
[ ]	Elements in square brackets are optional.
[x | y | z]	Optional alternative keywords are grouped in square brackets and separated by vertical bars.
[ ]	Default responses to system prompts are also in square brackets.
< >	Non-printing characters such as passwords are in angle brackets.
!, #	An exclamation point (!) or a number sign (#) at the beginning of a line of code indicates a comment line.

Reader Alerts

This document uses the following for reader alerts:



Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.



Tip Means *the following information will help you solve a problem.*



Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.



Timesaver Means *the described action saves time.* You can save time by performing the action described in the paragraph.



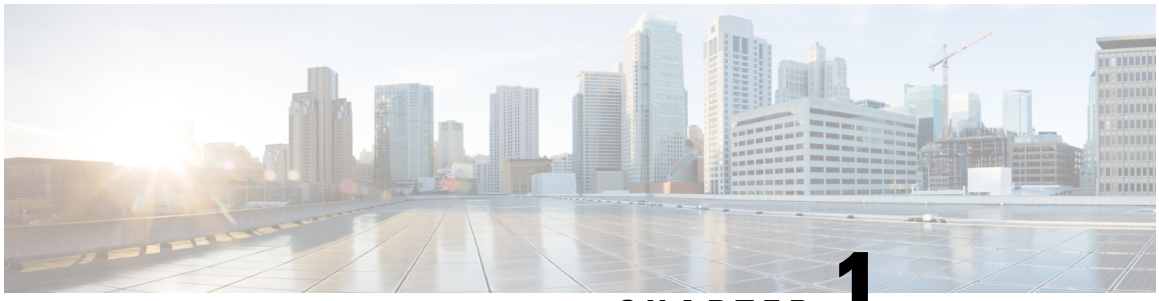
Warning **Means *reader be warned.* In this situation, you might perform an action that could result in bodily injury.**

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business results you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



CHAPTER 1

Introduction to Secure Firewall ASA-Firewall Services

Firewall services are those ASA features that are focused on controlling access to the network, including services that block traffic and services that enable traffic flow between internal and external networks. These services include those that protect the network against threats, such as Denial of Service (DoS) and other attacks.

The following topics provide an overview of firewall services.

- How to Implement Firewall Services, on page 1
- Basic Access Control, on page 2
- URL Filtering, on page 2
- Threat Protection, on page 2
- Firewall Services for Virtual Environments, on page 3
- Network Address Translation, on page 3
- Application Inspection, on page 4
- Use Case: Expose a Server to the Public, on page 4

How to Implement Firewall Services

The following procedure provides a general sequence for implementing firewall services. However, each step is optional, needed only if you want to provide the service to your network.

Before you begin

Configure the ASA according to the general operations configuration guide, including at minimum basic settings, interface configuration, routing, and management access.

Procedure

Step 1 Implement access control for the network. See Basic Access Control, on page 2.

Step 2 Implement URL filtering. See URL Filtering, on page 2.

Step 3 Implement threat protection. See Threat Protection, on page 2.

Step 4 Implement firewall services that are tailored to virtual environments. See Firewall Services for Virtual Environments, on page 3.

Step 5 Implement Network Address Translation (NAT). See Network Address Translation, on page 3.

Step 6 Implement application inspection if the default settings are insufficient for your network. See Application Inspection, on page 4.

Basic Access Control

Access rules, applied per interface or globally, are your first line of defense. You can drop, upon entry, specific types of traffic, or traffic from (or to) specific hosts or networks. By default, the ASA allows traffic to flow freely from an inside network (higher security level) to an outside network (lower security level).

You can apply an access rule to limit traffic from inside to outside, or allow traffic from outside to inside.

Basic access rules control traffic using a “5-tuple” of source address and port, destination address and port, and protocol. See Access Rules, on page 51 and Access Control Lists, on page 27.

You can augment your rules by making them identity aware. To implement identity control, install Cisco Identity Services Engine (ISE) on a separate server to implement Cisco Trustsec. You can then add security group criteria to your access rules. See ASA and Cisco TrustSec, on page 67.

URL Filtering

URL filtering denies or allows traffic based on the URL of the destination site.

To implement URL filtering, subscribe to the Cisco Umbrella service, where you configure the Enterprise Security policy to block malicious sites based on the fully-qualified domain name (FQDN). For FQDNs that are considered suspicious, you can redirect user connections to the Cisco Umbrella intelligent proxy, which performs URL filtering. The Umbrella service works by handling users' DNS lookup requests, returning the IP address for a block page or the IP address of the intelligent proxy. The service returns the real IP address for an FQDN for allowed domains. See Cisco Umbrella, on page 97.

Threat Protection

You can implement a number of measures to protect against scanning, denial of service (DoS), and other attacks. A number of ASA features help protect against attacks by applying connection limits and dropping abnormal TCP packets. Some features are automatic, others are configurable but have defaults appropriate in most cases, while others are completely optional and you must configure them if you want them.

Following are the threat protection services available with the ASA.

- IP packet fragmentation protection—The ASA performs full reassembly of all ICMP error messages and virtual reassembly of the remaining IP fragments that are routed through the ASA, and drops fragments that fail the security check. No configuration is necessary.
- Connection limits, TCP normalization, and other connection-related features—Configure connection-related services such as TCP and UDP connection limits and timeouts, TCP sequence number

randomization, TCP normalization, and TCP state bypass. TCP normalization is designed to drop packets that do not appear normal. See Connection Settings, on page 377.

For example, you can limit TCP and UDP connections and embryonic connections (a connection request that has not finished the necessary handshake between source and destination). Limiting the number of connections and embryonic connections protects you from a DoS attack. The ASA uses the embryonic limit to trigger TCP Intercept, which protects inside systems from a DoS attack perpetrated by flooding an interface with TCP SYN packets.

- Threat detection—Implement threat detection on the ASA to collect statistics to help identify attacks. Basic threat detection is enabled by default, but you can implement advanced statistics and scanning threat detection. You can shun hosts that are identified as a scanning threat. See Threat Detection, on page 423.

Firewall Services for Virtual Environments

Virtual environments deploy servers as virtual machines, for example, in VMware ESXi. The firewalls in a virtual environment can be traditional hardware devices, or they can also be virtual machine firewalls, such as the ASA Virtual.

Traditional and next-generation firewall services apply to virtual environments in the same way that they apply to environments that do not use virtual machine servers. However, virtual environments can provide additional challenges, because it is easy to create and tear down servers.

Additionally, traffic between servers within the data center might require as much protection as traffic between the data center and external users. For example, if an attacker gains control of a server within the data center, that could open up attacks on other servers in the data center.

Firewall services for virtual environments add capabilities to apply firewall protection specifically to virtual machines. Following are the firewall services available for virtual environments:

- Attribute-based access control—You can configure network objects to match traffic based on attributes, and use those objects in access control rules. This lets you decouple firewall rules from network topology. For example, you can allow all hosts with the Engineering attribute to access hosts with the Lab Server attribute. You could then add/remove hosts with these attributes and the firewall policy would be applied automatically without the need for updating access rules. For more information, see Attribute-Based Access Control, on page 117.

Network Address Translation

One of the main functions of Network Address Translation (NAT) is to enable private IP networks to connect to the Internet. NAT replaces a private IP address with a public IP address, translating the private addresses in the internal private network into legal, routable addresses that can be used on the public Internet. In this way, NAT conserves public addresses because you can advertise at a minimum only one public address for the entire network to the outside world.

Other functions of NAT include:

- Security—Keeping internal IP addresses hidden discourages direct attacks.
- IP routing solutions—Overlapping IP addresses are not a problem when you use NAT.

- Flexibility—You can change internal IP addressing schemes without affecting the public addresses available externally; for example, for a server accessible to the Internet, you can maintain a fixed IP address for Internet use, but internally, you can change the server address.
- Translating between IPv4 and IPv6 (Routed mode only)—If you want to connect an IPv6 network to an IPv4 network, NAT lets you translate between the two types of addresses.

NAT is not required. If you do not configure NAT for a given set of traffic, that traffic will not be translated, but will have all of the security policies applied as normal.

See:

- Network Address Translation (NAT), on page 129
- NAT Examples and Reference, on page 181

Application Inspection

Application inspection engines are required for services that embed IP addressing information in the user data packet or that open secondary channels on dynamically assigned ports. These protocols require the ASA to do a deep packet inspection, to open the required pinholes and to apply network address translation (NAT).

The default ASA policy already applies inspection globally for many popular protocols, such as DNS, FTP, SIP, ESMTP, TFTP, and others. The default inspections might be all you require for your network.

However, you might need to enable inspection for other protocols, or fine-tune an inspection. Many inspections include detailed options that let you control packets based on their contents. If you know a protocol well, you can apply fine-grained control on that traffic.

You use service policies to configure application inspection. You can configure a global service policy, or apply a service policy to each interface, or both.

See:

- Service Policy, on page 223
- Getting Started with Application Layer Protocol Inspection, on page 245
- Inspection of Basic Internet Protocols, on page 265
- Inspection for Voice and Video Protocols, on page 305
- Inspection for Mobile Networks, on page 329.

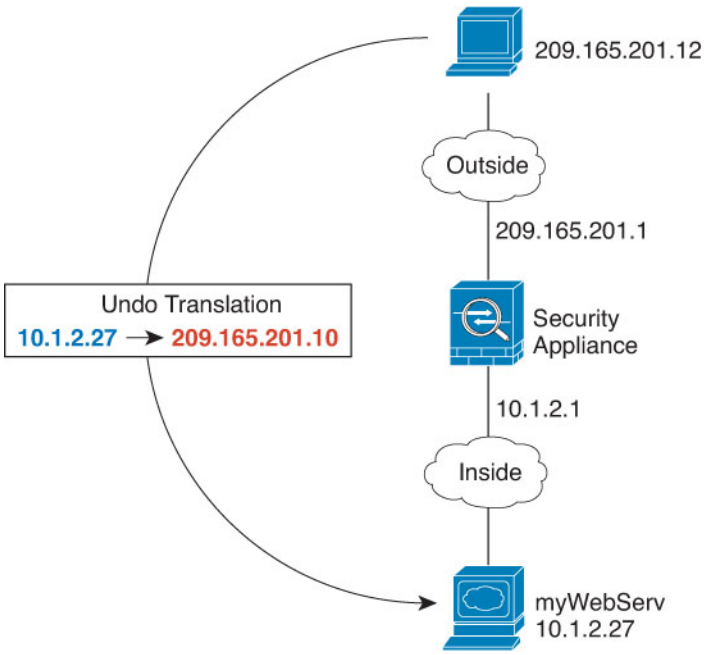
Use Case: Expose a Server to the Public

You can make certain application services on a server available to the public. For example, you could expose a web server, so that users can connect to the web pages but not make any other connections to the server.

To expose a server to the public, you typically need to create access rules that allow the connection and NAT rules to translate between the server's internal IP address and an external address that the public can use. In addition, you can use port address translation (PAT) to map an internal port to an external port, if you do not want the externally exposed service to use the same port as the internal server. For example, if the internal web server is not running on TCP/80, you can map it to TCP/80 to make connections easier for external users.

The following example makes a web server on the inside private network available for public access.

Figure 1: Static NAT for an Inside Web Server



Procedure

Step 1 Create a network object for the internal web server.

```
hostname(config)# object network myWebServ
hostname(config-network-object)# host 10.1.2.27
```

Step 2 Configure static NAT for the object:

```
hostname(config-network-object)# nat (inside,outside) static 209.165.201.10
```

Step 3 Add an access rule to the access group attached to the outside interface to permit web access to the server.

```
hostname(config)# access-list outside_access_in line 1 extended
permit tcp any4 object myWebServ eq http
```

Step 4 If you do not already have an access group on the outside interface, apply it using the access-group command:

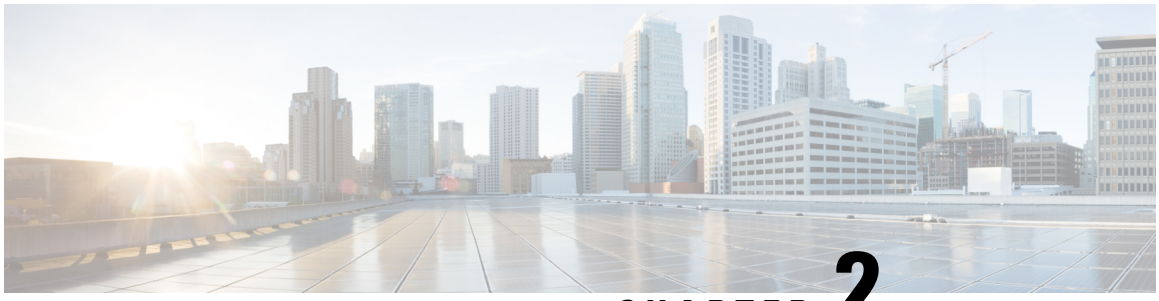
```
hostname(config)# access-group outside_access_in in interface outside
```



PART I

Access Control

- Objects for Access Control, on page 9
- Access Control Lists, on page 27
- Access Rules, on page 51
- ASA and Cisco TrustSec, on page 67
- Cisco Umbrella, on page 97



CHAPTER 2

Objects for Access Control

Objects are reusable components for use in your configuration. You can define and use them in ASA configurations in the place of inline IP addresses, services, names, and so on. Objects make it easy to maintain your configurations because you can modify an object in one place and have it be reflected in all other places that are referencing it. Without objects you would have to modify the parameters for every feature when required, instead of just once. For example, if a network object defines an IP address and subnet mask, and you want to change the address, you only need to change it in the object definition, not in every feature that refers to that IP address.

- Guidelines for Objects, on page 9
- Configure Objects, on page 10
- Monitoring Objects, on page 24
- History for Objects, on page 24

Guidelines for Objects

IPv6 Guidelines

Supports IPv6 with the following restrictions:

- You can mix IPv4 and IPv6 entries in a network object group, but you cannot use a mixed object group for NAT.

Additional Guidelines and Limitations

- Objects must have unique names, because objects and object groups share the same name space. While you might want to create a network object group named “Engineering” and a service object group named “Engineering,” you need to add an identifier (or “tag”) to the end of at least one object group name to make it unique. For example, you can use the names “Engineering_admins” and “Engineering_hosts” to make the object group names unique and to aid in identification.
- Object names are limited to 64 characters, including letters, numbers, and these characters: .!@#$%^&()-_{}. Object names are case-sensitive.

Configure Objects

The following sections describe how to configure objects that are primarily used on access control.

Configure Network Objects and Groups

Network objects and groups identify IP addresses or host names. Use these objects in access control lists to simplify your rules.

Configure a Network Object

A network object can contain a host, a network IP address, a range of IP addresses, or a fully qualified domain name (FQDN).

You can also enable NAT rules on the object (excepting FQDN objects). For more information about configuring object NAT, see Network Address Translation (NAT), on page 129.

Procedure

Step 1 Create or edit a network object using the object name: **object network** *object_name*

Example:

```
hostname(config)# object network email-server
```

Step 2 Add an address to the object using one of the following commands. Use the **no** form of the command to remove the object.

- **host** {*IPv4_address* | *IPv6_address*}—The IPv4 or IPv6 address of a single host. For example, 10.1.1.1 or 2001:DB8::0DB8:800:200C:417A.
- **subnet** {*IPv4_address IPv4_mask* | *IPv6_address/IPv6_prefix*}—The address of a network. For IPv4 subnets, include the mask after a space, for example, 10.0.0.0 255.0.0.0. For IPv6, include the address and prefix as a single unit (no spaces), such as 2001:DB8:0:CD30::/60.
- **range** *start_address end_address*—A range of addresses. You can specify IPv4 or IPv6 ranges. Do not include masks or prefixes.
- **fqdn** [**v4** | **v6**] *fully_qualified_domain_name*—A fully-qualified domain name, that is, the name of a host, such as www.example.com. Specify **v4** to limit the address to IPv4, and **v6** for IPv6. If you do not specify an address type, IPv4 is assumed.

Example:

```
hostname(config-network-object)# host 10.2.2.2
```

Step 3 (Optional) Add a description: **description** *string*

Configure a Network Object Group

Network object groups can contain multiple network objects as well as inline networks or hosts. Network object groups can include a mix of both IPv4 and IPv6 addresses.

However, you cannot use a mixed IPv4 and IPv6 object group for NAT, or object groups that include FQDN objects.

Procedure

Step 1 Create or edit a network object group using the object name: **object-group network** *group_name*

Example:

```
hostname(config)# object-group network admin
```

Step 2 Add objects and addresses to the network object group using one or more of the following commands. Use the **no** form of the command to remove an object.

- **network-object host** {*IPv4_address* | *IPv6_address*}—The IPv4 or IPv6 address of a single host. For example, 10.1.1.1 or 2001:DB8::0DB8:800:200C:417A.
- **network-object** {*IPv4_address IPv4_mask* | *IPv6_address/IPv6_prefix*}—The address of a network or host. For IPv4 subnets, include the mask after a space, for example, 10.0.0.0 255.0.0.0. For IPv6, include the address and prefix as a single unit (no spaces), such as 2001:DB8:0:CD30::/60.
- **network-object object** *object_name*—The name of an existing network object.
- **group-object** *object_group_name*—The name of an existing network object group.

Example:

```
hostname(config-network-object-group)# network-object 10.1.1.0 255.255.255.0
hostname(config-network-object-group)# network-object 2001:db8:0:cd30::/60
hostname(config-network-object-group)# network-object host 10.1.1.1
hostname(config-network-object-group)# network-object host 2001:DB8::0DB8:800:200C:417A
hostname(config-network-object-group)# network-object object existing-object-1
hostname(config-network-object-group)# group-object existing-network-object-group
```

Step 3 (Optional) Add a description: **description** *string*

Examples

To create a network group that includes the IP addresses of three administrators, enter the following commands:

```
hostname (config)# object-group network admins
hostname (config-protocol)# description Administrator Addresses
hostname (config-protocol)# network-object host 10.2.2.4
hostname (config-protocol)# network-object host 10.2.2.78
```

```
hostname (config-protocol)# network-object host 10.2.2.34
```

Create network object groups for privileged users from various departments by entering the following commands:

```
hostname (config)# object-group network eng
hostname (config-network)# network-object host 10.1.1.5
hostname (config-network)# network-object host 10.1.1.9
hostname (config-network)# network-object host 10.1.1.89

hostname (config)# object-group network hr
hostname (config-network)# network-object host 10.1.2.8
hostname (config-network)# network-object host 10.1.2.12

hostname (config)# object-group network finance
hostname (config-network)# network-object host 10.1.4.89
hostname (config-network)# network-object host 10.1.4.100
```

You then nest all three groups together as follows:

```
hostname (config)# object-group network admin
hostname (config-network)# group-object eng
hostname (config-network)# group-object hr
hostname (config-network)# group-object finance
```

Configure Service Objects and Service Groups

Service objects and groups identify protocols and ports. Use these objects in access control lists to simplify your rules.

Configure a Service Object

A service object can contain a single protocol specification.

Procedure

Step 1 Create or edit a service object using the object name: **object service** *object_name*

Example:

```
hostname(config)# object service web
```

Step 2 Add a service to the object using one of the following commands. Use the **no** form of the command to remove an object.

- **service** *protocol*—The name or number (0-255) of an IP protocol. Specify **ip** to apply to all protocols.
- **service** {**icmp** | **icmp6**} [*icmp-type* [*icmp_code*]]—For ICMP or ICMP version 6 messages. You can optionally specify the ICMP type by name or number (0-255) to limit the object to that message type. If you specify a type, you can optionally specify an ICMP code for that type (1-255). If you do not specify the code, then all codes are used.

- **service** {**tcp** | **udp** | **sctp**} [**source** *operator port*] [**destination** *operator port*]—For TCP, UDP or SCTP. You can optionally specify ports for the source, destination, or both. You can specify the port by name or number. The operator can be one of the following:
  - **lt**—less than.
  - **gt**—greater than.
  - **eq**—equal to.
  - **neq**—not equal to.
  - **range**—an inclusive range of values. When you use this operator, specify two port numbers, for example, **range 100 200**.

Example:

```
hostname(config-service-object)# service tcp destination eq http
```

Step 3 (Optional) Add a description: **description** *string*

Configure a Service Group

A service object group includes a mix of protocols, if desired, including optional source and destination ports for protocols that use them, and ICMP type and code.

Before you begin

You can model all services using the generic service object group, which is explained here. However, you can still configure the types of service group objects that were available prior to ASA 8.3(1). These legacy objects include TCP/UDP/TCP-UDP port groups, protocol groups, and ICMP groups. The contents of these groups are equivalent to the associated configuration in the generic service object group, with the exception of ICMP groups, which do not support ICMP6 or ICMP codes. If you still want to use these legacy objects, for detailed instructions, see the **object-service** command description in the command reference on Cisco.com.

Procedure

Step 1 Create or edit a service object group using the object name: **object-group service** *object_name*

Example:

```
hostname(config)# object-group service general-services
```

Step 2 Add objects and services to the service object group using one or more of the following commands. Use the **no** form of the command to remove an object.

- **service-object** *protocol*—The name or number (0-255) of an IP protocol. Specify **ip** to apply to all protocols.

- **service-object** {**icmp** | **icmp6**} [*icmp-type* [*icmp_code*]]—For ICMP or ICMP version 6 messages. You can optionally specify the ICMP type by name or number (0-255) to limit the object to that message type. If you specify a type, you can optionally specify an ICMP code for that type (1-255). If you do not specify the code, then all codes are used.
- **service-object** {**tcp** | **udp** | **tcp-udp** | **sctp**} [**source** *operator port*] [**destination** *operator port*]—For TCP, UDP, or both, or for SCTP. You can optionally specify ports for the source, destination, or both. You can specify the port by name or number. The operator can be one of the following:
  - **lt**—less than.
  - **gt**—greater than.
  - **eq**—equal to.
  - **neq**—not equal to.
  - **range**—an inclusive range of values. When you use this operator, specify two port numbers, for example, **range 100 200**.
- **service-object object** *object_name*—The name of an existing service object.
- **group-object** *object_group_name*—The name of an existing service object group.

Example:

```
hostname(config-service-object-group)# service-object ipsec
hostname(config-service-object-group)# service-object tcp destination eq domain
hostname(config-service-object-group)# service-object icmp echo
hostname(config-service-object-group)# service-object object my-service
hostname(config-service-object-group)# group-object Engineering_groups
```

Step 3 (Optional) Add a description: **description** *string*

Examples

The following example shows how to add both TCP and UDP services to a service object group:

```
hostname(config)# object-group service CommonApps
hostname(config-service-object-group)# service-object tcp destination eq ftp
hostname(config-service-object-group)# service-object tcp-udp destination eq www
hostname(config-service-object-group)# service-object tcp destination eq h323
hostname(config-service-object-group)# service-object tcp destination eq https
hostname(config-service-object-group)# service-object udp destination eq ntp
```

The following example shows how to add multiple service objects to a service object group:

```
hostname(config)# object service SSH
hostname(config-service-object)# service tcp destination eq ssh
hostname(config)# object service EIGRP
hostname(config-service-object)# service eigrp
hostname(config)# object service HTTPS
hostname(config-service-object)# service tcp source range 1 1024 destination eq https
hostname(config)# object-group service Group1
```

```
hostname(config-service-object-group)# service-object object SSH
hostname(config-service-object-group)# service-object object EIGRP
hostname(config-service-object-group)# service-object object HTTPS
```

Configuring Network-Service Objects and Groups

A network-service object or group defines an application. An application can consist of a DNS domain name (such as example.com), IP subnet, and optionally, protocol and port, such as TCP/80. Thus, a network-service object or group can combine the contents of separate network and service objects into a single object.

You can use the network-service object group in an extended ACL for use with routes maps (in policy-based routing), access control rules, and VPN filter. Note that you cannot directly use a network-service object (not group) in an ACL: you must first put objects in a group object, then you can use the group object.

When you use domain name specifications, the system uses DNS snooping to get the IP addresses that are obtained through the user's DNS request prior to the start of the connection. This ensures that an IP address is available at the start of a connection, so that the connection is handled correctly, by route maps and access control rules, from the first packet.

Guidelines for Network-Service Objects

- DNS inspection is required if you include DNS domain name specifications in a network-service object. DNS inspection is enabled by default. Do not disable it if you use network-service objects.
- DNS snooping is done on UDP DNS packets only, it is not done on TCP or HTTP DNS packets. Unlike fully-qualified domain name objects, network-service domain specifications are snooped immediately, even if you do not use the object in an access list.
- You cannot enable dnscrypt in the DNS inspection policy map; it is not compatible with the DNS snooping that is required to obtain IP addresses for domains used in network-service objects. Any network-service objects that include domain specifications will become inoperable and the related access control entries will not be matched.
- You can define a maximum of 1024 network-service groups. However, this limit is shared with identity firewall local user groups. For each network-service group defined, you can create 2 fewer user groups.
- The contents of network-service groups can overlap, but you cannot create a complete duplicate of a network-service group.
- If a network-service object or group is being used in an ACL, deleting the object simply clears the object contents. The object itself remains defined in the configuration.
- The **dynamic** keyword on an object or group indicates that the object will not be saved to the running configuration, it will be shown in the **show object** output only. Do not directly configure this keyword. It is primarily for use by external device managers.
- The **app-id** command for a network-service object defines a Cisco-assigned number for a particular application, in the range 1-4294967295. Do not directly configure this command. This command is mainly for the use of external device managers.

Configure Trusted DNS Servers

If you configure domain names in network-service objects, the system snoops DNS request/response traffic to gather IP addresses for DNS domain names and caches the results. Any DNS request/response can be snooped.

The records snooped are A, AAAA, and MX. The time-to-live (TTL) of each resolved name is honored within limits: the minimum TTL is 2 minutes, the maximum is 24 hours. This ensures that the cache does not become stale.

For security reasons, you can limit the scope of DNS snooping by defining which DNS servers should be trusted. Any DNS traffic to non-trusted DNS servers is ignored and not used to obtain mappings for network-service objects. By default, all configured and learned DNS servers are trusted; you need to change this only if you want to limit the trusted list.

Note The key is that you define the DNS servers used by your clients as trusted servers.

Before you begin

DNS snooping depends on DNS inspection, which is enabled by default. Ensure that you do not disable the inspection. In addition, DSN snooping is incompatible with the **dnscrypt** feature, so do not enable that command in the DNS inspection policy map.

Procedure

Step 1 Use the **show dns trusted-source detail** command to determine the current trusted servers that have been downloaded to the data path for use with network-service object domain resolution.

The default is to trust any DNS server that you configure in a DNS group, or that is configured through DHCP client/server or relay. This command shows you the current settings and the servers being trusted.

Example:

```
ciscoasa# show dns trusted-source detail
DNS Trusted Source enabled for DHCP Server Configured
DNS Trusted Source enabled for DHCP Client Learned
DNS Trusted Source enabled for DHCP Relay Learned
DNS Trusted Source enabled for DNS Server Configured
DNS Trusted Source not enabled for Trust-any
DNS Trusted Source: Type: IPs : Interface : Idle/Timeout (sec)
    DNS Server Configured: 10.163.47.11: management : N/A
    DNS Server Configured: 10.37.137.85: management : N/A
    DNS Server Configured: 10.37.142.73: management : N/A
Data-Path DNS Trusted Source (count 3): <ip>/<refcnt>; Trust-any disabled
  10.37.142.73/1
  10.37.137.85/1
  10.163.47.11/1
```

Step 2 (Optional.) Add or remove explicitly-configured trusted DNS servers:

dns trusted-source *ip_list*

The *ip_list* is a space separated list of the IP addresses of DNS servers that should be trusted. You can list up to 12 IPv4 and IPv6 addresses. Specify **any** to cover all DNS servers. Use the **no** form of the command to remove a server.

Step 3 (Optional.) Specify whether servers configured in DNS server groups should be trusted.

dns trusted-source configured-servers

A configured server is any server specified in DNS groups or name-server commands. This option is enabled by default. Use the **no** form of the command to disable it.

Step 4 (Optional.) Specify whether the DNS servers that are configured in the DHCP pools for clients that obtain addresses through DHCP servers running on the device interfaces should be trusted.

dns trusted-source dhcp-pools

These are the servers that are configured on the **dhcpd dns** command, and thus are IPv4 only. This option is enabled by default. Use the **no** form of the command to disable it.

Step 5 (Optional.) Specify whether the servers that are learned by snooping DHCP relay messages between a DHCP client and DHCP server are considered trusted DNS servers.

dns trusted-source dhcp-relay

This option is enabled by default. Use the **no** form of the command to disable it.

Step 6 (Optional.) Specify whether the servers that are learned by snooping messages between a DHCP client and DHCP server are considered trusted DNS servers.

dns trusted-source dhcp-client

This option applies when you configure the **dhcpd auto_config** command to configure DHCP servers on inside interfaces using the information obtained from device interfaces that use DHCP client to obtain an IP address. This option is enabled by default. Use the **no** form of the command to disable it.

Configure Network-Service Objects

A network-service object defines a single application. It defines the application location either by subnet specification or more commonly, DNS domain name. Optionally, you can include protocol and port to narrow the scope of the application.

You can use these objects in network-service group objects only; you cannot directly use a network-service object in an access control list entry (ACE).

Procedure

Step 1 Create or edit a network-service object using the object name:

object network-service *object_name*

The name can be up to 128 characters, and can include spaces. If you include spaces, you must enclose the name in double quotation marks.

Example:

```
ciscoasa(config)# object network-service webex
```

Step 2 Add one or more application locations and optional services to the object using one of the following commands. Use the **no** form of the command to remove the location. You can enter these commands multiple times.

- **domain** *domain_name* [*service*]—The DNS name, up to 253 characters. This can be fully-qualified (such as www.example.com) or partial (such as example.com), in which case the object matches all subdomains, that is, servers with the partial name (such as www.example.com, www1.example.com, long.server.name.example.com, and so forth). Connections will be matched against the longest name if an exact match is available. The domain name can resolve to multiple IP addresses.
- **subnet** {*IPv4_address IPv4_mask* | *IPv6_address/IPv6_prefix*} [*service*]—The address of a network. For IPv4 subnets, include the mask after a space, for example, 10.0.0.0 255.0.0.0. For IPv6, include the address and prefix as a single unit (no spaces), such as 2001:DB8:0:CD30::/60.

The service specification for these commands is the same. Specify the service only if you want to limit the scope of the connections matched. By default, any connection to the resolved IP addresses matches the object.

protocol [*operator port*]

where:

- *protocol* is the protocol used in the connection, such as tcp, udp, ip, and so forth. Use ? to see the list of protocols.
- (TCP/UDP only.) *operator* is one of the following:
  - **eq** equals the port number specified.
  - **lt** means any port less than the specified port number.
  - **gt** means any port greater than the specified port number.
  - **range** means any port between the two ports specified.
- (TCP/UDP only.) *port* is the port number, 1-65535 or a mnemonic, such as www. Use ? to see the mnemonics. For ranges, you must specify two ports, with the first port being a lower number than the second port.

Step 3 (Optional) Add a description (up to 200 characters): **description** *string*

Examples

```
object network-service outlook365
    description This defines Microsoft office365 'outlook' application.
  domain outlook.office.com tcp eq 443
object network-service webex
  domain webex.com tcp eq 443
object network-service partner
  subnet 10.34.56.0 255.255.255.0 ip
```

Configure Network-Service Object Groups

Network-service groups can contain network-service objects and explicit subnet or domain specifications. You can use network-service objects in access control list entries (ACEs) for policy-based routing, access control, and VPN filter.

Use network-service groups to define a category of applications that should be handled in the same manner. For example, you could create a single group that defines the applications whose traffic should be directed to the Internet rather than to the site-to-site VPN tunnel to the corporate hub.

There is no limit to how many applications you include in a network-service object group, either explicitly or by reference to network-service objects.

Procedure

Step 1 Create or edit a network-service object group using the group name:

object-group network-service *group_name*

The name can be up to 128 characters, and can include spaces. If you include spaces, you must enclose the name in double quotation marks.

Example:

```
ciscoasa(config)# object-group network-service SaaS_Applications
```

Step 2 Add one or more application locations and optional services to the object using one of the following commands. Use the **no** form of the command to remove the location. You can enter these commands multiple times.

- **network-service-member** *object_name*—The name of a network-service object to include in the group. If there are spaces in the name, enclose the name in double quotation marks.
- **domain** *domain_name* [*service*]—The DNS name, up to 253 characters. This can be fully-qualified (such as www.example.com) or partial (such as example.com), in which case the object matches all subdomains, that is, servers with the partial name (such as www.example.com, www1.example.com, long.server.name.example.com, and so forth). Connections will be matched against the longest name if an exact match is available. The domain name can resolve to multiple IP addresses.
- **subnet** {*IPv4_address IPv4_mask* | *IPv6_address/IPv6_prefix*} [*service*]—The address of a network. For IPv4 subnets, include the mask after a space, for example, 10.0.0.0 255.0.0.0. For IPv6, include the address and prefix as a single unit (no spaces), such as 2001:DB8:0:CD30::/60.

The service specification for these commands is the same. Specify the service only if you want to limit the scope of the connections matched. By default, any connection to the resolved IP addresses matches the object.

protocol [*operator port*]

where:

- *protocol* is the protocol used in the connection, such as tcp, udp, ip, and so forth. Use ? to see the list of protocols.
- (TCP/UDP only.) *operator* is one of the following:
  - **eq** equals the port number specified.

- **lt** means any port less than the specified port number.
- **gt** means any port greater than the specified port number.
- **range** means any port between the two ports specified.

- (TCP/UDP only.) *port* is the port number, 1-65535 or a mnemonic, such as www. Use ? to see the mnemonics. For ranges, you must specify two ports, with the first port being a lower number than the second port.

Step 3 (Optional) Add a description (up to 200 characters): **description** *string*

Examples

Configure a set of SaaS applications using previously-defined network-service objects.

```
object-group network-service SaaS_Applications
    description This group includes relevant 'Software as a Service' applications
    network-service-member "outlook 365"
    network-service-member webex
    network-service-member box
```

Configure Local User Groups

You can create local user groups for use in features that support the identity firewall by including the group in an extended ACL, which in turn can be used in an access rule, for example.

The ASA sends an LDAP query to the Active Directory server for user groups globally defined in the Active Directory domain controller. The ASA imports these groups for identity-based rules. However, the ASA might have localized network resources that are not defined globally that require local user groups with localized security policies. Local user groups can contain nested groups and user groups that are imported from Active Directory. The ASA consolidates local and Active Directory groups.

A user can belong to local user groups and user groups imported from Active Directory.

Because you can use usernames and user group names directly in an ACL, you need to configure local user groups only if:

- You want to create a group of users defined in the LOCAL database.
- You want to create a group of users or user groups that are not captured in a single user group defined on the AD server.

Procedure

Step 1 Create or edit a user object group using the object name: **object-group user** *group_name*

Example:

```
hostname(config)# object-group user admins
```

Step 2 Add users and groups to the user object group using one or more of the following commands. Use the **no** form of the command to remove an object.

- **user** [*domain_NETBIOS_name*\]*username*—A username. If there is a space in the domain name or username, you must enclose the domain name and user name in quotation marks. The domain name can be LOCAL (for users defined in the local database) or an Active Directory (AD) domain name as specified in the **user-identity domain** *domain_NetBIOS_name* **aaa-server** *aaa_server_group_tag* command. When adding users defined in an AD domain, the *user_name* must be the Active Directory sAMAccountName, which is unique, instead of the common name (cn), which might not be unique. If you do not specify a domain name, the default is used, which is either LOCAL or the one defined on the **user-identity default-domain** command.
- **user-group** [*domain_NETBIOS_name*\\]*username*—A user group. If there is a space in the domain name or group name, you must enclose the domain name and group name in quotation marks. Note the double \\ that separates the domain and group names.
- **group-object** *object_group_name*—The name of an existing user object group.

Example:

```
hostname(config-user-object-group)# user EXAMPLE\admin
hostname(config-user-object-group)# user-group EXAMPLE\\managers
hostname(config-user-object-group)# group-object local-admins
```

Step 3 (Optional) Add a description: **description** *string*

Configure Security Group Object Groups

You can create security group object groups for use in features that support Cisco TrustSec by including the group in an extended ACL, which in turn can be used in an access rule, for example.

When integrated with Cisco TrustSec, the ASA downloads security group information from the ISE. The ISE acts as an identity repository, by providing Cisco TrustSec tag-to-user identity mapping and Cisco TrustSec tag-to-server resource mapping. You provision and manage security group ACLs centrally on the ISE.

However, the ASA might have localized network resources that are not defined globally that require local security groups with localized security policies. Local security groups can contain nested security groups that are downloaded from the ISE. The ASA consolidates local and central security groups.

To create local security groups on the ASA, you create a local security object group. A local security object group can contain one or more nested security object groups or Security IDs or security group names. You can also create a new Security ID or security group name that does not exist on the ASA.

You can use the security object groups you create on the ASA to control access to network resources. You can use the security object group as part of an access group or service policy.

Tip If you create a group with tags or names that are not known to the ASA, any rules that use the group will be inactive until the tags or names are resolved with ISE.

Procedure

Step 1 Create or edit a security group object group using the object name: **object-group security** *group_name*

Example:

```
hostname(config)# object-group security mktg-sg
```

Step 2 Add objects to the service group object group using one or more of the following commands. Use the **no** form of the command to remove an object.

- **security-group** {**tag** *sgt_number* | **name** *sg_name*}—A security group tag (SGT) or name. A tag is a number from 1 to 65533 and is assigned to a device through IEEE 802.1X authentication, web authentication, or MAC authentication bypass (MAB) by the ISE. Security group names are created on the ISE and provide user-friendly names for security groups. The security group table maps SGTs to security group names. Consult your ISE configuration for the valid tags and names.
- **group-object** *object_group_name*—The name of an existing security group object group.

Example:

```
hostname(config-security-object-group)# security-group tag 1
hostname(config-security-object-group)# security-group name mgkt
hostname(config-security-object-group)# group-object local-sg
```

Step 3 (Optional) Add a description: **description** *string*

Configure Time Ranges

A time range object defines a specific time consisting of a start time, an end time, and optional recurring entries. You use these objects on ACL rules to provide time-based access to certain features or assets. For example, you could create an access rule that allows access to a particular server during working hours only.

Note You can include multiple periodic entries in a time range object. If a time range has both absolute and periodic values specified, then the periodic values are evaluated only after the absolute start time is reached, and they are not further evaluated after the absolute end time is reached.

Creating a time range does not restrict access to the device. This procedure defines the time range only. You must then use the object in an access control rule.

Procedure

Step 1 Create the time range: **time-range** *name*

Step 2 (Optional.) Add a start or end time (or both) to the time range.

absolute [**start** *time date*] [**end** *time date*]

If you do not specify a start time, the default start time is now.

The *time* is in the 24-hour format *hh*:*mm*. For example, 8:00 is 8:00 a.m. and 20:00 is 8:00 p.m.

The *date* is in the format *day month year*; for example, **1 January 2014**.

Step 3 (Optional.) Add recurring time periods.

periodic *days-of-the-week time* **to** [*days-of-the-week*] *time*

You can specify the following values for *days-of-the-week*. Note that you can specify a second day of the week only if you specify a single day for the first argument.

- **Monday**, **Tuesday**, **Wednesday**, **Thursday**, **Friday**, **Saturday**, or **Sunday**. You can specify more than one of these, separated by spaces, for the first *days-of-the-week* argument.
- **daily**
- **weekdays**
- **weekend**

The *time* is in the 24-hour format *hh*:*mm*. For example, 8:00 is 8:00 a.m. and 20:00 is 8:00 p.m.

You can repeat this command to configure more than one recurring period.

Examples

The following is an example of an absolute time range beginning at 8:00 a.m. on January 1, 2006. Because no end time and date are specified, the time range is in effect indefinitely.

```
hostname(config)# time-range for2006
hostname(config-time-range)# absolute start 8:00 1 january 2006
```

The following is an example of a weekly periodic time range from 8:00 a.m. to 6:00 p.m on weekdays:

```
hostname(config)# time-range workinghours
hostname(config-time-range)# periodic weekdays 8:00 to 18:00
```

The following example establishes an end date for the time range, and sets a weekday period from 8 a.m. to 5 p.m., plus different hours after 5 for Monday, Wednesday, Friday compared to Tuesday, Thursday.

```
asa4(config)# time-range contract-A-access
```

```
asa4(config-time-range)# absolute end 12:00 1 September 2025
asa4(config-time-range)# periodic weekdays 08:00 to 17:00
asa4(config-time-range)# periodic Monday Wednesday Friday 18:00 to 20:00
asa4(config-time-range)# periodic Tuesday Thursday 17:30 to 18:30
```

Monitoring Objects

To monitor objects and groups, enter the following commands:

- **show access-list**

  Displays the access list entries. Entries that include objects are also expanded out into individual entries based on the object contents.

- **show running-config object** [**id** *object_id*]

  Displays all current objects. Use the **id** keyword to view a single object by name.

- **show running-config object** *object_type*

  Displays the current objects by their type, **network** or **service**.

- **show running-config object-group** [**id** *group_id*]

  Displays all current object groups. Use the **id** keyword to view a single object group by name.

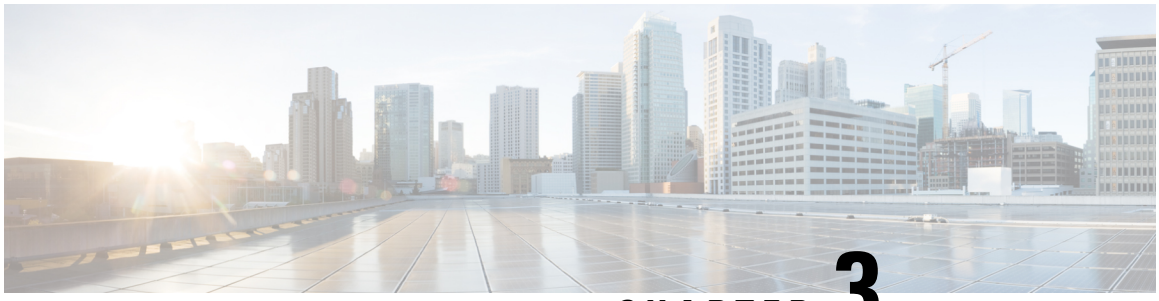
- **show running-config object-group** *grp_type*

  Displays the current object groups by their group type.

History for Objects

Feature Name	Platform Releases	Description
Object groups	7.0(1)	Object groups simplify ACL creation and maintenance. We introduced or modified the following commands: **object-group** *protocol*, **object-group** *network*, **object-group** *service*, **object-group** *icmp_type*.
Regular expressions and policy maps	7.2(1)	Regular expressions and policy maps were introduced to be used under inspection policy maps. The following commands were introduced: **class-map type regex**, **regex**, **match regex**.
Objects	8.3(1)	Object support was introduced. We introduced or modified the following commands: **object-network**, **object-service**, **object-group** network, **object-group** service, **network object**, **access-list extended**, **access-list webtype**, **access-list remark**.

Feature Name	Platform Releases	Description
User Object Groups for Identity Firewall	8.4(2)	User object groups for identity firewall were introduced. We introduced the following commands: **object-network user**, **user**.
Security Group Object Groups for Cisco TrustSec	8.4(2)	Security group object groups for Cisco TrustSec were introduced. We introduced the following commands: **object-network security**, **security**.
Mixed IPv4 and IPv6 network object groups	9.0(1)	Previously, network object groups could only contain all IPv4 addresses or all IPv6 addresses. Now network object groups can support a mix of both IPv4 and IPv6 addresses. **Note** You cannot use a mixed object group for NAT. We modified the following commands: **object-group network**.
Extended ACL and object enhancement to filter ICMP traffic by ICMP code	9.0(1)	ICMP traffic can now be permitted/denied based on ICMP code. We introduced or modified the following commands: **access-list extended**, **service-object**, **service**.
Service object support for Stream Control Transmission Protocol (SCTP)	9.5(2)	You can now create service objects and groups that specific SCTP ports. We modified the following commands: **service-object**, **service**.
Network-service objects and their use in policy-based routing and access control.	9.17(1)	You can configure network-service objects and use them in extended access control lists for use in policy-based routing route maps and access control groups. Network-service objects include IP subnet or DNS domain name specifications, and optionally protocol and port specifications, that essentially combine network and service objects. This feature also includes the ability to define trusted DNS servers, to ensure that any DNS domain name resolutions acquire IP addresses from trusted sources. We added or modified the following commands: **access-list extended**, **app-id**, **clear configure object network-service**, **clear configure object-group network-service**, **clear dns ip-cache**, **clear object**, **clear object-group**, **debug network-service**, **description**, **dns trusted-source**, **domain**, **network-service-member**, **network-service reload**, **object-group network-service**, **object network-service**, **policy-route cost**, **set adaptive-interface cost**, **show asp table classify**, **show asp table network-service**, **show dns trusted-source**, **show dns ip-cache**, **show object**, **show object-group**, **show running-config**, **subnet**.
Network-service groups support	9.19(1)	You can now define a maximum of 1024 network service groups.



CHAPTER 3

Access Control Lists

Access control lists (ACLs) are used by many different features. When applied to interfaces or globally as access rules, they permit or deny traffic that flows through the appliance. For other features, the ACL selects the traffic to which the feature will apply, performing a matching service rather than a control service.

The following sections explain the basics of ACLs and how to configure and monitor them. Access rules, ACLs applied globally or to interfaces, are explained in more detail in Access Rules, on page 51.

- About ACLs, on page 27
- Licensing for Access Control Lists, on page 31
- Guidelines for ACLs, on page 31
- Configure ACLs, on page 32
- Edit ACLs in an Isolated Configuration Session, on page 46
- Monitoring ACLs, on page 48
- History for ACLs, on page 48

About ACLs

Access control lists (ACLs) identify traffic flows by one or more characteristics, including source and destination IP address, IP protocol, ports, EtherType, and other parameters, depending on the type of ACL. ACLs are used in a variety of features. ACLs are made up of one or more access control entries (ACEs).

ACL Types

The ASA uses the following types of ACLs:

- Extended ACLs—Extended ACLs are the main type that you will use. These ACLs are used for access rules to permit and deny traffic through the device, and for traffic matching by many features, including service policies, AAA rules, WCCP, Botnet Traffic Filter, and VPN group and DAP policies. See Configure Extended ACLs, on page 33.
- EtherType ACLs—EtherType ACLs apply to non-IP layer-2 traffic on bridge group member interfaces only. You can use these rules to permit or drop traffic based on the EtherType value in the layer-2 packet. With EtherType ACLs, you can control the flow of non-IP traffic across the device. See Configure EtherType ACLs, on page 45.
- Webtype ACLs—Webtype ACLs are used for filtering clientless SSL VPN traffic. These ACLs can deny access based on URLs or destination addresses. See Configure Webtype ACLs, on page 41.

- Standard ACLs—Standard ACLs identify traffic by destination address only. There are few features that use them: route maps and VPN filters. Because VPN filters also allow extended access lists, limit standard ACL use to route maps. See Configure Standard ACLs, on page 41.

The following table lists some common uses for ACLs and the type to use.

Table 1: ACL Types and Common Uses

ACL Use	ACL Type	Description
Control network access for IP traffic (routed and transparent mode)	Extended	The ASA does not allow any traffic from a lower security interface to a higher security interface unless it is explicitly permitted by an extended ACL. In routed mode, you must use an ACL to permit traffic between a bridge group member interface and an interface outside same the bridge group. **Note** To access the ASA interface for management access, you do not also need an ACL allowing the host IP address. You only need to configure management access according to the general operations configuration guide.
Identify traffic for AAA rules	Extended	AAA rules use ACLs to identify traffic.
Augment network access control for IP traffic for a given user	Extended, downloaded from a AAA server per user	You can configure the RADIUS server to download a dynamic ACL to be applied to the user, or the server can send the name of an ACL that you already configured on the ASA.
VPN access and filtering	Extended Standard	Group policies for remote access and site to site VPNs use standard or extended ACLs for filtering. Remote access VPNs also use extended ACLs for client firewall configurations and dynamic access policies.
Identify traffic in a traffic class map for Modular Policy Framework	Extended	ACLs can be used to identify traffic in a class map, which is used for features that support Modular Policy Framework. Features that support Modular Policy Framework include TCP and general connection settings, and inspection.
For bridge group member interfaces, control network access for non-IP traffic	EtherType	You can configure an ACL that controls traffic based on its EtherType for any interface that is a member of a bridge group.
Identify route filtering and redistribution	Standard Extended	Various routing protocols use standard ACLs for route filtering and redistribution (through route maps) for IPv4 addresses, and extended ACLs for IPv6.
Filtering for clientless SSL VPN	Webtype	You can configure a webtype ACL to filter URLs and destinations.

ACL Names

Each ACL has a name or numeric ID, such as outside_in, OUTSIDE_IN, or 101. Limit the names to 241 characters or fewer.Consider using all uppercase letters to make it easier to find the name when viewing a running configuration.

Develop a naming convention that will help you identify the intended purpose of the ACL. For example, ASDM uses the convention *interface-name_purpose_direction*, such as “outside_access_in”, for an ACL applied to the “outside” interface in the inbound direction.

Traditionally, ACL IDs were numbers. Standard ACLs were in the range 1-99 or 1300-1999. Extended ACLs were in the range 100-199 or 2000-2699. The ASA does not enforce these ranges, but if you want to use numbers, you might want to stick to these conventions to maintain consistency with routers running IOS Software.

Access Control Entry Order

An ACL is made up of one or more ACEs. Unless you explicitly insert an ACE at a given line, each ACE that you enter for a given ACL name is appended to the end of the ACL.

The order of ACEs is important. When the ASA decides whether to forward or drop a packet, the ASA tests the packet against each ACE in the order in which the entries are listed. After a match is found, no more ACEs are checked.

Thus, if you place a more specific rule after a more general rule, the more specific rule might never be hit. For example, if you want to permit network 10.1.1.0/24, but drop traffic from host 10.1.1.15 on that subnet, the ACE that denies 10.1.1.15 must come before the one that permits 10.1.1.0/24. If the permit 10.1.1.0/24 ACE comes first, 10.1.1.15 will be allowed, and the deny ACE will never be matched.

In an extended ACL, use the **line** *number* parameter on the **access-list** command to insert rules at the right location. Use the **show access-list** *name* command to view the ACL entries and their line numbers to help determine the right number to use. For other types of ACL, you must rebuild the ACL (or better, use ASDM) to change the order of ACEs.

Permit/Deny vs. Match/Do Not Match

Access control entries either “permit” or “deny” traffic that matches the rule. When you apply an ACL to a feature that determines whether traffic is allowed through the ASA or is dropped, such as global and interface access rules, “permit” and “deny” mean what they say.

For other features, such as service policy rules, “permit” and “deny” actually mean “match” or “do not match.” In these cases, the ACL is selecting the traffic that should receive the services of that feature, such as application inspection or redirection to a service module. “Denied” traffic is simply traffic that does not match the ACL, and thus will not receive the service.

Access Control Implicit Deny

ACLs that are used for through-the-box access rules have an implicit deny statement at the end. Thus, for traffic controlling ACLs such as those applied to interfaces, if you do not explicitly permit a type of traffic, that traffic is dropped. For example, if you want to allow all users to access a network through the ASA except for one or more particular addresses, then you need to deny those particular addresses and then permit all others.

For management (control plane) ACLs, which control to-the-box traffic, there is no implicit deny at the end of a set of management rules for an interface. Instead, any connection that does not match a management access rule is then evaluated by regular access control rules.

For ACLs used to select traffic for a service, you must explicitly "permit" the traffic; any traffic not "permitted" will not be matched for the service; "denied" traffic bypasses the service.

For EtherType ACLs, the implicit deny at the end of the ACL does not affect IP traffic or ARPs; for example, if you allow EtherType 8037, the implicit deny at the end of the ACL does not now block any IP traffic that you previously allowed with an extended ACL (or implicitly allowed from a high security interface to a low security interface). However, if you *explicitly* deny all traffic with an EtherType ACE, then IP and ARP traffic is denied; only physical protocol traffic, such as auto-negotiation, is still allowed.

IP Addresses Used for Extended ACLs When You Use NAT

When you use NAT or PAT, you are translating addresses or ports, typically mapping between internal and external addresses. If you need to create an extended ACL that applies to addresses or ports that have been translated, you need to determine whether to use the real (untranslated) addresses or ports or the mapped ones. The requirement differs by feature.

Using the real address and port means that if the NAT configuration changes, you do not need to change the ACLs.

Features That Use Real IP Addresses

The following commands and features use real IP addresses in the ACLs, even if the address as seen on an interface is the mapped address:

- Access Rules (extended ACLs referenced by the **access-group** command)
- Service Policy Rules (Modular Policy Framework **match access-list** command)
- Botnet Traffic Filter traffic classification (**dynamic-filter enable classify-list** command)
- AAA Rules (**aaa ... match** commands)
- WCCP (**wccp redirect-list group-list** command)

For example, if you configure NAT for an inside server, 10.1.1.5, so that it has a publicly routable IP address on the outside, 209.165.201.5, then the access rule to allow the outside traffic to access the inside server needs to reference the server's real IP address (10.1.1.5), and not the mapped address (209.165.201.5).

```
hostname(config)# object network server1
hostname(config-network-object)# host 10.1.1.5
hostname(config-network-object)# nat (inside,outside) static 209.165.201.5

hostname(config)# access-list OUTSIDE extended permit tcp any host 10.1.1.5 eq www
hostname(config)# access-group OUTSIDE in interface outside
```

Features That Use Mapped IP Addresses

The following features use ACLs, but these ACLs use the mapped values as seen on an interface:

- IPsec ACLs
- **capture** command ACLs

- Per-user ACLs
- Routing protocol ACLs
- All other feature ACLs.

Time-Based ACEs

You can apply time range objects to extended and webtype ACEs so that the rules are active for specific time periods only. These types of rules let you differentiate between activity that is acceptable at certain times of the day but that is unacceptable at other times. For example, you could provide additional restrictions during working hours, and relax them after work hours or at lunch. Conversely, you could essentially shut your network down during non-work hours.

You cannot create time-based rules that have the exact same protocol, source, destination, and service criteria of a rule that does not include a time range object. The non-time-based rule always overrides the duplicate time-based rule, as they are redundant.



Note Users could experience a delay of approximately 80 to 100 seconds after the specified end time for the ACL to become inactive. For example, if the specified end time is 3:50, because the end time is inclusive, the command is picked up anywhere between 3:51:00 and 3:51:59. After the command is picked up, the ASA finishes any currently running task and then services the command to deactivate the ACL.

Licensing for Access Control Lists

Access control lists do not require a special license.

However, to use **sctp** as the protocol in an entry, you must have a Carrier license.

Guidelines for ACLs

Firewall Mode

- Extended and standard ACLs are supported in routed and transparent firewall modes.
- Webtype ACLs are supported in routed mode only.
- EtherType ACLs are supported for bridge group member interfaces only, in routed and transparent modes.

Failover and Clustering

Configuration sessions are not synchronized across failover or clustered units. When you commit the changes in a session, they are made in all failover and cluster units as normal.

IPv6

- Extended and webtype ACLs allow a mix of IPv4 and IPv6 addresses.

- Standard ACLs do not allow IPv6 addresses.
- EtherType ACLs do not contain IP addresses.

Additional Guidelines

- When you specify a network mask, the method is different from the Cisco IOS software **access-list** command. The ASA uses a network mask (for example, 255.255.255.0 for a Class C mask). The Cisco IOS mask uses wildcard bits (for example, 0.0.0.255).
- (Extended ACL only) The following features use ACLs, but cannot accept an ACL with identity firewall (specifying user or group names), FQDN (fully-qualified domain names), or Cisco TrustSec values:
  - VPN **crypto map** command
  - VPN **group-policy** command, except for **vpn-filter**
  - WCCP
  - DAP

Configure ACLs

The following sections explain how to configure the various types of ACL, Read the section on ACL basics to get the big picture, then the sections on specific types of ACL for the details.

Basic ACL Configuration and Management Options

An ACL is made up of one or more access control entries (ACEs) with the same ACL ID or name. To create a new ACL, you simply create an ACE with a new ACL name, and it becomes the first rule in the new ACL.

Working with an ACL, you can do the following things:

Examine the ACL contents and determine line numbers and hit counts

Use the **show access-list** *name* command to view the contents of the ACL. Each row is an ACE, and includes the line number, which you will need to know if you want to insert new entries into an extended ACL. The information also includes a hit count for each ACE, which is how many times the rule was matched by traffic. For example:

```
hostname# show access-list outside_access_in
access-list outside_access_in; 3 elements; name hash: 0x6892a938
access-list outside_access_in line 1 extended permit ip 10.2.2.0 255.255.255.0 any
(hitcnt=0) 0xcc48b55c
access-list outside_access_in line 2 extended permit ip host
2001:DB8::0DB8:800:200C:417A any (hitcnt=0) 0x79797f94
access-list outside_access_in line 3 extended permit ip user-group
LOCAL\\usergroup any any (hitcnt=0) 0xb0f5b1e1
```

Add an ACE

The command for adding an ACE is **access-list** *name* [**line** *line-num*] *type parameters*. The line number argument works for extended ACLs only. If you include the line number, the ACE is inserted at that

location in the ACL, and the ACE that was at that location is moved down, along with the remainder of the ACEs (that is, inserting an ACE at a line number does not replace the old ACE at that line). If you do not include a line number, the ACE is added to the end of the ACL. The parameters available differ based on the ACL type; see the specific topics on each ACL type for details.

Add comments to an ACL (all types except webtype)

Use the **access-list** *name* [**line** *line-num*] **remark** *text* command to add remarks into an ACL to help explain the purpose of an ACE. Best practice is to insert the remark before the ACE; if you view the configuration in ASDM, remarks will be associated with the ACE that follows the remarks. You can enter multiple remarks before an ACE to include an expanded comment. Each remark is limited to 100 characters. You can include leading spaces to help set off the remarks. If you do not include a line number, the remark is added to the end of the ACL. For example, you could add remarks before adding each ACE:

```
hostname(config)# access-list OUT remark - this is the inside admin address
hostname(config)# access-list OUT extended permit ip host 209.168.200.3 any
hostname(config)# access-list OUT remark - this is the hr admin address
hostname(config)# access-list OUT extended permit ip host 209.168.200.4 any
```

Edit or move an ACE or remark

You cannot edit or move an ACE or remark. Instead, you must create a new ACE or remark with the desired values at the right location (using the line number), then delete the old ACE or remark. Because you can insert ACEs in extended ACLs only, you need to rebuild standard, webtype, or EtherType ACLs if you need to edit or move ACEs. It is far easier to reorganize a long ACL using ASDM.

Delete an ACE or remark

Use the **no access-list** *parameters* command to remove an ACE or remark. Use the **show access-list** command to view the parameter string that you must enter: the string must exactly match an ACE or remark to delete it, with the exception of the **line** *line-num* argument, which is optional on the **no access-list** command.

Delete an entire ACL, including remarks

Use the **clear configure access-list** *name* command. USE CAUTION! The command does not ask you for confirmation. If you do not include a name, every access list on the ASA is removed.

Rename an ACL

Use the **access-list** *name* **rename** *new_name* command.

Apply the ACL to a policy

Creating an ACL in and of itself does nothing to traffic. You must apply the ACL to a policy. For example, you can use the **access-group** command to apply an extended ACL to an interface, thus denying or permitting traffic that goes through the interface.

Configure Extended ACLs

An extended ACL is composed of all ACEs with the same ACL ID or name. Extended ACLs are the most complex and feature-rich type of ACL, and you can use them for many features. The most noteworthy use of extended ACLs is as access groups applied globally or to interfaces, which determine the traffic that will be denied or permitted to flow through the box. But extended ACLs are also used to determine the traffic to which other services will be provided.

Because extended ACLs are complex, the following sections focus on creating ACEs to provide specific types of traffic matching. The first sections, on basic address-based ACEs and on TCP/UDP ACEs, build the foundation for the remaining sections.

Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching

The basic extended ACE matches traffic based on source and destination addresses, including IPv4 and IPv6 addresses and fully-qualified domain names (FQDN), such as www.example.com. In fact, every type of extended ACE must include some specification for source and destination address, so this topic explains the minimum extended ACE.



Tip Tip If you want to match traffic based on FQDN, you must create a network object for each FQDN.

To add an ACE for IP address or FQDN matching, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} *protocol_argument source_address_argument dest_address_argument* [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time-range** *time_range_name*] [**inactive**]

Example:

```
hostname(config)# access-list ACL_IN extended permit ip any any
hostname(config)# access-list ACL_IN extended permit object service-obj-http any any
```

The options are:

- *access_list_name*—The name of the new or existing ACL.
- Line number—The **line** *line_number* option specifies the line number at which to insert the ACE; otherwise, the ACE is added to the end of the ACL.
- Permit or Deny—The **deny** keyword denies or exempts a packet if the conditions are matched. The **permit** keyword permits or includes a packet if the conditions are matched.
- Protocol—The *protocol_argument* specifies the IP protocol. If you use network-service object groups that specify the protocol and ports, you should specify **ip** in this argument.
  - *name* or *number*—Specifies the protocol name or number. Specify **ip** to apply to all protocols.
  - **object-group** *protocol_grp_id*—Specifies a protocol object group created using the **object-group protocol** command.
  - **object** *service_obj_id*—Specifies a service object created using the **object service** command. The object can include port or ICMP type and code specifications if desired.
  - **object-group** *service_grp_id*—Specifies a service object group created using the **object-group service** command.
- Source Address, Destination Address—The *source_address_argument* specifies the IP address or FQDN from which the packet is being sent, and the *dest_address_argument* specifies the IP address or FQDN to which the packet is being sent:
  - **host** *ip_address*—Specifies an IPv4 host address.

- *ip_address mask*—Specifies an IPv4 network address and subnet mask, such as 10.100.10.0 255.255.255.0.
- *ipv6-address/prefix-length*—Specifies an IPv6 host or network address and prefix.
- **any**, **any4**, and **any6**—**any** specifies both IPv4 and IPv6 traffic; **any4** specifies IPv4 traffic only; and **any6** specifies IPv6 traffic only.
- **interface** *interface_name*—Specifies the name of an ASA interface. Use the interface name rather than IP address to match traffic based on which interface is the source or destination of the traffic.
- **object** *nw_obj_id*—Specifies a network object created using the **object network** command.
- **object-group** *nw_grp_id*—Specifies a network object group created using the **object-group network** command.
- **object-group-network-service** *name*—Specifies the name of a network-service object group.

- Logging—**log** arguments set logging options when an ACE matches a connection for network access (an ACL applied with the **access-group** command). If you enter the **log** option without any arguments, you enable syslog message 106100 at the default level (6) and for the default interval (300 seconds). Log options are:
  - *level*—A severity level between 0 and 7. The default is 6 (informational). If you change this level for an active ACE, the new level applies to new connections; existing connections continue to be logged at the previous level.
  - **interval** *secs*—The time interval in seconds between syslog messages, from 1 to 600. The default is 300. This value is also used as the timeout value for deleting an inactive flow from the cache used to collect drop statistics.
  - **disable**—Disables all ACE logging.
  - **default**—Enables logging to message 106023 for denied packets. This setting is the same as not including the **log** option.
- Time Range—The **time-range** *time_range_name* option specifies a time range object, which determines the times of day and days of the week in which the ACE is active. If you do not include a time range, the ACE is always active.
- Activation—Use the **inactive** option to disable the ACE without deleting it. To reenable it, enter the entire ACE without the inactive keyword.

Add an Extended ACE for Port-Based Matching

If you specify service objects in an ACE, the service objects can include protocols with port specifications, such as TCP/80. Alternatively, you can specify the ports directly in the ACE. With port-based matching, you can target certain types of traffic for port-based protocols rather than all traffic for the protocol.



Note If you use network-service object groups that specify the protocol and ports, you should not specify ports as described in this topic. Specify **ip** as the protocol so that the protocol/port defined in the object can be matched.

The port-based extended ACE is just the basic address-matching ACE where the protocol is **tcp**, **udp**, or **sctp**. To add port specifications, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} {**tcp** | **udp** | **sctp**} *source_address_argument* [*port_argument*] *dest_address_argument* [*port_argument*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**] [**time-range** *time-range-name*] [**inactive**]

Example:

```
hostname(config)# access-list ACL_IN extended deny tcp any host 209.165.201.29 eq www
```

The *port_argument* option specifies the source or destination port. If you do not specify ports, all ports are matched. Available arguments include:

- *operator port*—The *port* can be the integer or name of a port. The *operator* can be one of the following:
  - **lt**—less than
  - **gt**—greater than
  - **eq**—equal to
  - **neq**—not equal to
  - **range**—an inclusive range of values. When you use this operator, specify two port numbers, for example, **range 100 200**.

Note DNS, Discard, Echo, Ident, NTP, RPC, SUNRPC, and Talk each require one definition for TCP and one for UDP. TACACS+ requires one definition for port 49 on TCP.

- **object-group** *service_grp_id*—Specifies a service object group created using the **object-group service** {**tcp** | **udp** | **tcp-udp**} command. Note that these object types are no longer recommended.

  You cannot specify the recommended generic service objects, where the protocol and port are defined within the object, as the port argument. You specify these objects as part of the protocol argument, as explained in Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 34.

For an explanation of the other keywords, and how to use service objects to specify protocols and ports, see Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 34.

Add an Extended ACE for ICMP-Based Matching

To match ICMP traffic, use one of the ICMP protocol keywords: **icmp** or **icmp6**. The **icmp** protocol matches IPv4 addresses only, whereas the **icmp6** protocol matches IPv6 addresses only. Ensure that your network addresses match the protocol you select, or the ACE will match no connections.

If you specify service objects in an ACE, the service objects can include the ICMP/ICMP6 protocols ICMP type and code specifications. Alternatively, you can specify the ICMP type and code directly in the ACE. For example, you can target ICMP Echo Request traffic (pings).

The ICMP extended ACE is just the basic address-matching ACE where the protocol is **icmp** or **icmp6**. Because these protocols have type and code values, you can add type and code specifications to the ACE.

To add an ACE for IP address or FQDN matching, where the protocol is ICMP or ICMP6, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} {**icmp** | **icmp6**} *source_address_argument dest_address_argument* [*icmp_argument*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time-range** *time_range_name*] [**inactive**]

Example:

```
hostname(config)# access-list abc extended permit icmp any any object-group obj_icmp_1
hostname(config)# access-list abc extended permit icmp any any echo
```

The *icmp_argument* option specifies the ICMP type and code.

- *icmp_type* [*icmp_code*]—Specifies the ICMP type by name or number, and the optional ICMP code for that type. If you do not specify the code, then all codes are used.
- **object-group** *icmp_grp_id*—Specifies an object group for ICMP/ICMP6 created using the (deprecated) **object-group icmp-type** command.

  You cannot specify the recommended generic service objects, where the protocol and type are defined within the object, as the ICMP argument. You specify these objects as part of the protocol argument, as explained in Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 34.

For an explanation of the other keywords, see Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 34.

Add an Extended ACE for User-Based Matching (Identity Firewall)

The user-based extended ACE is just the basic address-matching ACE where you include username or user group to the source matching criteria. By creating rules based on user identity, you can avoid tying rules to static host or network addresses. For example, if you define a rule for user1, and the identity firewall feature maps that user to a host assigned 10.100.10.3 one day, but 192.168.1.5 the next day, the user-based rule still applies.

Because you must still supply source and destination addresses, broaden the source address to include the likely addresses that will be assigned to the user (normally through DHCP). For example, user “LOCAL\user1 any” will match the LOCAL\user1 user no matter what address is assigned, whereas “LOCAL\user1 10.100.1.0 255.255.255.0” matches the user only if the address is on the 10.100.1.0/24 network.

By using group names, you can define rules based on entire classes of users, such as students, teachers, managers, engineers, and so forth.

To add an ACE for user or group matching, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} *protocol_argument* [*user_argument*] *source_address_argument* [*port_argument*] *dest_address_argument* [*port_argument*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time-range** *time_range_name*] [**inactive**]

Example:

```
hostname(config)# access-list v1 extended permit ip user LOCAL\idfw
```

```
any 10.0.0.0 255.255.255.0
```

The *user_argument* option specifies the user or group for which to match traffic in addition to the source address. Available arguments include the following:

- **object-group-user** *user_obj_grp_id*—Specifies a user object group created using the **object-group user** command.
- **user** {[*domain_nickname*\\]*name* | **any** | **none**}—Specifies a username. Specify **any** to match all users with user credentials, or **none** to match addresses that are not mapped to usernames. These options are especially useful for combining **access-group** and **aaa authentication match** policies.
- **user-group** [*domain_nickname*\\]*user_group_name*—Specifies a user group name. Note the double \\ separating the domain and group name.

For an explanation of the other keywords, see Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 34.

Tip You can include both user and Cisco Trustsec security groups in a given ACE.

Add an Extended ACE for Security Group-Based Matching (Cisco TrustSec)

The security group (Cisco TrustSec) extended ACE is just the basic address-matching ACE where you include security groups or tags to the source or destination matching criteria. By creating rules based on security groups, you can avoid tying rules to static host or network addresses. Because you must still supply source and destination addresses, broaden the addresses to include the likely addresses that will be assigned to users (normally through DHCP).

Tip Before adding this type of ACE, configure Cisco TrustSec.

To add an ACE for security group matching, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} *protocol_argument* [*security_group_argument*] *source_address_argument* [*port_argument*] [*security_group_argument*] *dest_address_argument* [*port_argument*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**inactive** | **time-range** *time_range_name*]

Example:

```
hostname(config)# access-list INSIDE_IN extended permit ip
security-group name my-group any any
```

The *security_group_argument* option specifies the security group for which to match traffic in addition to the source or destination address. Available arguments include the following:

- **object-group-security** *security_obj_grp_id*—Specifies a security object group created using the **object-group security** command.

- **security-group** {**name** *security_grp_id* | **tag** *security_grp_tag*}—Specifies a security group name or tag.

For an explanation of the other keywords, see Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 34.

Tip You can include both user and Cisco Trustsec security groups in a given ACE.

Examples for Extended ACLs

The following ACL allows all hosts (on the interface to which you apply the ACL) to go through the ASA:

```
hostname(config)# access-list ACL_IN extended permit ip any any
```

The following ACL prevents hosts on 192.168.1.0/24 from accessing the 209.165.201.0/27 network for TCP-based traffic. All other addresses are permitted.

```
hostname(config)# access-list ACL_IN extended deny tcp 192.168.1.0 255.255.255.0
209.165.201.0 255.255.255.224
hostname(config)# access-list ACL_IN extended permit ip any any
```

If you want to restrict access to selected hosts only, then enter a limited permit ACE. By default, all other traffic is denied unless explicitly permitted.

```
hostname(config)# access-list ACL_IN extended permit ip 192.168.1.0 255.255.255.0
209.165.201.0 255.255.255.224
```

The following ACL restricts all hosts (on the interface to which you apply the ACL) from accessing a website at address 209.165.201.29. All other traffic is allowed.

```
hostname(config)# access-list ACL_IN extended deny tcp any host 209.165.201.29 eq www
hostname(config)# access-list ACL_IN extended permit ip any any
```

The following ACL that uses object groups restricts several hosts on the inside network from accessing several web servers. All other traffic is allowed.

```
hostname(config-network)# access-list ACL_IN extended deny tcp object-group denied
object-group web eq www
hostname(config)# access-list ACL_IN extended permit ip any any
hostname(config)# access-group ACL_IN in interface inside
```

The following example temporarily disables an ACL that permits traffic from one group of network objects (A) to another group of network objects (B):

```
hostname(config)# access-list 104 permit ip host object-group A object-group B inactive
```

To implement a time-based ACE, use the **time-range** command to define specific times of the day and week. Then use the **access-list extended** command to bind the time range to an ACE. The following example binds an ACE in the "Sales" ACL to a time range named "New_York_Minute."

```
hostname(config)# access-list Sales line 1 extended deny tcp host 209.165.200.225 host
209.165.201.1 time-range New_York_Minute
```

The following example shows a mixed IPv4/IPv6 ACL:

```
hostname(config)# access-list demoacl extended permit ip 2001:DB8:1::/64 10.2.2.0
255.255.255.0
hostname(config)# access-list demoacl extended permit ip 2001:DB8:1::/64 2001:DB8:2::/64
hostname(config)# access-list demoacl extended permit ip host 10.3.3.3 host 10.4.4.4
```

Example of Converting Addresses to Objects for Extended ACLs

The following normal ACL that does not use object groups restricts several hosts on the inside network from accessing several web servers. All other traffic is allowed.

```
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.4 host 209.165.201.29
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.78 host 209.165.201.29
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.89 host 209.165.201.29
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.4 host 209.165.201.16
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.78 host 209.165.201.16
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.89 host 209.165.201.16
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.4 host 209.165.201.78
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.78 host 209.165.201.78
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.89 host 209.165.201.78
eq www
hostname(config)# access-list ACL_IN extended permit ip any any
hostname(config)# access-group ACL_IN in interface inside
```

If you make two network object groups, one for the inside hosts, and one for the web servers, then the configuration can be simplified and can be easily modified to add more hosts:

```
hostname(config)# object-group network denied
hostname(config-network)# network-object host 10.1.1.4
hostname(config-network)# network-object host 10.1.1.78
hostname(config-network)# network-object host 10.1.1.89

hostname(config-network)# object-group network web
hostname(config-network)# network-object host 209.165.201.29
hostname(config-network)# network-object host 209.165.201.16
hostname(config-network)# network-object host 209.165.201.78

hostname(config)# access-list ACL_IN extended deny tcp object-group denied object-group
web eq www
hostname(config)# access-list ACL_IN extended permit ip any any
```

```
hostname(config)# access-group ACL_IN in interface inside
```

Configure Standard ACLs

A standard ACL is composed of all ACEs with the same ACL ID or name. Standard ACLs are used for a limited number of features, such as route maps or VPN filters. A standard ACL uses IPv4 addresses only, and defines destination addresses only.

To add a standard access list entry, use the following command:

access-list *access_list_name* **standard** {**deny** | **permit**} {**any4** | **host** *ip_address* | *ip_address mask*}

Example:

```
hostname(config)# access-list OSPF standard permit 192.168.1.0 255.255.255.0
```

The options are:

- Name—The *access_list_name* argument specifies the name of number of an ACL. Traditional numbers for standard ACLs are 1-99 or 1300-1999, but you can use any name or number. You create a new ACL if the ACL does not already exist, otherwise, you are adding the entry to the end of the ACL.
- Permit or Deny—The **deny** keyword denies or exempts a packet if the conditions are matched. The **permit** keyword permits or includes a packet if the conditions are matched.
- Destination Address—The **any4** keyword matches all IPv4 addresses. The **host** *ip_address* argument matches a host IPv4 address. The *ip_address ip_mask* argument matches an IPv4 subnet, for example, 10.1.1.0 255.255.255.0.

Configure Webtype ACLs

Webtype ACLs are used for filtering clientless SSL VPN traffic, constraining user access to specific networks, subnets, hosts, and Web servers. If you do not define a filter, all connections are allowed. A webtype ACL is composed of all ACEs with the same ACL ID or name.

With webtype ACLs, you can match traffic based on URLs or destination addresses. A single ACE cannot mix these specifications. The following sections explain each type of ACE.

Add a Webtype ACE for URL Matching

To match traffic based on the URL the user is trying to access, use the following command;

access-list *access_list_name* **webtype** {**deny** | **permit**} **url** {*url_string* | **any**} [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time_range** *time_range_name*] [**inactive**]

Example:

```
hostname(config)# access-list acl_company webtype deny url http://*.example.com
```

The options are:

- *access_list_name*—The name of the new or existing ACL. If the ACL already exists, you are adding the ACE to the end of the ACL.

- Permit or Deny—The **deny** keyword denies or exempts a packet if the conditions are matched. The **permit** keyword permits or includes a packet if the conditions are matched.
- URL—The **url** keyword specifies the URL to match. Use **url any** to match all URL-based traffic. Otherwise, enter a URL string, which can include wildcards. Following are some tips and limitations on specifying URLs:
  - Specify **any** to match all URLs.
  - 'Permit url any' will allow all the URLs that have the format protocol://server-ip/path and will block traffic that does not match this pattern, such as port-forwarding. There should be an ACE to allow connections to the required port (port 1494 in the case of Citrix) so that an implicit deny does not occur.
  - Smart tunnel and ica plug-ins are not affected by an ACL with 'permit url any' because they match smart-tunnel:// and ica:// types only.
  - You can use these protocols: cifs://, citrix://, citrixs://, ftp://, http://, https://, imap4://, nfs://, pop3://, smart-tunnel://, and smtp://. You can also use wildcards in the protocol; for example, htt* matches http and https, and an asterisk * matches all protocols. For example, *://*.example.com matches any type URL-based traffic to the example.com network.
  - If you specify a smart-tunnel:// URL, you can include the server name only. The URL cannot contain a path. For example, smart-tunnel://www.example.com is acceptable, but smart-tunnel://www.example.com/index.html is not.
  - An asterisk * matches none or any number of characters. To match any http URL, enter http://*/*.
  - A question mark ? matches any one character exactly.
  - Square brackets [] are range operators, matching any character in the range. For example, to match both http://www.cisco.com:80/ and http://www.cisco.com:81/, enter **http://www.cisco.com:8[01]/**.
- Logging—**log** arguments set logging options when an ACE matches a packet. If you enter the **log** option without any arguments, you enable syslog message 106102 at the default level (6) and for the default interval (300 seconds). Log options are:
  - *level*—A severity level between 0 and 7. The default is 6.
  - **interval** *secs*—The time interval in seconds between syslog messages, from 1 to 600. The default is 300.
  - **disable**—Disables all ACL logging.
  - **default**—Enables logging to message 106103. This setting is the same as not including the **log** option.
- Time Range—The **time-range** *time_range_name* option specifies a time range object, which determines the times of day and days of the week in which the ACE is active. If you do not include a time range, the ACE is always active.
- Activation—Use the **inactive** option to disable the ACE without deleting it. To reenable it, enter the entire ACE without the inactive keyword.

Add a Webtype ACE for IP Address Matching

You can match traffic based on the destination address the user is trying to access. The webtype ACL can include a mix of IPv4 and IPv6 addresses in addition to URL specifications.

To add a webtype ACE for IP address matching, use the following command:

access-list *access_list_name* **webtype** {**deny** | **permit**} **tcp** *dest_address_argument* [*operator port*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time_range** *time_range_name*]] [**inactive**]]

Example:

```
hostname(config)# access-list acl_company webtype permit tcp any
```

For an explanation of keywords not explained here, see Add a Webtype ACE for URL Matching, on page 41. Keywords and arguments specific to this type of ACE include the following:

- **tcp**—The TCP protocol. Webtype ACLs match TCP traffic only.
- Destination Address—The *dest_address_argument* specifies the IP address to which the packet is being sent:
  - **host** *ip_address*—Specifies an IPv4 host address.
  - *dest_ip_address mask*—Specifies an IPv4 network address and subnet mask, such as 10.100.10.0 255.255.255.0.
  - *ipv6-address/prefix-length*—Specifies an IPv6 host or network address and prefix.
  - **any**, **any4**, and **any6**—**any** specifies both IPv4 and IPv6 traffic; **any4** specifies IPv4 traffic only; and **any6** specifies IPv6 traffic only.
- *operator port*—The destination port. If you do not specify ports, all ports are matched. The *port* can be the integer or name of a TCP port. The *operator* can be one of the following:
  - **lt**—less than
  - **gt**—greater than
  - **eq**—equal to
  - **neq**—not equal to
  - **range**—an inclusive range of values. When you use this operator, specify two port numbers, for example, **range 100 200**.

Examples for Webtype ACLs

The following example shows how to deny access to a specific company URL:

```
hostname(config)# access-list acl_company webtype deny url http://*.example.com
```

The following example shows how to deny access to a specific web page:

```
hostname(config)# access-list acl_file webtype deny url https://www.example.com/dir/file.html
```

The following example shows how to deny HTTP access to any URL on a specific server through port 8080:

```
hostname(config)# access-list acl_company webtype deny url http://my-server:8080/*
```

The following examples show how to use wildcards in webtype ACLs.

- The following example matches URLs such as http://www.example.com/layouts/1033:

  ```
  access-list VPN-Group webtype permit url http://www.example.com/*
  ```

- The following example matches URLs such as http://www.example.com/ and http://www.example.net/:

  ```
  access-list test webtype permit url http://www.example.*
  ```

- The following example matches URLs such as http://www.example.com and ftp://wwz.example.com:

  ```
  access-list test webtype permit url *://ww?.e*co*/
  ```

- The following example matches URLs such as http://www.cisco.com:80 and https://www.cisco.com:81:

  ```
  access-list test webtype permit url *://ww?.c*co*:8[01]/
  ```

  The range operator "[]" in the preceding example specifies that either character **0** or **1** can occur at that location.

- The following example matches URLs such as http://www.example.com and http://www.example.net:

  ```
  access-list test webtype permit url http://www.[a-z]xample?*/
  ```

  The range operator "[]" in the preceding example specifies that any character in the range from **a** to **z** can occur.

- The following example matches http or https URLs that include "cgi" somewhere in the file name or path.

  ```
  access-list test webtype permit url htt*://*/*cgi?*
  ```

Note To match any http URL, you must enter **http://*/*** instead of http://*.

The following example shows how to enforce a webtype ACL to disable access to specific CIFS shares.

In this scenario we have a root folder named "shares" that contains two sub-folders named "Marketing_Reports" and "Sales_Reports." We want to specifically deny access to the "shares/Marketing_Reports" folder.

```
access-list CIFS_Avoid webtype deny url cifs://172.16.10.40/shares/Marketing_Reports.
```

However, due to the implicit “deny all” at the end of the ACL, the above ACL makes all of the sub-folders inaccessible (“shares/Sales_Reports” and “shares/Marketing_Reports”), including the root folder (“shares”).

To fix the problem, add a new ACL to allow access to the root folder and the remaining sub-folders:

```
access-list CIFS_Allow webtype permit url cifs://172.16.10.40/shares*
```

Configure EtherType ACLs

EtherType ACLs apply to non-IP layer-2 traffic on bridge group member interfaces. You can use these rules to permit or drop traffic based on the EtherType value in the layer-2 packet. With EtherType ACLs, you can control the flow of non-IP traffic across the bridge group. Note that 802.3-formatted frames are not handled by the ACL because they use a length field as opposed to a type field.

To add an EtherType ACE, use the following command:

access-list *access_list_name* **ethertype** {**deny** | **permit**} {**any** | **bpdu** | **dsap** {*hex_address* | **bpdu** | **ipx** | **isis** | **raw-ipx**} | **eii-ipx** | **ipx** | **isis** | **mpls-multicast** | **mpls-unicast** | *hex_number*}

Example:

```
hostname(config)# access-list ETHER ethertype deny mpls-multicast
```

The options are:

- *access_list_name*—The name of the new or existing ACL. If the ACL already exists, you are adding the ACE to the end of the ACL.
- Permit or Deny—The **deny** keyword denies a packet if the conditions are matched. The **permit** keyword permits a packet if the conditions are matched.
- Traffic Matching Criteria—You can match traffic using the following options:
  - **any**—Matches all layer 2 traffic.
  - **bpdu**—bridge protocol data units (dsap 0x42), which are allowed by default. This keyword is converted to **dsap bpdu**.
  - **dsap** {*hex_address* | **bpdu** | **ipx** | **isis** | **raw-ipx**}—The IEEE 802.2 Logical Link Control (LLC) packet's Destination Service Access Point address. Include the address you want to permit or deny in hexadecimal, from 0x01 to 0xff. You can also use the following keywords to create rules for common values:
    - **bpdu** for 0x42, bridge protocol data units.
    - **ipx** for 0xe0, Internet Packet Exchange (IPX) 802.2 LLC.
    - **isis** for 0xfe, Intermediate System to Intermediate System (IS-IS).
    - **raw-ipx** for 0xff, raw IPX 802.3 format.
  - **eii-ipx**—Ethernet II IPX format, EtherType 0x8137.
  - **ipx**—Internet Packet Exchange (IPX). This keyword is a shortcut for configuring three separate rules, for **dsap ipx**, **dsap raw-ipx**, and **eii-ipx**.

- **isis**—Intermediate System to Intermediate System (IS-IS). This keyword is converted to **dsap isis**.
- **mpls-multicast**— MPLS multicast.
- **mpls-unicast**—MPLS unicast.
- *hex_number*—Any EtherType that can be identified by a 16-bit hexadecimal number 0x600 to 0xffff. See RFC 1700, "Assigned Numbers," at http://www.ietf.org/rfc/rfc1700.txt for a list of EtherTypes.

Examples for EtherType ACLs

The following examples shows how to configure EtherType ACLs, including how to apply them to an interface.

For example, the following sample ACL allows common EtherTypes originating on the inside interface:

```
hostname(config)# access-list ETHER ethertype permit ipx
INFO: ethertype ipx is saved to config as ethertype eii-ipx
INFO: ethertype ipx is saved to config as ethertype dsap ipx
INFO: ethertype ipx is saved to config as ethertype dsap raw-ipx
hostname(config)# access-list ETHER ethertype permit mpls-unicast
hostname(config)# access-group ETHER in interface inside
```

The following example allows some EtherTypes through the ASA, but it denies all others:

```
hostname(config)# access-list ETHER ethertype permit 0x1234
hostname(config)# access-list ETHER ethertype permit mpls-unicast
hostname(config)# access-group ETHER in interface inside
hostname(config)# access-group ETHER in interface outside
```

The following example denies traffic with EtherType 0x1256 but allows all others on both interfaces:

```
hostname(config)# access-list nonIP ethertype deny 1256
hostname(config)# access-list nonIP ethertype permit any
hostname(config)# access-group nonIP in interface inside
hostname(config)# access-group nonIP in interface outside
```

Edit ACLs in an Isolated Configuration Session

When you edit an ACL used for access rules or any other purpose, the change is immediately implemented and impacts traffic. With access rules, you can enable the transactional commit model to ensure that new rules become active only after rule compilation is complete, but the compilation happens after each ACE you edit.

If you want to further isolate the impact of editing ACLs, you can make your changes in a "configuration session," which is an isolated mode that allows you to edit several ACEs and objects before explicitly committing your changes. Thus, you can ensure that all of your intended changes are complete before you change device behavior.

Before you begin

- You can edit ACLs that are referenced by an access-group command, but you cannot edit ACLs that are referenced by any other command. You can also edit unreferenced ACLs or create new ones.

- You can create or edit objects and object groups, but if you create one in a session, you cannot edit it in the same session. If the object is not defined as desired, you must commit your changes and then edit the object, or discard the entire session and start over.
- When you edit an ACL that is referenced by an access-group command (access rules), the transactional commit model is used when you commit the session. Thus, the ACL is completely compiled before the new ACL replaces the old version.

Procedure

Step 1 Start the session.

```
hostname#configure session session_name
hostname(config-s)#
```

If the *session_name* already exists, you open that session. Otherwise, you are creating a new session.

Use the **show configuration session** command to view the existing sessions. You can have at most 3 sessions active at a time. If you need to delete an old unused session, use the **clear configuration session** *session_name* command.

If you cannot open an existing session because someone else is editing it, you can clear the flag that indicates the session is being edited. Do this only if you are certain the session is not actually being edited. Use the **clear session** *session_name* **access** command to reset the flag.

Step 2 (Uncommitted sessions only.) Make your changes. You can use the following basic commands with any of their parameters:

- **access-list**
- **object**
- **object-group**

Step 3 Decide what to do with the session. The commands available depend on whether you have previously committed the session. Possible commands are:

- **exit**—To simply exit the session without committing or discarding changes, so that you can return later.
- **commit** [**noconfirm** [**revert-save** | **config-save**]]—(Uncommitted sessions only.) To commit your changes. You are asked if you want to save the session. You can save the revert session (**revert-save**), which lets you undo your changes using the **revert** command, or the configuration session (**config-save**), which includes all of the changes made in the session (allowing you to commit the same changes again if you would like to). If you save the revert or configuration session, the changes are committed, but the session remains active. You can open the session and revert or recommit the changes. You can avoid the prompt by including the **noconfirm** option and optionally, the desired save option.
- **abort**—(Uncommitted sessions only.) To abandon your changes and delete the session. If you want to keep the session, exit the session and use the **clear session** *session_name* **configuration** command, which empties the session without deleting it.
- **revert**—(Committed sessions only.) To undo your changes, returning the configuration back to what it was before you committed the session, and delete the session.

- **show configuration session** [*session_name*]—To show the changes made in the session.

Monitoring ACLs

To monitor ACLs, enter one of the following commands:

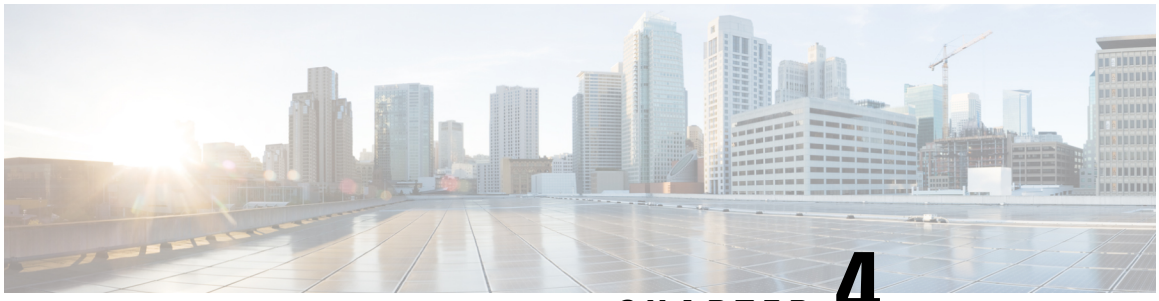
- **show access-list** [*name*]—Displays the access lists, including the line number for each ACE and hit counts. Include an ACL name or you will see all access lists.
- **show running-config access-list** [*name*]—Displays the current running access-list configuration. Include an ACL name or you will see all access lists.

History for ACLs

Feature Name	Releases	Description
Extended, standard, webtype ACLs	7.0(1)	ACLs are used to control network access or to specify traffic for many features to act upon. An extended access control list is used for through-the-box access control and several other features. Standard ACLs are used in route maps and VPN filters. Webtype ACLs are used in clientless SSL VPN filtering. EtherType ACLs control non-IP layer 2 traffic. We introduced the following commands: **access-list extended, access-list standard, access-list webtype**, **access-list ethertype**.
Real IP addresses in extended ACLs	8.3(1)	When using NAT or PAT, mapped addresses and ports are no longer used in an ACL for several features. You must use the real, untranslated addresses and ports for these features. Using the real address and port means that if the NAT configuration changes, you do not need to change the ACLs.
Support for Identity Firewall in extended ACLs	8.4(2)	You can now use identity firewall users and groups for the source and destination. You can use an identity firewall ACL with access rules, AAA rules, and for VPN authentication. We modified the following commands: **access-list extended**.
EtherType ACL support for IS-IS traffic	8.4(5), 9.1(2)	In transparent firewall mode, the ASA can now control IS-IS traffic using an EtherType ACL. We modified the following command: **access-list ethertype** {**permit** | **deny**} **isis**.
Support for Cisco TrustSec in extended ACLs	9.0(1)	You can now use Cisco TrustSec security groups for the source and destination. You can use an identity firewall ACL with access rules. We modified the following commands: **access-list extended**.

Feature Name	Releases	Description
Unified extended and webtype ACLs for IPv4 and IPv6	9.0(1)	Extended and webtype ACLs now support IPv4 and IPv6 addresses. You can even specify a mix of IPv4 and IPv6 addresses for the source and destination. The **any** keyword was changed to represent IPv4 and IPv6 traffic. The **any4** and **any6** keywords were added to represent IPv4-only and IPv6-only traffic, respectively. The IPv6-specific ACLs are deprecated. Existing IPv6 ACLs are migrated to extended ACLs. See the release notes for more information about migration. We modified the following commands: **access-list extended**, **access-list webtype**. We removed the following commands: **ipv6 access-list**, **ipv6 access-list webtype**, **ipv6-vpn-filter**.
Extended ACL and object enhancement to filter ICMP traffic by ICMP code	9.0(1)	ICMP traffic can now be permitted/denied based on ICMP code. We introduced or modified the following commands: **access-list extended** , **service-object**, **service**.
Configuration session for editing ACLs and objects. Forward referencing of objects and ACLs in access rules.	9.3(2)	You can now edit ACLs and objects in an isolated configuration session. You can also forward reference objects and ACLs, that is, configure rules and access groups for objects or ACLs that do not yet exist. We introduced the **clear configuration session**, **clear session**, **configure session**, **forward-reference**, and **show configuration session** commands.
ACL support for Stream Control Transmission Protocol (SCTP)	9.5(2)	You can now create ACL rules using the **sctp** protocol, including port specifications. We modified the following command: **access-list extended** .
Ethertype rule support for the IEEE 802.2 Logical Link Control packet's Destination Service Access Point address.	9.6(2)	You can now write Ethertype access control rules for the IEEE 802.2 Logical Link Control packet's Destination Service Access Point address. Because of this addition, the **bpdu** keyword no longer matches the intended traffic. Rewrite **bpdu** rules for **dsap 0x42**. We modified the following commands: **access-list ethertype**
Support in routed mode for Ethertype rules on bridge group member interfaces and extended access rules on Bridge Group Virtual Interfaces (BVI).	9.7(1)	You can now create Ethertype ACLs and apply them to bridge group member interfaces in routed mode. You can also apply extended access rules to the Bridge Virtual Interface (BVI) in addition to the member interfaces. We modified the following commands: **access-group**, **access-list ethertype** .

Feature Name	Releases	Description
EtherType access control list changes.	9.9(1)	EtherType access control lists now support Ethernet II IPX (EII IPX). In addition, new keywords are added to the DSAP keyword to support common DSAP values: BPDU (0x42), IPX (0xE0), Raw IPX (0xFF), and ISIS (0xFE). Consequently, existing EtherType access control entries that use the BPDU or ISIS keywords will be converted automatically to use the DSAP specification, and rules for IPX will be converted to 3 rules (DSAP IPX, DSAP Raw IPX, and EII IPX). In addition, packet capture that uses IPX as an EtherType value has been deprecated, because IPX corresponds to 3 separate EtherTypes. We modified the following commands: **access-list ethertype** added the new keywords **eii-ipx** and **dsap** {**bpdu** | **ipx** | **isis** | **raw-ipx**}; **capture ethernet-type** no longer supports the **ipx** keyword.
Support for network-service object groups in extended ACLs.	9.17(1)	You can use network-service objects as the source and destination criteria in extended ACLs and access control rules. We changed the following command: **access-list extended**.
Forward referencing of ACLs and objects is always enabled. In addition, object group search for access control is now enabled by default.	9.18(1)	You can refer to ACLs or network objects that do not yet exist when configuring access groups or access rules. In addition, object group search is now enabled by default for access control for *new* deployments. Upgrading devices will continue to have this command disabled. If you want to enable it (recommended), you must do so manually. We removed the **forward-reference enable** command, and changed the default for **object-group-search access-control** to enabled.



CHAPTER 4

Access Rules

This chapter describes how to control network access through or to the ASA using access rules. You use access rules to control network access in both routed and transparent firewall modes. In transparent mode, you can use both access rules (for Layer 3 traffic) and EtherType rules (for Layer 2 traffic).

Note To access the ASA interface for management access, you do not also need an access rule allowing the host IP address. You only need to configure management access according to the general operations configuration guide.

- Controlling Network Access, on page 51
- Licensing for Access Rules, on page 57
- Guidelines for Access Control, on page 57
- Configure Access Control, on page 59
- Monitoring Access Rules, on page 61
- Configuration Examples for Permitting or Denying Network Access, on page 63
- History for Access Rules, on page 64

Controlling Network Access

Access rules determine which traffic is allowed through the ASA. There are several different layers of rules that work together to implement your access control policy:

- Extended access rules (Layer 3+ traffic) assigned to interfaces—You can apply separate rule sets (ACLs) in the inbound and outbound directions. An extended access rule permits or denies traffic based on the source and destination traffic criteria.
- Extended access rules (Layer 3+ traffic) assigned to Bridge Virtual Interfaces (BVI; routed mode)—If you name a BVI, you can apply separate rule sets in the inbound and outbound direction, and you can also apply rule sets to the bridge group member interfaces. When both the BVI and member interface have access rules, the order of processing depends on direction. Inbound, the member access rules are evaluated first, then the BVI access rules. Outbound, the BVI rules are considered first, then the member interface rules.
- Extended access rules assigned globally—You can create a single global rule set, which serves as your default access control. The global rules are applied after interface rules.

- Management access rules (Layer 3+ traffic)—You can apply a single rule set to cover traffic directed at an interface, which would typically be management traffic. In the CLI, these are “control plane” access groups. For ICMP traffic directed at the device, you can alternatively configure ICMP rules.
- EtherType rules (Layer 2 traffic) assigned to interfaces (bridge group member interfaces only)—You can apply separate rule sets in the inbound and outbound directions. EtherType rules control network access for non-IP traffic. An EtherType rule permits or denies traffic based on the EtherType. You can also apply extended access rules to bridge group member interfaces to control Layer 3+ traffic.

General Information About Rules

The following topics provide general information about access rules and EtherType rules.

Interface Access Rules and Global Access Rules

You can apply an access rule to a specific interface, or you can apply an access rule globally to all interfaces. You can configure global access rules in conjunction with interface access rules, in which case, the specific inbound interface access rules are always processed before the general global access rules. Global access rules apply only to inbound traffic.

Inbound and Outbound Rules

You can configure access rules based on the direction of traffic:

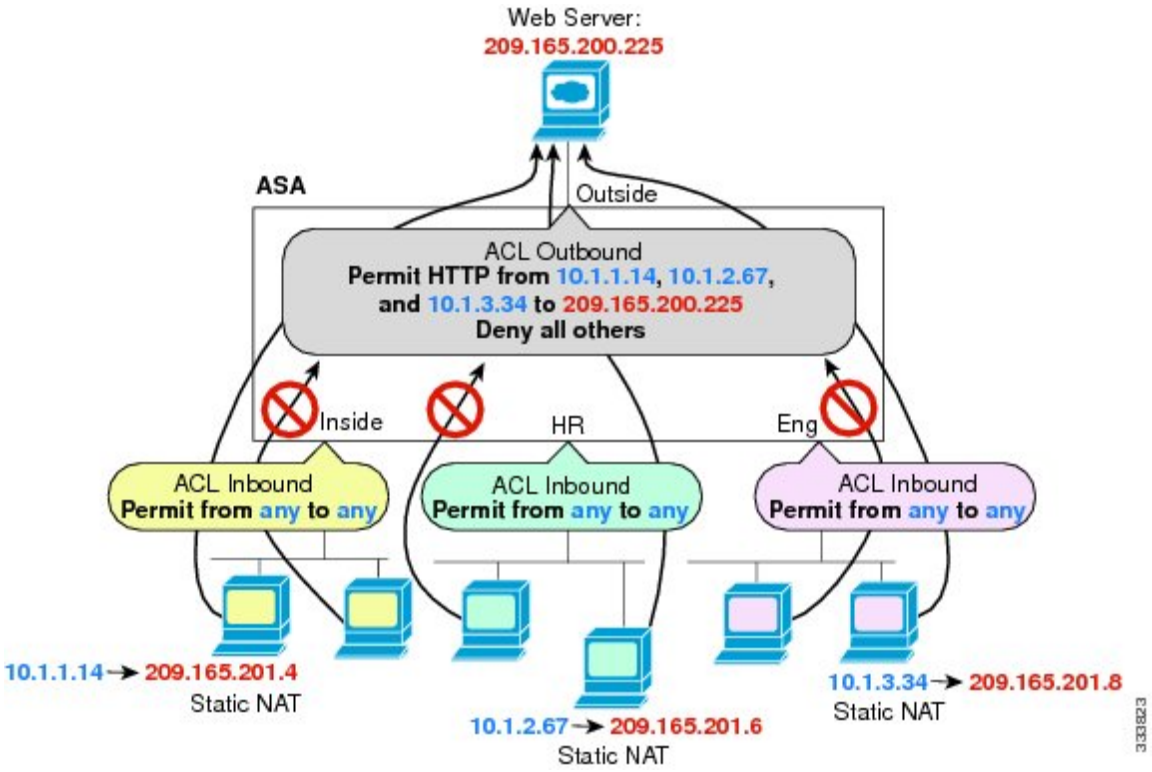
- Inbound—Inbound access rules apply to traffic as it enters an interface. Global and management access rules are always inbound.
- Outbound—Outbound rules apply to traffic as it exits an interface.



Note “Inbound” and “outbound” refer to the application of an ACL on an interface, either to traffic entering the ASA on an interface or traffic exiting the ASA on an interface. These terms do not refer to the movement of traffic from a lower security interface to a higher security interface, commonly known as inbound, or from a higher to lower interface, commonly known as outbound.

An outbound ACL is useful, for example, if you want to allow only certain hosts on the inside networks to access a web server on the outside network. Rather than creating multiple inbound ACLs to restrict access, you can create a single outbound ACL that allows only the specified hosts. (See the following figure.) The outbound ACL prevents any other hosts from reaching the outside network.

Figure 2: Outbound ACL



See the following commands for this example:

```
hostname(config)# access-list OUTSIDE extended permit tcp host 10.1.1.14 host 209.165.200.225
 eq www
hostname(config)# access-list OUTSIDE extended permit tcp host 10.1.2.67 host 209.165.200.225
 eq www
hostname(config)# access-list OUTSIDE extended permit tcp host 10.1.3.34 host 209.165.200.225
 eq www
hostname(config)# access-group OUTSIDE out interface outside
```

Rule Order

The order of rules is important. When the ASA decides whether to forward or drop a packet, the ASA tests the packet against each rule in the order in which the rules are listed in the applied ACL. After a match is found, no more rules are checked. For example, if you create an access rule at the beginning that explicitly permits all traffic for an interface, no further rules are ever checked.

Implicit Permits

Unicast IPv4 and IPv6 traffic from a higher security interface to a lower security interface is allowed through by default. This includes traffic between standard routed interfaces and Bridge Virtual Interfaces (BVI) in routed mode.

For bridge group member interfaces, this implicit permit from a higher to a lower security interface applies to interfaces within the same bridge group only. There are no implicit permits between a bridge group member interface and a routed interface or a member of a different bridge group.

Bridge group member interfaces (routed or transparent mode) also allow the following by default:

- ARPs in both directions. (You can control ARP traffic using ARP inspection, but you cannot control it by access rule.)
- BPDUs in both directions. (You can control these using Ethertype rules.)

For other traffic, you need to use either an extended access rule (IPv4 and IPv6) or an EtherType rule (non-IP).

Implicit Deny

ACLs have an implicit deny at the end of the list, so unless you explicitly permit it, traffic cannot pass. For example, if you want to allow all users to access a network through the ASA except for particular addresses, then you need to deny the particular addresses and then permit all others.

For management (control plane) ACLs, which control to-the-box traffic, there is no implicit deny at the end of a set of management rules for an interface. Instead, any connection that does not match a management access rule is then evaluated by regular access control rules.

For EtherType ACLs, the implicit deny at the end of the ACL does not affect IP traffic or ARPs; for example, if you allow EtherType 8037, the implicit deny at the end of the ACL does not now block any IP traffic that you previously allowed with an extended ACL (or implicitly allowed from a high security interface to a low security interface). However, if you explicitly deny all traffic with an EtherType rule, then IP and ARP traffic is denied; only physical protocol traffic, such as auto-negotiation, is still allowed.

If you configure a global access rule, then the implicit deny comes *after* the global rule is processed. See the following order of operations:

1. Interface access rule.
2. For bridge group member interfaces, the Bridge Virtual Interface (BVI) access rule.
3. Global access rule.
4. Implicit deny.

NAT and Access Rules

Access rules always use the real IP addresses when determining an access rule match, even if you configure NAT. For example, if you configure NAT for an inside server, 10.1.1.5, so that it has a publicly routable IP address on the outside, 209.165.201.5, then the access rule to allow the outside traffic to access the inside server needs to reference the server's real IP address (10.1.1.5), and not the mapped address (209.165.201.5).

Same Security Level Interfaces and Access Rules

Each interface has a security level, and security level checking is performed before access rules are considered. Thus, even if you allow a connection in an access rule, it can be blocked due to same-security-level checking at the interface level. You might want to ensure that your configuration allows same-security-level connections so that your access rules are always considered for permit/deny decisions.

- Connections between the same security level ingress and egress interfaces are subject to the same-security-traffic inter-interface check.

  To allow these connections, enter the **same-security-traffic permit inter-interface** command.

To allow these connections, choose **Configuration** > **Device Setup** > **Interface Settings** > **Interfaces**, then select the **Enable traffic between two or more interfaces which are configured with the same security levels** option.

- Connections with the same ingress and egress interfaces are subject to the same-security-traffic intra-interface check.

  To allow these connections, enter the **same-security-traffic permit intra-interface** command.

  To allow these connections, choose **Configuration** > **Device Setup** > **Interface Settings** > **Interfaces**, then select the **Enable traffic between two or more hosts connected to the same interface** option.

Extended Access Rules

This section describes information about extended access rules.

Extended Access Rules for Returning Traffic

For TCP, UDP, and SCTP connections for both routed and transparent mode, you do not need an access rule to allow returning traffic because the ASA allows all returning traffic for established, bidirectional connections.

For connectionless protocols such as ICMP, however, the ASA establishes unidirectional sessions, so you either need access rules to allow ICMP in both directions (by applying ACLs to the source and destination interfaces), or you need to enable the ICMP inspection engine. The ICMP inspection engine treats ICMP sessions as bidirectional connections. For example, to control ping, specify **echo-reply** (**0**) (ASA to host) or **echo** (**8**) (host to ASA).

Allowing Broadcast and Multicast Traffic

In routed firewall mode, broadcast and multicast traffic is blocked even if you allow it in an access rule, including unsupported dynamic routing protocols and DHCP. You must configure the dynamic routing protocols or DHCP relay to allow this traffic.

For interfaces that are members of the same bridge group in transparent or routed firewall mode, you can allow any IP traffic through using access rules.

Note Because these special types of traffic are connectionless, you need to apply an access rule to both the inbound and outbound interfaces, so returning traffic is allowed through.

The following table lists common traffic types that you can allow using access rules between interfaces that are members of the same bridge group.

Table 2: Special Traffic for Access Rules between Members of the Same Bridge Group

Traffic Type	Protocol or Port	Notes
DHCP	UDP ports 67 and 68	If you enable the DHCP server, then the ASA does not pass DHCP packets.
EIGRP	Protocol 88	—
OSPF	Protocol 89	—

Traffic Type	Protocol or Port	Notes
Multicast streams	The UDP ports vary depending on the application.	Multicast streams are always destined to a Class D address (224.0.0.0 to 239.x.x.x).
RIP (v1 or v2)	UDP port 520	—

Management Access Rules

You can configure access rules that control management traffic destined to the ASA. Access control rules for to-the-box management traffic (defined by such commands as **http**, **ssh**, or **telnet**) have higher precedence than a management access rule applied with the **control-plane** option. Therefore, such permitted management traffic will be allowed to come in even if explicitly denied by the to-the-box ACL.

Unlike regular access rules, there is no implicit deny at the end of a set of management rules for an interface. Instead, any connection that does not match a management access rule is then evaluated by regular access control rules.

Alternatively, you can use ICMP rules to control ICMP traffic to the device. Use regular extended access rules to control ICMP traffic through the device.

EtherType Rules

This section describes EtherType rules.

Supported EtherTypes and Other Traffic

An EtherType rule controls the following:

- EtherType identified by a 16-bit hexadecimal number, including common types IPX and MPLS unicast or multicast.
- Ethernet V2 frames.
- BPDUs, which are permitted by default. BPDUs are SNAP-encapsulated, and the ASA is designed to specifically handle BPDUs.
- Trunk port (Cisco proprietary) BPDUs. Trunk BPDUs have VLAN information inside the payload, so the ASA modifies the payload with the outgoing VLAN if you allow BPDUs.
- Intermediate System to Intermediate System (IS-IS).
- The IEEE 802.2 Logical Link Control packet. You can control access based on the Destination Service Access Point address.

The following types of traffic are not supported:

- 802.3-formatted frames—These frames are not handled by the rule because they use a length field as opposed to a type field.

EtherType Rules for Returning Traffic

Because EtherTypes are connectionless, you need to apply the rule to both interfaces if you want traffic to pass in both directions.

Allowing MPLS

If you allow MPLS, ensure that Label Distribution Protocol and Tag Distribution Protocol TCP connections are established through the ASA by configuring both MPLS routers connected to the ASA to use the IP address on the ASA interface as the router-id for LDP or TDP sessions. (LDP and TDP allow MPLS routers to negotiate the labels (addresses) used to forward packets.)

On Cisco IOS routers, enter the appropriate command for your protocol, LDP or TDP. The *interface* is the interface connected to the ASA.

mpls ldp router-id *interface* **force**

Or

tag-switching tdp router-id *interface* **force**

Licensing for Access Rules

Access control rules do not require a special license.

However, to use **sctp** as the protocol in a rule, you must have a Carrier license.

Guidelines for Access Control

IPv6 Guidelines

Supports IPv6. The source and destination addresses can include any mix of IPv4 and IPv6 addresses.

Per-User ACL Guidelines

- The per-user ACL uses the value in the **timeout uauth** command, but it can be overridden by the AAA per-user session timeout value.
- If traffic is denied because of a per-user ACL, syslog message 109025 is logged. If traffic is permitted, no syslog message is generated. The **log** option in the per-user ACL has no effect.

Additional Guidelines and Limitations

- Over time, your list of access rules can grow to include many obsolete rules. Eventually, the ACLs for the access groups can become so large that they impact overall system performance. If you find that the system is having issues sending syslog messages, communicating for failover synchronization, establishing and maintaining SSH/HTTPS management access connections, and so forth, you might need to prune your access rules. In general, you should actively maintain your rule lists to remove obsolete rules, rules that are never hit, FQDN objects that can no longer be resolved, and so forth. Also consider implementing object group search.
- Object group search is enabled by default for new deployments.

  You can reduce the memory required to search access rules by enabling object group search, but this is at the expense rule of lookup performance and increased CPU utilization. When enabled, object group search does not expand network or service objects, but instead searches access rules for matches based

on those group definitions. You can set this option using the **object-group-search access-control** command.

You can use the **object-group-search threshold** command to enable a threshold to help prevent performance degradation. When operating with a threshold, for each connection, both the source and destination IP addresses are matched against network objects. If the number of objects matched by the source address times the number matched by the destination address exceeds 10,000, the connection is dropped. Configure your rules to prevent an excessive number of matches.

Note Object group search works with network and service objects only. It does not work with security group or user objects. Do not enable this feature if the ACLs include security groups. The result can be inactive ACLs or other unexpected behavior.

- You can improve system performance and reliability by using the transactional commit model for access groups. However, transactional commit is not recommended if you use FQDN objects for hostnames whose resolution can frequently change, as the access group compilation might never completely resolve. See the basic settings chapter in the general operations configuration guide for more information. Use the **asp rule-engine transactional-commit access-group** command.
- In ASDM, rule descriptions are based on the access list remarks that come before the rule in the ACL; for new rules you create in ASDM, any descriptions are also configured as remarks before the related rule. However, the packet tracer in ASDM matches the remark that is configured after the matching rule in the CLI.
- To use fully-qualified domain name (FQDN) network objects as source or destination criteria, you must also configure DNS for the data interfaces.

  Note that controlling access by FQDN is a best-effort mechanism. Consider the following points:

  - Because DNS replies can be spoofed, only use fully trusted internal DNS servers.
  - Some FQDNs, especially for very popular servers, can have hundreds if not thousands of IP addresses, and these can frequently change. Because the system uses cached DNS lookup results, users might get addresses that are not yet in the cache, and their connections will not match the FQDN rule. Rules that use FQDN network objects function effectively only for names that resolve to fewer than 100 addresses.

    We recommend that you do not create network object rules for an FQDN that resolves to more than 100 addresses, as the likelihood of the address in a connection being one that has been resolved and available in the DNS cache on the device is low.
  - For popular FQDNs, different DNS servers can return a different set of IP addresses. Thus, if your users use a different DNS server than the one you configure, FQDN-based access control rules might not apply to all IP addresses for the site that are used by your clients, and you will not get the intended results for your rules.
  - Some FQDN DNS entries have very short time to live (TTL) values. This can result in frequent recompliation of the lookup table, which can impact overall system performance.

Configure Access Control

The following topics explain how to configure access control.

Configure an Access Group

Before you can create an access group, create the ACL.

To bind an ACL to an interface or to apply it globally, use the following command:

access-group *access_list* { {**in** | **out**} **interface** *interface_name* [**per-user-override** | **control-plane**] | **global**}

For an interface-specific access group:

- Specify the extended or EtherType ACL name. You can configure one **access-group** command per ACL type per interface per direction, and one control plane ACL. The control plane ACL must be an extended ACL. Ethertype ACLs are allowed on bridge group member interfaces only. For bridge groups in routed mode, you can specify extended ACLs for each direction on both the Bridge Virtual Interface (BVI) and each bridge group member interface.
- The **in** keyword applies the ACL to inbound traffic. The **out** keyword applies the ACL to the outbound traffic.
- Specify the **interface** name.
- The per-user-override keyword (for inbound extended ACLs only) allows dynamic user ACLs that are downloaded for user authorization to override the ACL assigned to the interface. For example, if the interface ACL denies all traffic from 10.0.0.0, but the dynamic ACL permits all traffic from 10.0.0.0, then the dynamic ACL overrides the interface ACL for that user.

  By default, VPN remote access traffic is not matched against interface ACLs. However, if you use the **no sysopt connection permit-vpn** command to turn off this bypass, the behavior depends on whether there is a **vpn-filter** applied in the group policy and whether you set the **per-user-override** option:

  - No **per-user-override**, no **vpn-filter**—Traffic is matched against the interface ACL.
  - No **per-user-override**, **vpn-filter**—Traffic is matched first against the interface ACL, then against the VPN filter.
  - **per-user-override**, **vpn-filter**—Traffic is matched against the VPN filter only.

- The **control-plane** keyword specifies if the extended ACL is for to-the-box traffic.

  Unlike regular access rules, there is no implicit deny at the end of a set of management (control plane) rules for an interface. Instead, any connection that does not match a management access rule is then evaluated by regular access control rules.

For a global access group, specify the **global** keyword to apply the extended ACL to the inbound direction of all interfaces.

Example

The following example shows how to use the **access-group** command:

```
hostname(config)# access-list outside_access permit tcp any host 209.165.201.3 eq 80
hostname(config)# access-group outside_access in interface outside
```

The **access-list** command lets any host access the host address using port 80. The **access-group** command specifies that the **access-list** command applies to traffic entering the outside interface.

Configure ICMP Access Rules

By default, you can send ICMP packets to any interface using either IPv4 or IPv6, with these exceptions:

- The ASA does not respond to ICMP echo requests directed to a broadcast address.
- The ASA only responds to ICMP traffic sent to the interface that traffic comes in on; you cannot send ICMP traffic through an interface to a far interface.

To protect the device from attacks, you can use ICMP rules to limit ICMP access to interfaces to particular hosts, networks, or ICMP types. ICMP rules function like access rules, where the rules are ordered, and the first rule that matches a packet defines the action.

If you configure any ICMP rule for an interface, an implicit deny ICMP rule is added to the end of the ICMP rule list, changing the default behavior. Thus, if you want to simply deny a few message types, you must include a permit any rule at the end of the ICMP rule list to allow the remaining message types.

We recommend that you always grant permission for the ICMP unreachable message type (type 3). Denying ICMP unreachable messages disables ICMP path MTU discovery, which can halt IPsec and PPTP traffic. Additionally ICMP packets in IPv6 are used in the IPv6 neighbor discovery process.

Procedure

Step 1 Create rules for ICMP traffic.

icmp {**permit** | **deny**} {**host** *ip_address* | *ip_address mask* | **any**} [*icmp_type*] *interface_name*

If you do not specify an *icmp_type*, the rule applies to all types. You can enter the number or the name. To control ping, specify echo-reply (0) (ASA-to-host) or echo (8) (host-to-ASA).

For the address, you can apply the rule to **any** address, to a single **host**, or to a network (*ip_address mask*).

Step 2 Create rules for ICMPv6 (IPv6) traffic.

ipv6 icmp {**permit** | **deny**} {**host** *ipv6_address* | *ipv6-network*/*prefix-length* | **any**} [*icmp_type*] *interface_name*

If you do not specify an *icmp_type*, the rule applies to all types.

For the address, you can apply the rule to **any** address, to a single **host**, or to a network (*ipv6-network*/*prefix-length*).

Step 3 (Optional.) Set rate limits on ICMP Unreachable messages so that the ASA will appear on trace route output.

icmp unreachable rate-limit *rate* **burst-size** *size*

The rate limit can be 1-100, with 1 being the default. The burst size can be 1-10. The burst size number of reponses are sent, but subsequent replies are not sent until the rate limit is reached.

Example:

Increasing the rate limit, along with enabling the **set connection decrement-ttl** command in a service policy, is required to allow a traceroute through the ASA that shows the ASA as one of the hops. For example, the following policy increases the rate limit and decrements the time-to-live (TTL) value for all traffic through the ASA.

```
icmp unreachable rate-limit 50 burst-size 10
class-map global-class
  match any
policy-map global_policy
  class global-class
  set connection decrement-ttl
```

Examples

The following example shows how to allow all hosts except the one at 10.1.1.15 to use ICMP to the inside interface:

```
hostname(config)# icmp deny host 10.1.1.15 inside
hostname(config)# icmp permit any inside
```

The following example shows how to allow the host at 10.1.1.15 to use only ping to the inside interface:

```
hostname(config)# icmp permit host 10.1.1.15 inside
```

The following example shows how to deny all ping requests and permit all packet-too-big messages (to support path MTU discovery) at the outside interface:

```
hostname(config)# ipv6 icmp deny any echo-reply outside
hostname(config)# ipv6 icmp permit any packet-too-big outside
```

The following example shows how to permit host 2000:0:0:4::2 or hosts on prefix 2001::/64 to ping the outside interface:

```
hostname(config)# ipv6 icmp permit host 2000:0:0:4::2 echo-reply outside
hostname(config)# ipv6 icmp permit 2001::/64 echo-reply outside
hostname(config)# ipv6 icmp permit any packet-too-big outside
```

Monitoring Access Rules

To monitor network access, enter the following commands:

- **clear access-list** *id* **counters**

  Clear the hit counts for the access list.

- **show access-list** [*name*]

  Displays the access lists, including the line number for each ACE and hit counts. Include an ACL name or you will see all access lists.

- **show running-config access-group**

  Displays the current ACL bound to the interfaces.

Evaluating Syslog Messages for Access Rules

Use a syslog event viewer, such as the one in ASDM, to view messages related to access rules.

If you use default logging, you see syslog message 106023 for explicitly denied flows only. Traffic that matches the "implicit deny" entry that ends the rule list is not logged.

If the ASA is attacked, the number of syslog messages for denied packets can be very large. We recommend that you instead enable logging using syslog message 106100, which provides statistics for each rule (including permit rules) and enables you to limit the number of syslog messages produced. Alternatively, you can disable all logging for a given rule.

When you enable logging for message 106100, if a packet matches an ACE, the ASA creates a flow entry to track the number of packets received within a specific interval. The ASA generates a syslog message at the first hit and at the end of each interval, identifying the total number of hits during the interval and the time stamp for the last hit. At the end of each interval, the ASA resets the hit count to 0. If no packets match the ACE during an interval, the ASA deletes the flow entry. When you configure logging for a rule, you can control the interval and even the severity level of the log message, per rule.

A flow is defined by the source and destination IP addresses, protocols, and ports. Because the source port might differ for a new connection between the same two hosts, you might not see the same flow increment because a new flow was created for the connection.

Permitted packets that belong to established connections do not need to be checked against ACLs; only the initial packet is logged and included in the hit count. For connectionless protocols, such as ICMP, all packets are logged, even if they are permitted, and all denied packets are logged.

See the *syslog messages guide* for detailed information about these messages.



Tip When you enable logging for message 106100, if a packet matches an ACE, the ASA creates a flow entry to track the number of packets received within a specific interval. The ASA has a maximum of 32 K logging flows for ACEs. A large number of flows can exist concurrently at any point of time. To prevent unlimited consumption of memory and CPU resources, the ASA places a limit on the number of concurrent *deny* flows; the limit is placed on deny flows only (not on permit flows) because they can indicate an attack. When the limit is reached, the ASA does not create a new deny flow for logging until the existing flows expire, and issues message 106101. You can control the frequency of this message using the **access-list alert-interval** *secs* command, and the maximum number of deny flows cached using the **access-list deny-flow-max** *number* command.

Configuration Examples for Permitting or Denying Network Access

Following are some typical configuration examples for permitting or denying network access.

Extended ACL Examples

The following example adds a network object for inside server 1, performs static NAT for the server, and enables access from the outside for inside server 1.

```
hostname(config)# object network inside-server1
hostname(config)# host 10.1.1.1
hostname(config)# nat (inside,outside) static 209.165.201.12

hostname(config)# access-list outside_access extended permit tcp any object inside-server1
 eq www
hostname(config)# access-group outside_access in interface outside
```

The following example allows all hosts to communicate between the inside and hr networks but only specific hosts to access the outside network:

```
hostname(config)# access-list ANY extended permit ip any any
hostname(config)# access-list OUT extended permit ip host 209.168.200.3 any
hostname(config)# access-list OUT extended permit ip host 209.168.200.4 any

hostname(config)# access-group ANY in interface inside
hostname(config)# access-group ANY in interface hr
hostname(config)# access-group OUT out interface outside
```

The following example uses object groups to permit specific traffic on the inside interface:

```
!
hostname (config)# object-group service myaclog
hostname (config-service)# service-object tcp source range 2000 3000
hostname (config-service)# service-object tcp source range 3000 3010 destinatio$
hostname (config-service)# service-object ipsec
hostname (config-service)# service-object udp destination range 1002 1006
hostname (config-service)# service-object icmp echo

hostname(config)# access-list outsideacl extended permit object-group myaclog interface
inside any
```

EtherType Examples

For example, the following sample ACL allows common EtherTypes originating on the inside interface:

```
hostname(config)# access-list ETHER ethertype permit ipx
INFO: ethertype ipx is saved to config as ethertype eii-ipx
INFO: ethertype ipx is saved to config as ethertype dsap ipx
INFO: ethertype ipx is saved to config as ethertype dsap raw-ipx
hostname(config)# access-list ETHER ethertype permit mpls-unicast
hostname(config)# access-group ETHER in interface inside
```

The following example allows some EtherTypes through the ASA, but it denies all others:

```
hostname(config)# access-list ETHER ethertype permit 0x1234
hostname(config)# access-list ETHER ethertype permit mpls-unicast
hostname(config)# access-group ETHER in interface inside
hostname(config)# access-group ETHER in interface outside
```

The following example denies traffic with EtherType 0x1256 but allows all others on both interfaces:

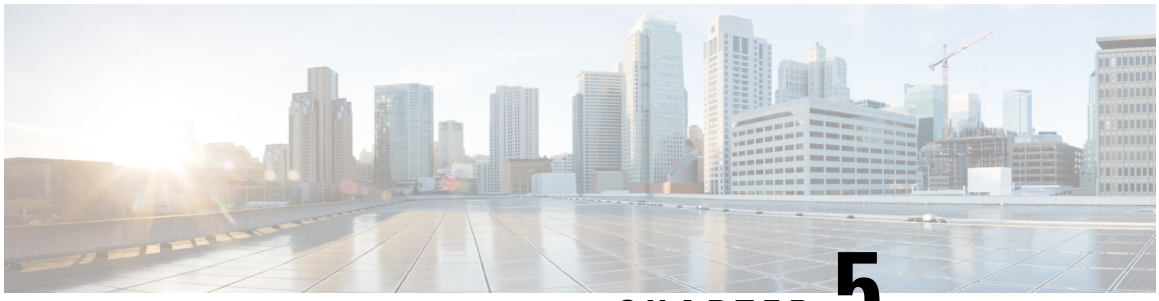
```
hostname(config)# access-list nonIP ethertype deny 1256
hostname(config)# access-list nonIP ethertype permit any
hostname(config)# access-group nonIP in interface inside
hostname(config)# access-group nonIP in interface outside
```

History for Access Rules

Feature Name	Platform Releases	Description
Interface access rules	7.0(1)	Controlling network access through the ASA using ACLs. We introduced the following command: **access-group**.
Global access rules	8.3(1)	Global access rules were introduced. We modified the following command: **access-group.**
Support for Identity Firewall	8.4(2)	You can now use identity firewall users and groups for the source and destination. You can use an identity firewall ACL with access rules, AAA rules, and for VPN authentication. We modified the following commands: **access-list extended**.
EtherType ACL support for IS-IS traffic	8.4(5), 9.1(2)	In transparent firewall mode, the ASA can now pass IS-IS traffic using an EtherType ACL. We modified the following command: **access-list ethertype** {**permit** | **deny**} **isis**.
Support for TrustSec	9.0(1)	You can now use TrustSec security groups for the source and destination. You can use an identity firewall ACL with access rules. We modified the following commands: **access-list extended**.

Feature Name	Platform Releases	Description
Unified ACL for IPv4 and IPv6	9.0(1)	ACLs now support IPv4 and IPv6 addresses. You can even specify a mix of IPv4 and IPv6 addresses for the source and destination. The **any** keyword was changed to represent IPv4 and IPv6 traffic. The **any4** and **any6** keywords were added to represent IPv4-only and IPv6-only traffic, respectively. The IPv6-specific ACLs are deprecated. Existing IPv6 ACLs are migrated to extended ACLs. See the release notes for more information about migration. We modified the following commands: **access-list extended**, **access-list webtype**. We removed the following commands: **ipv6 access-list**, **ipv6 access-list webtype**, **ipv6-vpn-filter**
Extended ACL and object enhancement to filter ICMP traffic by ICMP code	9.0(1)	ICMP traffic can now be permitted/denied based on ICMP code. We introduced or modified the following commands: **access-list extended** , **service-object, service**.
Transactional Commit Model on Access Group Rule Engine	9.1(5)	When enabled, a rule update is applied after the rule compilation is completed; without affecting the rule matching performance. We introduced the following commands: **asp rule-engine transactional-commit**, **show running-config asp rule-engine transactional-commit**, **clear configure asp rule-engine transactional-commit**.
Configuration session for editing ACLs and objects. Forward referencing of objects and ACLs in access rules.	9.3(2)	You can now edit ACLs and objects in an isolated configuration session. You can also forward reference objects and ACLs, that is, configure rules and access groups for objects or ACLs that do not yet exist. We introduced the **clear config-session**, **clear session**, **configure session**, **forward-reference**, and **show config-session** commands.
Access rule support for Stream Control Transmission Protocol (SCTP)	9.5(2)	You can now create access rules using the **sctp** protocol, including port specifications. We modified the following command: **access-list extended** .
Ethertype rule support for the IEEE 802.2 Logical Link Control packet's Destination Service Access Point address.	9.6(2)	You can now write Ethertype access control rules for the IEEE 802.2 Logical Link Control packet's Destination Service Access Point address. Because of this addition, the **bpdu** keyword no longer matches the intended traffic. Rewrite **bpdu** rules for **dsap 0x42**. We modified the following commands: **access-list ethertype**
Support in routed mode for Ethertype rules on bridge group member interfaces and extended access rules on Bridge Group Virtual Interfaces (BVI).	9.7(1)	You can now create Ethertype ACLs and apply them to bridge group member interfaces in routed mode. You can also apply extended access rules to the Bridge Virtual Interface (BVI) in addition to the member interfaces. We modified the following commands: **access-group**, **access-list ethertype** .

Feature Name	Platform Releases	Description
EtherType access control list changes.	9.9(1)	EtherType access control lists now support Ethernet II IPX (EII IPX). In addition, new keywords are added to the DSAP keyword to support common DSAP values: BPDU (0x42), IPX (0xE0), Raw IPX (0xFF), and ISIS (0xFE). Consequently, existing EtherType access control entries that use the BPDU or ISIS keywords will be converted automatically to use the DSAP specification, and rules for IPX will be converted to 3 rules (DSAP IPX, DSAP Raw IPX, and EII IPX). In addition, packet capture that uses IPX as an EtherType value has been deprecated, because IPX corresponds to 3 separate EtherTypes. We modified the following commands: **access-list ethertype** added the new keywords **eii-ipx** and **dsap** {**bpdu** | **ipx** | **isis** | **raw-ipx**}; **capture ethernet-type** no longer supports the **ipx** keyword.
The object group search threshold is now disabled by default.	9.12(1)	If you enabled object group search, the feature was subject to a threshold to help prevent performance degradation. That threshold is now disabled by default. You can enable it by using the **object-group-search threshold** command. We added the following command: **object-group-search threshold**.
Forward referencing of ACLs and objects is always enabled. In addition, object group search for access control is now enabled by default.	9.18(1)	You can refer to ACLs or network objects that do not yet exist when configuring access groups or access rules. In addition, object group search is now enabled by default for access control for *new* deployments. Upgrading devices will continue to have this command disabled. If you want to enable it (recommended), you must do so manually. We removed the **forward-reference enable** command, and changed the default for **object-group-search access-control** to enabled.
Object group search optimization.	9.22(1)	The object group search feature has been enhanced to reduce object lookup time when evaluating access control rules to match connections and to reduce CPU overhead. There are no changes to configuring object group search, the optimized behavior happens automatically. We added the following commands in the device CLI, or enhanced command output: **clear asp table network-object**, **debug ac logs**, **packet-tracer**, **show access-list**, **show asp table network-group**, **show object-group**.



CHAPTER 5

ASA and Cisco TrustSec

This chapter describes how to implement Cisco TrustSec for the ASA.

- About Cisco TrustSec, on page 67
- Guidelines for Cisco TrustSec, on page 75
- Configure the ASA to Integrate with Cisco Trustsec, on page 77
- Example for Cisco TrustSec, on page 91
- Secure Client VPN Support for Cisco TrustSec, on page 91
- Monitoring Cisco TrustSec, on page 93
- History for Cisco TrustSec, on page 94

About Cisco TrustSec

Traditionally, security features such as firewalls performed access control based on predefined IP addresses, subnets, and protocols. However, with enterprises transitioning to borderless networks, both the technology used to connect people and organizations and the security requirements for protecting data and networks have evolved significantly. Endpoints are becoming increasingly nomadic and users often employ a variety of endpoints (for example, laptop versus desktop, smart phone, or tablet), which means that a combination of user attributes plus endpoint attributes provide the key characteristics (in addition to existing 6-tuple based rules), that enforcement devices such as switches and routers with firewall features or dedicated firewalls can reliably use for making access control decisions.

As a result, the availability and propagation of endpoint attributes or client identity attributes have become increasingly important requirements to enable security across the customers' networks, at the access, distribution, and core layers of the network, and in the data center.

Cisco TrustSec provides access control that builds upon an existing identity-aware infrastructure to ensure data confidentiality between network devices and integrate security access services on one platform. In the Cisco TrustSec feature, enforcement devices use a combination of user attributes and endpoint attributes to make role-based and identity-based access control decisions. The availability and propagation of this information enables security across networks at the access, distribution, and core layers of the network.

Implementing Cisco TrustSec into your environment has the following advantages:

- Provides a growing mobile and complex workforce with appropriate and more secure access from any device
- Lowers security risks by providing comprehensive visibility of who and what is connecting to the wired or wireless network

- Offers exceptional control over activity of network users accessing physical or cloud-based IT resources
- Reduces total cost of ownership through centralized, highly secure access policy management and scalable enforcement mechanisms
- For more information, see the following URLs:
  - Description of the Cisco TrustSec system and architecture for the enterprise.

    http://www.cisco.com/c/en/us/solutions/enterprise-networks/trustsec/index.html
  - Instructions for deploying the Cisco TrustSec solution in the enterprise, including links to component design guides.

    http://www.cisco.com/c/en/us/solutions/enterprise/design-zone-security/landing_DesignZone_TrustSec.html
  - An overview of the Cisco TrustSec solution when used with the ASA, switches, wireless LAN (WLAN) controllers, and routers.

    http://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/trustsec/solution_overview_c22-591771.pdf
  - The Cisco TrustSec Platform Support Matrix, which lists the Cisco products that support the Cisco TrustSec solution.

    http://www.cisco.com/c/en/us/solutions/enterprise-networks/trustsec/trustsec_matrix.html

About SGT and SXP Support in Cisco TrustSec

In the Cisco TrustSec feature, security group access transforms a topology-aware network into a role-based network, which enables end-to-end policies enforced on the basis of role-based access control (RBAC). Device and user credentials acquired during authentication are used to classify packets by security groups. Every packet entering the Cisco TrustSec cloud is tagged with a security group tag (SGT). The tagging helps trusted intermediaries identify the source identity of the packet and enforce security policies along the data path. An SGT can indicate a privilege level across the domain when the SGT is used to define a security group ACL.

An SGT is assigned to a device through IEEE 802.1X authentication, web authentication, or MAC authentication bypass (MAB), which occurs with a RADIUS vendor-specific attribute. An SGT can be assigned statically to a particular IP address or to a switch interface. An SGT is passed along dynamically to a switch or access point after successful authentication.

The Security-group eXchange Protocol (SXP) is a protocol developed for Cisco TrustSec to propagate the IP-to-SGT mapping database across network devices that do not have SGT-capable hardware support to hardware that supports SGTs and security group ACLs. SXP, a control plane protocol, passes IP-SGT mapping from authentication points (such as legacy access layer switches) to upstream devices in the network.

The SXP connections are point-to-point and use TCP as the underlying transport protocol. SXP uses TCP port 64999 to initiate a connection. Additionally, an SXP connection is uniquely identified by the source and destination IP addresses.

Roles in the Cisco TrustSec Feature

To provide identity and policy-based access enforcement, the Cisco TrustSec feature includes the following roles:

- Access Requester (AR)—Access requesters are endpoint devices that request access to protected resources in the network. They are primary subjects of the architecture and their access privilege depends on their Identity credentials.

  Access requesters include endpoint devices such PCs, laptops, mobile phones, printers, cameras, and MACsec-capable IP phones.

- Policy Decision Point (PDP)—A policy decision point is responsible for making access control decisions. The PDP provides features such as 802.1x, MAB, and web authentication. The PDP supports authorization and enforcement through VLAN, DACL, and security group access (SGACL/SXP/SGT).

  In the Cisco TrustSec feature, the Cisco Identity Services Engine (ISE) acts as the PDP. The Cisco ISE provides identity and access control policy functionality.

- Policy Information Point (PIP)—A policy information point is a source that provides external information (for example, reputation, location, and LDAP attributes) to policy decision points.

  Policy information points include devices such as Session Directory, Sensor IPS, and Communication Manager.

- Policy Administration Point (PAP)—A policy administration point defines and inserts policies into the authorization system. The PAP acts as an identity repository by providing Cisco TrustSec tag-to-user identity mapping and Cisco TrustSec tag-to-server resource mapping.

  In the Cisco TrustSec feature, the Cisco Secure Access Control System (a policy server with integrated 802.1x and SGT support) acts as the PAP.

- Policy Enforcement Point (PEP)—A policy enforcement point is the entity that carries out the decisions (policy rules and actions) made by the PDP for each AR. PEP devices learn identity information through the primary communication path that exists across networks. PEP devices learn the identity attributes of each AR from many sources, such as endpoint agents, authorization servers, peer enforcement devices, and network flows. In turn, PEP devices use SXP to propagate IP-SGT mapping to mutually trusted peer devices across the network.

  Policy enforcement points include network devices such as Catalyst switches, routers, firewalls (specifically the ASA), servers, VPN devices, and SAN devices.

The ASA serves the PEP role in the identity architecture. Using SXP, the ASA learns identity information directly from authentication points and uses it to enforce identity-based policies.

Security Group Policy Enforcement

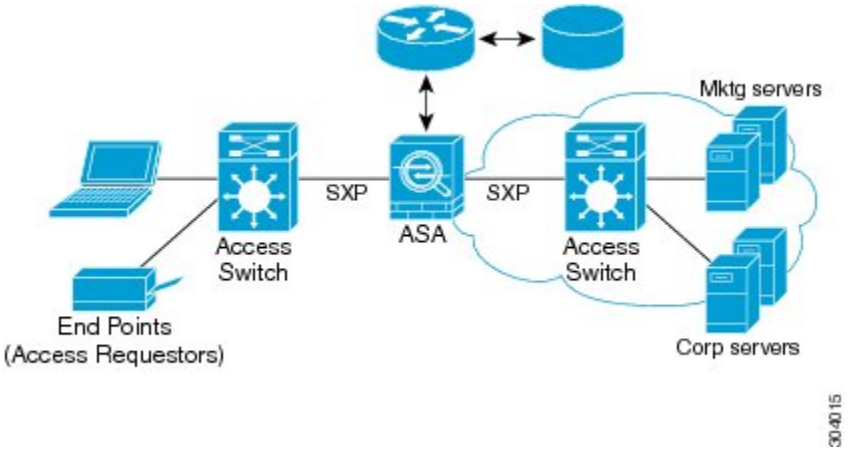
Security policy enforcement is based on security group name. An endpoint device attempts to access a resource in the data center. Compared to traditional IP-based policies configured on firewalls, identity-based policies are configured based on user and device identities. For example, mktg-contractor is allowed to access mktg-servers; mktg-corp-users are allowed to access mktg-server and corp-servers.

The benefits of this type of deployment include the following:

- User group and resource are defined and enforced using single object (SGT) simplified policy management.
- User identity and resource identity are retained throughout the Cisco TrustSec-capable switch infrastructure.

The following figure shows a deployment for security group name-based policy enforcement.

Figure 3: Security Group Name-Based Policy Enforcement Deployment



Implementing Cisco TrustSec allows you to configure security policies that support server segmentation and includes the following features:

- A pool of servers can be assigned an SGT for simplified policy management.
- The SGT information is retained within the infrastructure of Cisco TrustSec-capable switches.
- The ASA can use the IP-SGT mapping for policy enforcement across the Cisco TrustSec domain.
- Deployment simplification is possible because 802.1x authorization for servers is mandatory.

How the ASA Enforces Security Group-Based Policies

Note User-based security policies and security-group based policies can coexist on the ASA. Any combination of network, user-based, and security-group based attributes can be configured in a security policy.

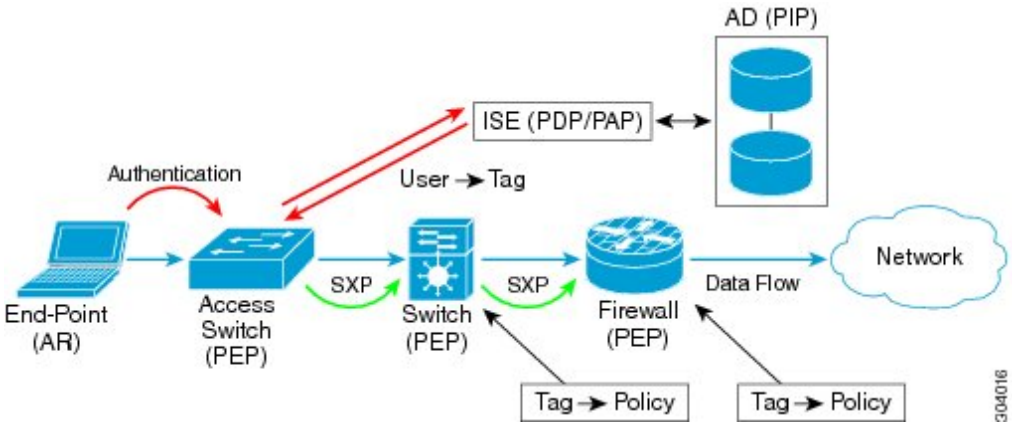
To configure the ASA to function with Cisco TrustSec, you must import a Protected Access Credential (PAC) file from the ISE.

Importing the PAC file to the ASA establishes a secure communication channel with the ISE. After the channel is established, the ASA initiates a PAC secure RADIUS transaction with the ISE and downloads Cisco TrustSec environment data (that is, the security group table). The security group table maps SGTs to security group names. Security group names are created on the ISE and provide user-friendly names for security groups.

The first time that the ASA downloads the security group table, it walks through all entries in the table and resolves all the security group names included in security policies that have been configured on it; then the ASA activates those security policies locally. If the ASA cannot resolve a security group name, it generates a syslog message for the unknown security group name.

The following figure shows how a security policy is enforced in Cisco TrustSec.

Figure 4: Security Policy Enforcement



1. An endpoint device connects to an access layer device directly or via remote access and authenticates with Cisco TrustSec.
2. The access layer device authenticates the endpoint device with the ISE by using authentication methods such as 802.1X or web authentication. The endpoint device passes role and group membership information to classify the device into the appropriate security group.
3. The access layer device uses SXP to propagate the IP-SGT mapping to the upstream devices.
4. The ASA receives the packet and looks up the SGTs for the source and destination IP addresses using the IP-SGT mapping passed by SXP.

   If the mapping is new, the ASA records it in its local IP-SGT Manager database. The IP-SGT Manager database, which runs in the control plane, tracks IP-SGT mapping for each IPv4 or IPv6 address. The database records the source from which the mapping was learned. The peer IP address of the SXP connection is used as the source of the mapping. Multiple sources can exist for each IP-SGT mapped entry.

   If the ASA is configured as a Speaker, the ASA transmits all IP-SGT mapping entries to its SXP peers.
5. If a security policy is configured on the ASA with that SGT or security group name, the ASA enforces the policy. (You can create security policies on the ASA that include SGTs or security group names. To enforce policies based on security group names, the ASA needs the security group table to map security group names to SGTs.)

   If the ASA cannot find a security group name in the security group table and it is included in a security policy, the ASA considers the security group name to be unknown and generates a syslog message. After the ASA refreshes the security group table from the ISE and learns the security group name, the ASA generates a syslog message indicating that the security group name is known.

Effects of Changes to Security Groups on the ISE

The ASA periodically refreshes the security group table by downloading an updated table from the ISE. Security groups can change on the ISE between downloads. These changes are not reflected on the ASA until it refreshes the security group table.

Tip We recommend that you schedule policy configuration changes on the ISE during a maintenance window, then manually refresh the security group table on the ASA to make sure the security group changes have been incorporated.

Handling policy configuration changes in this way maximizes the chances of security group name resolution and immediate activation of security policies.

The security group table is automatically refreshed when the environment data timer expires. You can also trigger a security group table refresh on demand.

If a security group changes on the ISE, the following events occur when the ASA refreshes the security group table:

- Only security group policies that have been configured using security group names need to be resolved with the security group table. Policies that include security group tags are always active.
- When the security group table is available for the first time, all policies with security group names are walked through, security group names are resolved, and policies are activated. All policies with tags are walked through, and syslogs are generated for unknown tags.
- If the security group table has expired, policies continue to be enforced according to the most recently downloaded security group table until you clear it, or a new table becomes available.
- When a resolved security group name becomes unknown on the ASA, it deactivates the security policy; however, the security policy persists in the ASA running configuration.
- If an existing security group is deleted on the PAP, a previously known security group tag can become unknown, but no change in policy status occurs on the ASA. A previously known security group name can become unresolved, and the policy is then inactivated. If the security group name is reused, the policy is recompiled using the new tag.
- If a new security group is added on the PAP, a previously unknown security group tag can become known, a syslog message is generated, but no change in policy status occurs. A previously unknown security group name can become resolved, and associated policies are then activated.
- If a tag has been renamed on the PAP, policies that were configured using tags display the new name, and no change in policy status occurs. Policies that were configured with security group names are recompiled using the new tag value.

Speaker and Listener Roles on the ASA

The ASA supports SXP to send and receive IP-SGT mapping entries to and from other network devices. Using SXP allows security devices and firewalls to learn identity information from access switches without the need for hardware upgrades or changes. SXP can also be used to pass IP-SGT mapping entries from upstream devices (such as data center devices) back to downstream devices. The ASA can receive information from both upstream and downstream directions.

When configuring an SXP connection on the ASA to an SXP peer, you must designate the ASA as a Speaker or a Listener for that connection so that it can exchange Identity information:

- Speaker mode—Configures the ASA so that it can forward all active IP-SGT mapping entries collected on the ASA to upstream devices for policy enforcement.

- Listener mode—Configures the ASA so that it can receive IP-SGT mapping entries from downstream devices (SGT-capable switches) and use that information to create policy definitions.

If one end of an SXP connection is configured as a Speaker, then the other end must be configured as a Listener, and vice versa. If both devices on each end of an SXP connection are configured with the same role (either both as Speakers or both as Listeners), the SXP connection fails and the ASA generates a syslog message.

Multiple SXP connections can learn IP-SGT mapping entries that have been downloaded from the IP-SGT mapping database. After an SXP connection to an SXP peer is established on the ASA, the Listener downloads the entire IP-SGT mapping database from the Speaker. All changes that occur after this are sent only when a new device appears on the network. As a result, the rate of SXP information flow is proportional to the rate at which end hosts authenticate to the network.

IP-SGT mapping entries that have been learned through SXP connections are maintained in the SXP IP-SGT mapping database. The same mapping entries may be learned through different SXP connections. The mapping database maintains one copy for each mapping entry learned. Multiple mapping entries of the same IP-SGT mapping value are identified by the peer IP address of the connection from which the mapping was learned. SXP requests that the IP-SGT Manager add a mapping entry when a new mapping is learned the first time and remove a mapping entry when the last copy in the SXP database is removed.

Whenever an SXP connection is configured as a Speaker, SXP requests that the IP-SGT Manager forward all the mapping entries collected on the device to the peer. When a new mapping is learned locally, the IP-SGT Manager requests that SXP forward it through connections that are configured as Speakers.

Configuring the ASA to be both a Speaker and a Listener for an SXP connection can cause SXP looping, which means that SXP data can be received by an SXP peer that originally transmitted it.

Register the ASA with the ISE

The ASA must be configured as a recognized Cisco TrustSec network device in the ISE before the ASA can successfully import a PAC file. To register the ASA with the ISE, perform the following steps:

Procedure

Step 1 Log into the ISE.

Step 2 Choose **Administration** > **Network Devices** > **Network Devices**.

Step 3 Click **Add**.

Step 4 Enter the IP address of the ASA.

Step 5 When the ISE is being used for user authentication, enter a shared secret in the Authentication Settings area.

When you configure the AAA sever on the ASA, provide the shared secret that you create here on the ISE. The AAA server on the ASA uses this shared secret to communicate with the ISE.

Step 6 Specify a device name, device ID, password, and a download interval for the ASA. See the ISE documentation for how to perform these tasks.

Create a Security Group on the ISE

When configuring the ASA to communicate with the ISE, you specify a AAA server. When configuring the AAA server on the ASA, you must specify a server group. The security group must be configured to use the RADIUS protocol. To create a security group on the ISE, perform the following steps:

Procedure

Step 1 Log into the ISE.

Step 2 Choose **Policy** > **Policy Elements** > **Results** > **Security Group Access** > **Security Group**.

Step 3 Add a security group for the ASA. (Security groups are global and not ASA specific.)

The ISE creates an entry under Security Groups with a tag.

Step 4 In the Security Group Access area, configure device ID credentials and a password for the ASA.

Generate the PAC File

To generate the PAC file, perform the following steps.



Note The PAC file includes a shared key that allows the ASA and ISE to secure the RADIUS transactions that occur between them. For this reason, make sure that you store it securely on the ASA.

Procedure

Step 1 Log into the ISE.

Step 2 Choose **Administration** > **Network Resources** > **Network Devices**.

Step 3 From the list of devices, choose the ASA.

Step 4 Under the Security Group Access (SGA), click **Generate PAC**.

Step 5 To encrypt the PAC file, enter a password.

The password (or encryption key) that you enter to encrypt the PAC file is independent of the password that was configured on the ISE as part of the device credentials.

The ISE generates the PAC file. The ASA can import the PAC file from flash or from a remote server via TFTP, FTP, HTTP, HTTPS, or SMB. (The PAC file does not have to reside on the ASA flash before you can import it.)

Guidelines for Cisco TrustSec

This section includes the guidelines and limitations that you should review before configuring Cisco TrustSec.

Failover

- You can configure security group-based policies on the ASA in both the Active/Active and Active/Standby configurations.
- When the ASA is part of a failover configuration, you must import the PAC file to the primary ASA device. You must also refresh the environment data on the primary device.
- The ASA can communicate with the ISE configured for high availability (HA).
- You can configure multiple ISE servers on the ASA and if the first server is unreachable, it continues to the next server, and so on. However, if the server list is downloaded as part of the Cisco TrustSec environment data, it is ignored.
- If the PAC file downloaded from the ISE expires on the ASA and it cannot download an updated security group table, the ASA continues to enforce security policies based on the last downloaded security group table until the ASA downloads an updated table.

Clustering

- When the ASA is part of a clustering configuration, you must import the PAC file to the control unit.
- When the ASA is part of a clustering configuration, you must refresh the environment data on the control unit.

IPv6

The ASA supports SXP for IPv6 and IPv6-capable network devices. The AAA server must use an IPv4 address.

Layer 2 SGT Imposition

- Supported only on physical interfaces, subinterfaces, and EtherChannel interfaces.
- Not supported on logical interfaces or virtual interfaces, such as a BVI.
- Does not support link encryption using SAP negotiation and MACsec.
- Not supported on failover links.
- Not supported on cluster control links.
- The ASA does not reclassify existing flows if the SGT is changed. Any policy decisions that were made based on the previous SGT remain in force for the life of the flow. However, the ASA can immediately reflect SGT changes on egress packets, even if the packets belong to a flow whose classification was based on a previous SGT.
- Firepower 1010 switch ports and VLAN interfaces do not support Layer 2 Security Group Tagging Imposition.

Additional Guidelines

- The ASA supports SXP Version 3. The ASA negotiates SXP versions with different SXP-capable network devices.
- The ASA does not support Change of Authorization (CoA) for the Cisco TrustSec (CTS) environment data.
- You can configure the ASA to refresh the security group table when the SXP reconcile timer expires and you can download the security group table on demand. When the security group table on the ASA is updated from the ISE, changes are reflected in the appropriate security policies.
- Cisco TrustSec supports the Smart Call Home feature in single context and multi-context mode, but not in the system context.
- The ASA can only be configured to interoperate in a single Cisco TrustSec domain.
- The ASA does not support static configuration of SGT-name mapping on the device.
- NAT is not supported in SXP messages.
- SXP conveys IP-SGT mapping to enforcement points in the network. If an access layer switch belongs to a different NAT domain than the enforcing point, the IP-SGT map that it uploads is invalid, and an IP-SGT mapping database lookup on the enforcement device does not yield valid results. As a result, the ASA cannot apply security group-aware security policy on the enforcement device.
- You can configure a default password for the ASA to use for SXP connections, or you can choose not to use a password; however, connection-specific passwords are not supported for SXP peers. The configured default SXP password should be consistent across the deployment network. If you configure a connection-specific password, connections may fail and a warning message appears. If you configure the connection with the default password, but it is not configured, the result is the same as when you have configured the connection with no password.
- The ASA can be configured as an SXP Speaker or Listener, or both. However, SXP connection loops can form when a device has bidirectional connections to a peer or is part of a unidirectionally connected chain of devices. (The ASA can learn IP-SGT mapping for resources from the access layer in the data center. The ASA might need to propagate these tags to downstream devices.) SXP connection loops can cause unexpected behavior of SXP message transport. In cases where the ASA is configured to be a Speaker and Listener, an SXP connection loop can occur, causing SXP data to be received by the peer that originally transmitted it.
- When changing the ASA local IP address, you must ensure that all SXP peers have updated their peer list. In addition, if SXP peers changes its IP addresses, you must ensure those changes are reflected on the ASA.
- Automatic PAC file provisioning is not supported. The ASA administrator must request the PAC file from the ISE administrative interface and import it into the ASA.
- PAC files have expiration dates. You must import the updated PAC file before the current PAC file expires; otherwise, the ASA cannot retrieve environment data updates. If the PAC file downloaded from the ISE expires on the ASA and it cannot download an updated security group table, the ASA continues to enforce security policies based on the last downloaded security group table until the ASA downloads an updated table.
- When a security group changes on the ISE (for example, it is renamed or deleted), the ASA does not change the status of any ASA security policies that contain an SGT or security group name associated

with the changed security group; however, the ASA generates a syslog message to indicate that those security policies changed.

- The multi-cast types are not supported in ISE 1.0.
- An SXP connection stays in the initializing state among two SXP peers interconnected by the ASA; as shown in the following example:

```
(SXP peer A) - - - - (ASA) - - - (SXP peer B)
```

Therefore, when configuring the ASA to integrate with Cisco TrustSec, you must enable the no-NAT, no-SEQ-RAND, and MD5-AUTHENTICATION TCP options on the ASA to configure SXP connections. Create a TCP state bypass policy for traffic destined to SXP port TCP 64999 among the SXP peers. Then apply the policy on the appropriate interfaces.

For example, the following set of commands shows how to configure the ASA for a TCP state bypass policy:

```
access-list SXP-MD5-ACL extended permit tcp host peerA host peerB eq 64999
access-list SXP-MD5-ACL extended permit tcp host peerB host peerA eq 64999

tcp-map SXP-MD5-OPTION-ALLOW
 tcp-options range 19 19 allow

class-map SXP-MD5-CLASSMAP
 match access-list SXP-MD5-ACL

policy-map type inspect dns preset_dns_map
 parameters
  message-length maximum 512
policy-map global_policy
class SXP-MD5-CLASSMAP
  set connection random-sequence-number disable
  set connection advanced-options SXP-MD5-OPTION-ALLOW
  set connection advanced-options tcp-state-bypass
service-policy global_policy global
```

Configure the ASA to Integrate with Cisco Trustsec

To configure the ASA to integrate with Cisco TrustSec, perform the following tasks.

Before you begin

Before configuring the ASA to integrate with Cisco TrustSec, you must complete the following tasks in ISE:

- Register the ASA with the ISE, on page 73
- Create a Security Group on the ISE, on page 74
- Generate the PAC File, on page 74

Procedure

Step 1 Configure the AAA Server for Cisco TrustSec Integration, on page 78

Step 2 Import a PAC File, on page 80

Step 3 Configure the Security Exchange Protocol, on page 81

This task enables and sets the default values for SXP.

Step 4 Add an SXP Connection Peer, on page 83

Step 5 Refresh Environment Data, on page 84

Do this as needed.

Step 6 Configure the Security Policy, on page 85

Step 7 Configure Layer 2 Security Group Tagging Imposition, on page 86

Configure the AAA Server for Cisco TrustSec Integration

This section describes how to integrate the AAA server for Cisco TrustSec. To configure the AAA server group to communicate with the ISE on the ASA, perform the following steps.

Before you begin

- The referenced server group must be configured to use the RADIUS protocol. If you add a non-RADIUS server group to the ASA, the configuration fails.
- If the ISE is also used for user authentication, obtain the shared secret that was entered on the ISE when you registered the ASA with the ISE. Contact your ISE administrator to obtain this information.

Procedure

Step 1 Create the AAA server group and configure the AAA server parameters for the ASA to communicate with the ISE server.

aaa-server *server-tag* **protocol radius**

Example:

```
ciscoasa(config)# aaa-server ISEserver protocol radius
```

The *server-tag* argument specifies the server group name.

Step 2 Exit from the aaa server group configuration mode.

exit

Example:

```
ciscoasa(config-aaa-server-group)# exit
```

Step 3 Configure a AAA server as part of a AAA server group and set host-specific connection data.

```
ciscoasa(config)# aaa-server server-tag(interface-name) host server-ip
```

Example:

```
ciscoasa(config)# aaa-server ISEserver (inside) host 192.0.2.1
```

The *interface-name* argument specifies the network interface where the ISE server resides. The parentheses are required in this parameter. The *server-tag* argument is the name of the AAA server group. The *server-ip* argument specifies the IP address of the ISE server.

Step 4 Specify the server secret value used to authenticate the ASA with the ISE server.

```
key key
```

Example:

```
ciscoasa(config-aaa-server-host)# key myexclusivekey
```

The *key* argument is an alphanumeric keyword up to 127 characters long.

If the ISE is also used for user authentication, enter the shared secret that was entered on the ISE when you registered the ASA with the ISE.

Step 5 Exit from the aaa server host configuration mode.

```
exit
```

Example:

```
ciscoasa(config-aaa-server-host)# exit
```

Step 6 Identify the AAA server group that is used by Cisco TrustSec for environment data retrieval.

```
cts server-group AAA-server-group-name
```

Example:

```
ciscoasa(config)# cts server-group ISEserver
```

The *AAA-server-group-name* argument is the name of the AAA server group that you specified in Step 1 in the *server-tag* argument.

Note
You may configure only one instance of the server group on the ASA for Cisco TrustSec.

The following example shows how to configure the ASA to communicate with the ISE server for Cisco TrustSec integration:

```
ciscoasa(config)#aaa-server ISEserver protocol radius
ciscoasa(config-aaa-server-group)# exit
ciscoasa(config)# aaa-server ISEserver (inside) host 192.0.2.1
ciscoasa(config-aaa-server-host)# key myexclusivemumblekey
ciscoasa(config-aaa-server-host)# exit
ciscoasa(config)# cts server-group ISEserver
```

Import a PAC File

This section describes how to import a PAC file.

Before you begin

- The ASA must be configured as a recognized Cisco TrustSec network device in the ISE before the ASA can generate a PAC file.
- Obtain the password used to encrypt the PAC file when generating it on the ISE. The ASA requires this password to import and decrypt the PAC file.
- When imported, the PAC file resides in NVRAM. When operating in HA mode, if you configure the failover and stateful links correctly, importing the PAC file into the active unit will result in replication to the secondary. Because the imported file resides in NVRAM, you must import the file again whenever the device reboots, for example, after a software upgrade.
- The device uses a single PAC file. If you import more than one, each imported PAC file replaces the previously imported file.
- The ASA requires access to the PAC file generated by the ISE. The ASA can import the PAC file from flash or from a remote server via TFTP, FTP, HTTP, HTTPS, or SMB. (The PAC file does not need to reside on the ASA flash before you can import it.)
- The server group has been configured for the ASA.

Procedure

Import a Cisco TrustSec PAC file.

cts import-pac *filepath* **password** *value*

Example:

```
ciscoasa(config)# cts import-pac disk0:/xyz.pac password IDFW-pac99
```

The *value* argument specifies the password used to encrypt the PAC file. The password is independent of the password that was configured on the ISE as part of the device credentials. The *filepath* argument is entered as one of the following options:

Single Mode

- **disk0**: Path and filename on disk0

- **disk1**: Path and filename on disk1
- **flash**: Path and filename on flash
- **ftp**: Path and filename on FTP
- **http**: Path and filename on HTTP
- **https**: Path and filename on HTTPS
- **smb**: Path and filename on SMB
- **tftp**: Path and filename on TFTP

Multi-mode

- **http**: Path and filename on HTTP
- **https**: Path and filename on HTTPS
- **smb**: Path and filename on SMB
- **tftp**: Path and filename on TFTP

The following example shows how to import a PAC file into the ASA:

```
ciscoasa(config)# cts import pac disk0:/pac123.pac password hideme
PAC file successfully imported
```

Configure the Security Exchange Protocol

You need to enable and configure the Security Exchange Protocol (SXP) to use Cisco Trustsec.

Before you begin

At least one interface must be in the UP/UP state. If you enable SXP with all interfaces down, the ASA does not display a message indicating that SXP is not working or it could not be enabled. If you check the configuration by entering the **show running-config** command, the command output displays the following message:

```
"WARNING: SXP configuration in process, please wait for a few moments and try again."
```

Procedure

Step 1 Enable SXP on the ASA. By default, SXP is disabled.

cts sxp enable

Example:

```
ciscoasa(config)# cts sxp enable
```

Step 2 (Optional; not recommended.) Configure the default source IP address for SXP connections.

cts sxp default source-ip *ipaddress*

Example:

```
ciscoasa(config)# cts sxp default source-ip 192.168.1.100
```

The *ipaddress* argument is an IPv4 or IPv6 address.

When you configure a default source IP address for SXP connections, you must specify the same address as the ASA outbound interface. If the source IP address does not match the address of the outbound interface, SXP connections fail.

When a source IP address for an SXP connection is not configured, the ASA performs a route/ARP lookup to determine the outbound interface for the SXP connection. We recommend that you do not configure a default source IP address for SXP connections and allow the ASA to perform a route/ARP lookup to determine the source IP address for an SXP connection.

Step 3 (Optional.) Configure the default password for TCP MD5 authentication with SXP peers. By default, SXP connections do not have a password.

cts sxp default password [**0** | **8**] *password*

Example:

```
ciscoasa(config)# cts sxp default password 8 IDFW-TrustSec-99
```

Configure a default password if and only if you configure the SXP connection peers to use the default password.

The length of the password depends on the decryption level, which defaults to 0 if you do not specify it:

- **0**—Unencrypted cleartext. The password can be up to 80 characters.
- **8**—Encrypted text. The password can be up to 162 characters.

Step 4 (Optional.) Specify the time interval between ASA attempts to set up new SXP connections between SXP peers.

cts sxp retry period *timervalue*

Example:

```
ciscoasa(config)# cts sxp retry period 60
```

The ASA continues to make connection attempts until a successful connection is made, waiting the retry interval before trying again after a failed attempt. You can specify a retry period from 0 to 64000 seconds. The default is 120 seconds. If you specify 0 seconds, the ASA does not try to connect to SXP peers.

We recommend that you configure the retry timer to a different value from its SXP peer devices.

Step 5 (Optional.) Specify the value of the reconciliation timer.

cts sxp reconciliation period *timervalue*

Example:

```
ciscoasa(config)# cts sxp reconciliation period 60
```

After an SXP peer terminates its SXP connection, the ASA starts a hold down timer. If an SXP peer connects while the hold down timer is running, the ASA starts the reconciliation timer; then, the ASA updates the SXP mapping database to learn the latest mappings.

When the reconciliation timer expires, the ASA scans the SXP mapping database to identify stale mapping entries (which were learned in a previous connection session). The ASA marks these connections as obsolete. When the reconciliation timer expires, the ASA removes the obsolete entries from the SXP mapping database.

You can specify a reconciliation period from 1 to 64000 seconds. The default is 120 seconds.

Step 6 (Optional.) Configure the delete-hold-down timer for the IP-SGT mappings learned from a peer after an SXP peer terminates its SXP connection.

cts sxp delete-hold-down period *timervalue*

The timer value specifies the number of seconds, 120-64000, that IP-SGT mappings learned from a torn-down SXP connection are held before being deleted.

Example:

```
ciscoasa(config)# cts sxp delete-hold-down period 240
```

Each SXP connection is associated with a delete hold down timer. This timer is triggered when an SXP connection on the listener side is torn down. The IP-SGT mappings learned from this SXP connection are not deleted immediately. Instead, they are held until the delete hold down timer expires. The mappings are deleted upon the expiry of this timer.

Step 7 (Optional.) Configure the depth of IPv4 subnet expansion when acting as a speaker to peers that use SXPv2 or lower.

cts sxp mapping network-map *maximum_hosts*

If a peer uses SXPv2 or lower, the peer cannot understand SGT to subnet bindings. The ASA can expand the IPv4 subnet bindings to individual host bindings (IPv6 bindings are not expanded). This command specifies the maximum number of host bindings that can be generated from a subnet binding.

You can specify the maximum number to be from 0 to 65535. The default is 0, which means that subnet bindings are not expanded to host bindings.

Add an SXP Connection Peer

To add an SXP connection peer, perform the following steps:

Procedure

Set up an SXP connection to an SXP peer.

cts sxp connection peer *peer_ip_address* [**source** *source_ip_address*] **password** {**default** | **none**} [**mode** {**local** | **peer**}] {**speaker** | **listener**}

Example:

```
ciscoasa(config)# cts sxp connection peer 192.168.1.100 password default mode peer speaker
```

SXP connections are set per IP address; a single device pair can service multiple SXP connections.

The *peer_ip_address* argument is the IPv4 or IPv6 address of the SXP peer. The peer IP address must be reachable from the ASA outgoing interface.

The *source_ip_address* argument is the local IPv4 or IPv6 address of the SXP connection. The source IP address must be the same as the ASA outbound interface or the connection fails.

We recommend that you do not configure a source IP address for an SXP connection and allow the ASA to perform a route/ARP lookup to determine the source IP address for the SXP connection.

Indicate whether or not to use the authentication key for the SXP connection:

- **default**—Use the default password configured for SXP connections.
- **none**—Do not use a password for the SXP connection.

Indicate the mode of the SXP connection:

- **local**—Use the local SXP device.
- **peer**—Use the peer SXP device.

Indicate whether the ASA functions as a Speaker or Listener for the SXP connection.

- **speaker**— The ASA can forward IP-SGT mapping to upstream devices.
- **listener**—The ASA can receive IP-SGT mapping from downstream devices.

The following example shows how to configure SXP peers on the ASA:

```
ciscoasa(config)# cts sxp connection peer 192.168.1.100 password default
mode peer speaker
ciscoasa(config)# cts sxp connection peer 192.168.1.101 password default
mode peer speaker
```

Refresh Environment Data

The ASA downloads environment data from the ISE, which includes the Security Group Tag (SGT) name table. The ASA automatically refreshes its environment data that is obtained from the ISE when you complete the following tasks on the ASA:

- Configure a AAA server to communicate with the ISE.
- Import a PAC file from the ISE.
- Identify the AAA server group that the ASA will use to retrieve Cisco TrustSec environment data.

Normally, you do not need to manually refresh the environment data from the ISE; however, security groups can change on the ISE. These changes are not reflected on the ASA until you refresh the data in the ASA security group table, so refresh the data on the ASA to make sure that any security group changes made on the ISE are reflected on the ASA.

Note We recommend that you schedule policy configuration changes on the ISE and the manual data refresh on the ASA during a maintenance window. Handling policy configuration changes in this way maximizes the chances of security group names getting resolved and security policies becoming active immediately on the ASA.

To refresh the environment data, perform the following steps:

Procedure

Refresh the environment data from the ISE and reset the reconcile timer to the configured default value.

```
cts refresh environment-data
```

Example:

```
ciscoasa(config)# cts refresh environment-data
```

Configure the Security Policy

You can incorporate Cisco TrustSec policy in many ASA features. Any feature that uses extended ACLs (unless listed in this chapter as unsupported) can take advantage of Cisco TrustSec. You can add security group arguments to extended ACLs, as well as traditional network-based parameters.

- To configure an extended ACL, see Add an Extended ACE for Security Group-Based Matching (Cisco TrustSec), on page 38.
- To configure security group object groups that can be used in the ACL, see Configure Security Group Object Groups, on page 21.

For example, an access rule permits or denies traffic on an interface using network information. With Cisco TrustSec, you can control access based on security group. For example, you could create an access rule for sample_securitygroup1 10.0.0.0 255.0.0.0, meaning the security group could have any IP address on subnet 10.0.0.0/8.

You can configure security policies based on combinations of security group names (servers, users, unmanaged devices, and so on), user-based attributes, and traditional IP-address-based objects (IP address, Active Directory object, and FQDN). Security group membership can extend beyond roles to include device and location attributes and is independent of user group membership.

The following example shows how to create an ACL that uses a locally defined security object group:

```
object-group security objgrp-it-admin
```

```
   security-group name it-admin-sg-name
   security-group tag 1
object-group security objgrp-hr-admin
   security-group name hr-admin-sg-name  // single sg_name
   group-object it-admin     // locally defined object-group as nested object
object-group security objgrp-hr-servers
security-group name hr-servers-sg-name
object-group security objgrp-hr-network
security-group tag 2
access-list hr-acl permit ip object-group-security objgrp-hr-admin any
object-group-security objgrp-hr-servers
```

The ACL configured in the previous example can be activated by configuring an access group or the Modular Policy Framework.

Additional examples:

```
!match src hr-admin-sg-name from any network to dst host 172.23.59.53
access-list idw-acl permit ip security-group name hr-admin-sg-name any host 172.23.59.53

!match src hr-admin-sg-name from host 10.1.1.1 to dst any
access-list idfw-acl permit ip security-group name hr-admin-sg-name host 10.1.1.1 any

!match src tag 22 from any network to dst hr-servers-sg-name any network
access-list idfw-acl permit ip security-group tag 22 any security-group
name hr-servers-sg-name any

!match src user mary from any host to dst hr-servers-sg-name any network
access-list idfw-acl permit ip user CSCO\mary any security-group
name hr-servers-sg-name any

!match src objgrp-hr-admin from any network to dst objgrp-hr-servers any network
access-list idfw-acl permit ip object-group-security objgrp-hr-admin any
object-group-security objgrp-hr-servers any

!match src user Jack from objgrp-hr-network and ip subnet 10.1.1.0/24
! to dst objgrp-hr-servers any network
access-list idfw-acl permit ip user CSCO\Jack object-group-security
objgrp-hr-network 10.1.1.0 255.255.255.0 object-group-security objgrp-hr-servers any

!match src user Tom from security-group mktg any google.com
object network net-google
fqdn google.com
access-list sgacl permit ip sec name mktg any object net-google

! If user Tom or object_group security objgrp-hr-admin needs to be matched,
! multiple ACEs can be defined as follows:
access-list idfw-acl2 permit ip user CSCO\Tom 10.1.1.0 255.255.255.0
    object-group-security objgrp-hr-servers any
access-list idfw-acl2 permit ip object-group-security objgrp-hr-admin
    10.1.1.0 255.255.255.0 object-group-security objgrp-hr-servers any
```

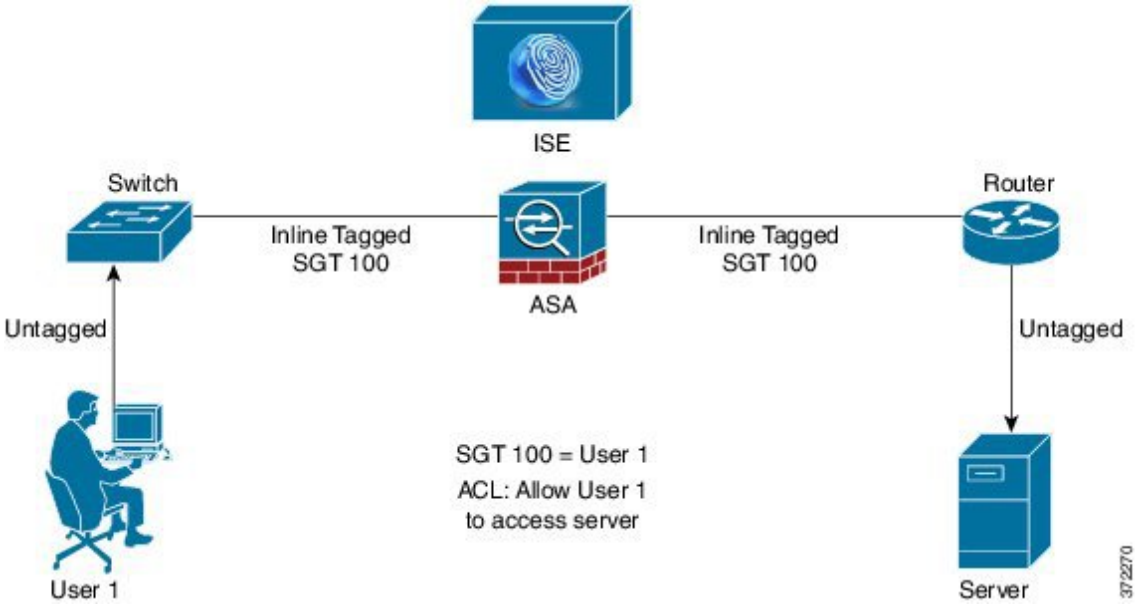
Configure Layer 2 Security Group Tagging Imposition

Cisco TrustSec identifies and authenticates each network user and resource and assigns a 16-bit number called a Security Group Tag (SGT). This identifier is in turn propagated between network hops, which allows any intermediary devices such as ASAs, switches, and routers to enforce polices based on this identity tag.

SGT plus Ethernet Tagging, also called Layer 2 SGT Imposition, enables the ASA to send and receive security group tags on Ethernet interfaces using Cisco proprietary Ethernet framing (EtherType 0x8909), which allows the insertion of source security group tags into plain-text Ethernet frames. The ASA inserts security group tags on the outgoing packet and processes security group tags on the incoming packet, based on a manual per-interface configuration. This feature allows inline hop-by-hop propagation of endpoint identity across network devices and provides seamless Layer 2 SGT Imposition between each hop.

The following figure shows a typical example of Layer 2 SGT Imposition.

Figure 5: Layer 2 SGT Imposition



Usage Scenarios

The following table describes the expected behavior for ingress traffic when configuring this feature.

Table 3: Ingress Traffic

Interface Configuration	Tagged Packet Received	Untagged Packet Received
No command is issued.	Packet is dropped.	SGT value is from the IP-SGT Manager.
The **cts manual** command is issued.	SGT value is from the IP-SGT Manager.	SGT value is from the IP-SGT Manager.
The **cts manual** command and the **policy static sgt** *sgt_number* command are both issued.	SGT value is from the **policy static sgt** *sgt_number* command.	SGT value is from the **policy static sgt** *sgt_number* command.
The **cts manual** command and the **policy static sgt** *sgt_number* **trusted** command are both issued.	SGT value is from the inline SGT in the packet.	SGT value is from the **policy static sgt** *sgt_number* command.

Note If there is no matched IP-SGT mapping from the IP-SGT Manager, then a reserved SGT value of "0x0" for "Unknown" is used.

The following table describes the expected behavior for egress traffic when configuring this feature.

Table 4: Egress Traffic

Interface Configuration	Tagged or Untagged Packet Sen
No command is issued.	Untagged
The **cts manual** command is issued.	Tagged
The **cts manual** command and the **propagate sgt** command are both issued.	Tagged
The **cts manual** command and the **no propagate sgt** command are both issued.	Untagged

The following table describes the expected behavior for to-the-box and from-the-box traffic when configuring this feature.

Table 5: To-the-box and From-the-box Traffic

Interface Configuration	Tagged or Untagged Packet Received
No command is issued on the ingress interface for to-the-box traffic.	Packet is dropped.
The **cts manual** command is issued on the ingress interface for to-the-box traffic.	Packet is accepted, but there is no policy enforcement propagation.
The **cts manual** command is not issued or the **cts manual** command and **no propagate sgt** command are both issued on the egress interface for from-the-box traffic.	Untagged packet is sent, but there is no policy enforce SGT number is from the IP-SGT Manager.
The **cts manual** command is issued or the **cts manual** command and the **propagate sgt** command are both issued on the egress interface for from-the-box traffic.	Tagged packet is sent. The SGT number is from the IP Manager.

Note If there is no matched IP-SGT mapping from the IP-SGT Manager, then a reserved SGT value of "0x0" for "Unknown" is used.

Configure a Security Group Tag on an Interface

To configure a security group tag on an interface, perform the following steps:

Procedure

Step 1 Specify an interface and enter interface configuration mode.

interface *id*

Example:

```
ciscoasa(config)# interface gigabitethernet 0/0
```

Step 2 Enable Layer 2 SGT Imposition and enter cts manual interface configuration mode.

cts manual

Example:

```
ciscoasa(config-if)# cts manual
```

Step 3 Enable propagation of a security group tag on an interface. Propagation is enabled by default.

propagate sgt

Example:

```
ciscoasa(config-if-cts-manual)# propagate sgt
```

Step 4 Apply a policy to a manually configured CTS link.

policy static sgt *sgt_number* [**trusted**]

Example:

```
ciscoasa(config-if-cts-manual)# policy static sgt 50 trusted
```

The **static** keyword specifies an SGT policy to incoming traffic on the link.

The **sgt** *sgt_number* keyword-argument pair specifies the SGT number to apply to incoming traffic from the peer. Valid values are from 2-65519.

The **trusted** keyword indicates that ingress traffic on the interface with the SGT specified in the command should not have its SGT overwritten. Untrusted is the default.

The following example enables an interface for Layer 2 SGT imposition and defines whether or not the interface is trusted:

```
ciscoasa(config)# interface gi0/0
ciscoasa(config-if)# cts manual
ciscoasa(config-if-cts-manual)# propagate sgt
ciscoasa(config-if-cts-manual)# policy static sgt 50 trusted
```

Configure IP-SGT Bindings Manually

To configure IP-SGT bindings manually, perform the following steps:

Procedure

Configure IP-SGT bindings manually.

cts role-based sgt-map {*IPv4_addr*[/*mask*] | *IPv6_addr*[/*prefix*]} **sgt** *sgt_value*

Example:

```
ciscoasa(config)# cts role-based sgt-map 10.2.1.2 sgt 50
```

You can specify an IPv4 or IPv6 host address. You can also specify network addresses by including a subnet mask or prefix value (for IPv6), such as 10.100.10.0/24. The *sgt_value* is the SGT number, from 2 to 65519.

Troubleshooting Tips

Use the **packet-tracer** command to determine why a particular session was allowed or denied, which SGT value is being used (from the SGT in the packet, from the IP-SGT manager, or from the **policy static sgt** command configured on the interface), and which security group-based security policies were applied.

The following example displays output from the **packet-tracer** command to show security group tag mapping to an IP address:

```
ciscoasa# packet-tracer input inside tcp inline-tag 100
security-group name alpha 30 security-group tag 31 300
Mapping security-group 30:alpha to IP address 10.1.1.2.
Mapping security-group 31:bravo to IP address 192.168.1.2.

Phase: 1
Type: ROUTE-LOOKUP
Subtype: input
Result: ALLOW
Config:
Additional Information:
in 192.168.1.0 255.255.255.0 outside....
----------------More---------------------
```

Use the **capture** *capture-name* **type inline-tag** *tag* command to capture only the Cisco CMD packets (EtherType 0x8909) with or without a specific SGT value.

The following example displays output from the **show capture** command for a specified SGT value:

```
ciscoasa# show capture my-inside-capture
1: 11:34:42.931012 INLINE-TAG 36 10.0.101.22 > 10.0.101.100: icmp: echo request
2: 11:34:42.931470 INLINE-TAG 48 10.0.101.100 > 10.0.101.22: icmp: echo reply
3: 11:34:43.932553 INLINE-TAG 36 10.0.101.22 > 10.0.101.100: icmp: echo request
4: 11.34.43.933164 INLINE-TAG 48 10.0.101.100 > 10.0.101.22: icmp: echo reply
```

Example for Cisco TrustSec

The following example shows how to configure the ASA to use Cisco TrustSec:

```
// Import an encrypted CTS PAC file
cts import-pac asa.pac password Cisco
// Configure ISE for environment data download
aaa-server cts-server-list protocol radius
aaa-server cts-server-list host 10.1.1.100 cisco123
cts server-group cts-server-list
// Configure SXP peers
cts sxp enable
cts sxp connection peer 192.168.1.100 password default mode peer speaker
//Configure security-group based policies
object-group security objgrp-it-admin
  security-group name it-admin-sg-name
  security-group tag 1
object-group security objgrp-hr-admin
security-group name hr-admin-sg-name
group-object it-admin
object-group security objgrp-hr-servers
security-group name hr-servers-sg-name
access-list hr-acl permit ip object-group-security objgrp-hr-admin any
object-group-security objgrp-hr-servers
//Configure security group tagging plus Ethernet tagging
   interface gi0/1
   cts manual
   propagate sgt
   policy static sgt 100 trusted
   cts role-based sgt-map 10.1.1.100 sgt 50
```

Secure Client VPN Support for Cisco TrustSec

ASA supports security group tagging of VPN sessions. You can assign a Security Group Tag (SGT) to a VPN session using an external AAA server, or by configuring a security group tag for a local user or for a VPN group policy. This tag can then be propagated through the Cisco TrustSec system over Layer 2 Ethernet. Security group tags are useful on group policies and for local users when the AAA server cannot provide an SGT.

Following is the typical process for assigning an SGT to a VPN user:

1. A user connects to a remote access VPN that uses a AAA server group containing ISE servers.
2. The ASA requests AAA information from ISE, which might include an SGT. The ASA also assigns an IP address for the user's tunneled traffic.
3. The ASA uses AAA information to authenticate the user and creates a tunnel.
4. The ASA uses the SGT from AAA information and the assigned IP address to add an SGT in the Layer 2 header.
5. Packets that include the SGT are passed to the next peer device in the Cisco TrustSec network.

If there is no SGT in the attributes from the AAA server to assign to a VPN user, then the ASA uses the SGT in the group policy. If there is no SGT in the group policy, then tag 0x0 is assigned.

Note You can also use ISE for policy enforcement using ISE Change of Authorization (CoA). For information on how to configure policy enforcement, see the VPN configuration guide.

Add an SGT to Remote Access VPN Group Policies and Local Users

To configure an SGT attribute on remote access VPN group policies, or on the VPN policy for a user defined in the LOCAL user database, perform the following steps.

There is no default SGT for group policies or local users.

Procedure

Step 1 To configure an SGT on a remote access VPN group policy:

a) Enter group-policy configuration mode:

group-policy *name*

Example:

```
ciscoasa(config)# group policy Grpolicy1
```

b) Configure the SGT for the group policy.

security-group-tag {**none** | **value** *sgt*}

If you set a tag using **value**, the tag can be from 2 to 65519. Specify **none** to set no SGT.

Example:

```
ciscoasa(config-group-policy# security-group-tag value 101
```

Step 2 To configure an SGT on for a user in the LOCAL database:

a) If necessary, create the user.

username *name* {**nopassword** | **password** *password* [**encrypted**]} [**privilege** *priv_level*]}

Example:

```
ciscoasa(config)# username newuser password changeme encrypted privilege 15
```

b) Enter username configuration mode.

username *name* **attributes**

Example:

```
asa3(config)# username newuser attributes
asa3(config-username)#
```

c) Configure the SGT for the user.

security-group-tag {**none** | **value** *sgt*}

If you set a tag using **value**, the tag can be from 2 to 65519. Specify **none** to set no SGT.

Example:

```
ciscoasa(config-username)# security-group-tag value 101
```

Monitoring Cisco TrustSec

See the following commands for monitoring Cisco TrustSec:

- **show running-config cts**
- **show running-config** [**all**] **cts role-based** [**sgt-map**]

  This command shows the user-defined IP-SGT binding table entries.
- **show cts sxp connections**

  This command shows the SXP connections on the ASA for a particular user context when multiple context mode is used.
- **show conn security-group**

  Shows data for all SXP connections.
- **show cts environment-data**

  Shows the Cisco TrustSec environment information contained in the security group table on the ASA.
- **show cts sgt-map**

  Shows the IP address-security group table manager entries in the control path.
- **show asp table cts sgt-map**

  This command shows the IP address-security group table mapping entries from the IP address-security group table mapping database maintained in the datapath.
- **show cts pac**

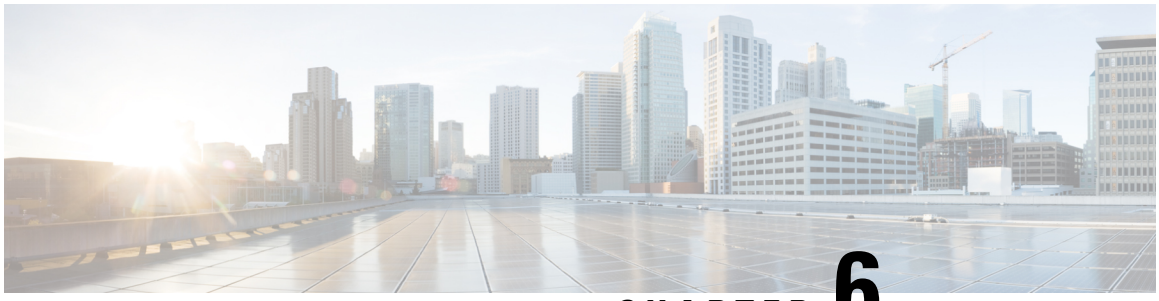
  Shows information about the PAC file imported into the ASA from the ISE and includes a warning message when the PAC file has expired or is within 30 days of expiration.

History for Cisco TrustSec

Table 6: History for Cisco TrustSec

Feature Name	Platform Releases	Description
Cisco TrustSec	9.0(1)	Cisco TrustSec provides access control that builds on an existing identity-aware infrastructure to ensure data confidentiality between network devices and integrate security access services on one platform. In the Cisco TrustSec feature, enforcement devices use a combination of user attributes and endpoint attributes to make role-based and identity-based access control decisions. In this release, the ASA integrates with Cisco TrustSec to provide security group-based policy enforcement. Access policies within the Cisco TrustSec domain are topology-independent, based on the roles of source and destination devices rather than on network IP addresses. The ASA can use Cisco TrustSec for other types of security group-based policies, such as application inspection; for example, you can configure a class map that includes an access policy based on a security group. We introduced or modified the following commands: **access-list extended**, **cts sxp enable**, **cts server-group**, **cts sxp default**, **cts sxp retry period**, **cts sxp reconciliation period**, **cts sxp connection peer**, **cts import-pac**, **cts refresh environment-data**, **object-group security**, **security-group**, **show running-config cts**, **show running-config object-group**, **clear configure cts**, **clear configure object-group**, **show cts pac**, **show cts environment-data**, **show cts environment-data sg-table**, **show cts sxp connections**, **show object-group**, **show configure security-group**, **clear cts environment-data**, **debug cts,** and **packet-tracer**.
Layer 2 Security Group Tag Imposition	9.3(1)	You can now use security group tagging combined with Ethernet tagging to enforce policies. SGT plus Ethernet Tagging, also called Layer 2 SGT Imposition, enables the ASA to send and receive security group tags on Ethernet interfaces using Cisco proprietary Ethernet framing (EtherType 0x8909), which allows the insertion of source security group tags into plain-text Ethernet frames. We introduced or modified the following commands: **cts manual**, **policy static sgt**, **propagate sgt**, **cts role-based sgt-map**, **show cts sgt-map**, **packet-tracer**, **capture**, **show capture**, **show asp drop**, **show asp table classify**, **show running-config all**, **clear configure all**, and **write memory**.
Cisco Trustsec support for Security Exchange Protocol (SXP) version 3.	9.6(1)	Cisco Trustsec on ASA now implements SXPv3, which enables SGT-to-subnet bindings, which are more efficient than host bindings. We introduced or modified the following commands: **cts sxp mapping network-map**, **cts role-based sgt-map**, **show cts sgt-map**, **show cts sxp sgt-map**, **show asp table cts sgt-map**.

Feature Name	Platform Releases	Description
Trustsec SXP connection configurable delete hold down timer	9.8(3)	The default SXP connection hold down timer is 120 seconds. You can now configure this timer, between 120 to 64000 seconds. New/Modified commands: **cts sxp delete-hold-down period**, **show cts sxp connection brief**, **show cts sxp connections**



CHAPTER 6

Cisco Umbrella

You can configure the device to redirect DNS requests to Cisco Umbrella, so that your FQDN policy defined in Cisco Umbrella can be applied to user connections. The following topics explain how to configure the Umbrella Connector to integrate the device with Cisco Umbrella.

- About Cisco Umbrella Connector, on page 97
- Licensing Requirements for Cisco Umbrella Connector, on page 99
- Guidelines and Limitations for Cisco Umbrella, on page 99
- Configure Cisco Umbrella Connector, on page 101
- Examples for the Umbrella Connector, on page 107
- Monitoring the Umbrella Connector , on page 109
- History for Cisco Umbrella Connector, on page 112

About Cisco Umbrella Connector

If you use Cisco Umbrella, you can configure the Cisco Umbrella Connector to redirect DNS queries to Cisco Umbrella. This allows Cisco Umbrella to identify requests to unpermitted or questionable domain names and apply your DNS-based security policy.

The Umbrella Connector is part of the system's DNS inspection. If your existing DNS inspection policy map decides to block or drop a request based on your DNS inspection settings, the request is not forwarded to Cisco Umbrella. Thus, you have two lines of protection: your local DNS inspection policy and your Cisco Umbrella cloud-based policy.

When redirecting DNS lookup requests to Cisco Umbrella, the Umbrella Connector adds an EDNS (Extension mechanisms for DNS) record. An EDNS record includes the device identifier information, organization ID, and client IP address. Your cloud-based policy can use those criteria to control access in addition to the reputation of the FQDN. You can also elect to encrypt the DNS request using DNSCrypt to ensure the privacy of usernames and internal IP addresses.

Cisco Umbrella Enterprise Security Policy

In your cloud-based Cisco Umbrella Enterprise Security policy, you can control access based on the reputation of the fully-qualified domain name (FQDN) in the DNS lookup request. Your Enterprise Security policy can enforce one of the following actions:

- Allow—If you have no block rules for an FQDN, and Cisco Umbrella determines that it belongs to a non-malicious site, then the site's actual IP address is returned. This is normal DNS lookup behavior.

- Proxy—If you have no block rules for an FQDN, and Cisco Umbrella determines that it belongs to a suspicious site, then the DNS reply returns the IP address of the Umbrella intelligent proxy. The proxy can then inspect the HTTP connection and apply URL filtering. You must ensure that intelligent proxy is enabled from the Cisco Umbrella dashboard (**Security Setting** > **Enable Intelligent Proxy**).
- Block—If you explicitly block an FQDN, or Cisco Umbrella determines that it belongs to a malicious site, then the DNS reply returns the IP address of the Umbrella cloud landing page for blocked connections.

Cisco Umbrella Registration

When you configure the Umbrella Connector on a device, it registers with Cisco Umbrella in the cloud. The registration process assigns a single device ID, which identifies one of the following:

- One standalone device in single context mode.
- One high availability pair in single context mode.
- One cluster in single context mode.
- One security context in a multiple-context standalone device.
- One security context of a high availability pair
- One security context of a cluster.

Once registered, the device details will appear on the Cisco Umbrella dashboard. You can then change which policy is attached to a device. During registration, either the policy you specify in the configuration is used, or the default policy is assigned. You can assign the same Umbrella policy to multiple devices. If you specify the policy, the device ID you receive differs from what you would get if you did not specify a policy.

Switching from Legacy API Token to API/Secret Keys

Umbrella changed the API registration mechanism from an API token to an API key and Secret key starting with ASA 9.23(1). If you are using legacy tokens, you can switch from the legacy API to the API/Secret key API by doing the following. See the related procedures for detailed steps.

Procedure

Step 1 Import the certificate for the new API. You need to install the ISRG Root X1 self-signed certificate.

Step 2 Disable Umbrella.

Step 3 Remove the existing token registration.

Step 4 Configure the new API and secret keys.

Step 5 Re-enable Umbrella.

Licensing Requirements for Cisco Umbrella Connector

To use the Cisco Umbrella Connector, you must have a 3DES license. If you are using Smart Licensing, your account must be enabled for export-controlled functionality.

The Cisco Umbrella portal has separate licensing requirements.

Guidelines and Limitations for Cisco Umbrella

Context Mode

- In multiple-context mode, you configure the Umbrella Connector in each context. Each context has a separate device ID, and is represented as a separate device in the Cisco Umbrella Connector dashboard. The device name is the hostname configured in the context, plus the hardware model, plus the context name. For example, CiscoASA-ASA5515-Context1.

Failover

- The active unit in the high availability pair registers the pair as a single unit with Cisco Umbrella. Both peers use the same device ID, which is formed from their serial numbers: *primary-serial-number_secondary-serial-number*. For multiple context mode, each pair of security contexts is considered a single unit. You must configure high availability, and the units must have successfully formed a high-availability group (even if the standby device is currently in a failed state), before enabling Cisco Umbrella, or the registration will fail.

Cluster

- The cluster control unit registers the cluster as a single unit with Cisco Umbrella. All peers use the same device ID. For multiple context mode, a security context in the cluster is considered a single unit across all peers.

Additional Guidelines

- Redirection to Cisco Umbrella is done for DNS requests in through traffic only. DNS requests that the system itself initiates are never redirected to Cisco Umbrella. For example, FQDN-based access control rules are never resolved based on Umbrella policy, nor are any FQDNs that are used in other commands or configuration settings.
- The Cisco Umbrella Connector works on any DNS request in through traffic. However, the block and proxy actions are effective only if the DNS response is then used for HTTP/HTTPS connections, because the IP address returned is for a web site. Any blocked or proxied addresses for non-HTTP/HTTPS connections will either fail or complete in a misleading fashion. For example, pinging a blocked FQDN would result in pinging the server that hosts the Cisco Umbrella cloud block page.

Note Cisco Umbrella does try to intelligently identify FQDNs that might be non-HTTP/HTTPS, and does not return the IP address to the intelligent proxy for those FQDNs for proxied domain names.

- The system sends DNS/UDP traffic only to Cisco Umbrella. If you enable DNS/TCP inspection, the system does not send any DNS/TCP requests to Cisco Umbrella. However, DNS/TCP requests do not increment the Umbrella bypass counter.
- If you enable DNScrypt for Umbrella inspection, the system uses UDP/443 for the encrypted session. You must include UDP/443 along with UDP/53 in the class map that applies DNS inspection for Cisco Umbrella for DNScrypt to work correctly. Both UDP/443 and UDP/53 are included in the default inspection class for DNS, but if you create a custom class, ensure that you define an ACL that includes both ports for the match class.
- DNScrypt uses IPv4 only for the certificate update handshake. However, DNSscrypt does encrypt both IPv4 and IPv6 traffic.
- There must be an IPv4 route to the Internet that can reach api.umbrella.com and api.opendns.com (registration uses IPv4 only). You also must have routes to the following DNS resolvers, and your access rules must allow DNS traffic to these hosts. These routes can go through either the data interfaces or the management interface; any valid route will work for both registration and DNS resolution. The default servers that the system uses are indicated; you can use the other servers by configuring the resolver in the Umbrella global settings.
  - 208.67.220.220 (system default for IPv4)
  - 208.67.222.222
  - 2620:119:53::53 (system default for IPv6)
  - 2620:119:35::35
- The system does not support the Umbrella FamilyShield service. If you configure the FamilyShield resolvers, you might get unexpected results.
- When evaluating whether to fail open, the system considers whether the Umbrella resolver is down, or if an intervening device drops the DNS request or response based on how long it has waited for the response after sending out the request. Other factors, such as no route to the Umbrella resolver, are not considered.
- To unregister a device, first delete the Umbrella configuration, then delete the device from the Cisco Umbrella dashboard.
- Any web requests that use IP addresses instead of FQDN will bypass Cisco Umbrella. In addition, if a roaming client obtains DNS resolution from a different WAN connection than the one that goes through an Umbrella-enabled device, connections that use those resolutions bypass Cisco Umbrella.
- If a user has an HTTP proxy, then the proxy might be doing DNS resolution, and the resolutions will not go through Cisco Umbrella.
- NAT DNS46 and DNS64 are not supported. You cannot translate DNS requests between IPv4 and IPv6 addressing.

- The EDNS record will include both the IPv4 and IPv6 host addresses.
- If the client uses DNS over HTTPS, then the cloud security service will not inspect DNS and HTTP/HTTPS traffic.

Configure Cisco Umbrella Connector

You can configure the device to interact with Cisco Umbrella in the cloud. The system redirects DNS lookup requests to Cisco Umbrella, which then applies your cloud-based Enterprise Security fully-qualified domain name (FQDN) policy. For malicious or suspicious traffic, users can be blocked from a site, or redirected to an intelligent proxy that can perform URL filtering based on your cloud-based policy.

The following procedure explains the end-to-end process for configuring the Cisco Umbrella Connector.

Before you begin

In multiple-context mode, perform this procedure in each security context that should use Cisco Umbrella.

Procedure

Step 1 Establish an account on Cisco Umbrella, https://umbrella.cisco.com.

Step 2 Install the CA Certificate from the Cisco Umbrella Registration Server, on page 102.

The device registration uses HTTPS, which requires that you install the root certificate.

Step 3 If it is not already enabled, configure DNS servers and enable DNS lookup on the interfaces.

You can use your own servers, or configure the Cisco Umbrella servers. DNS inspection automatically redirects to the Cisco Umbrella resolvers even if you configure different servers.

- 208.67.220.220
- 208.67.222.222
- 2620:119:53::53
- 2620:119:35::35

Example:

```
ciscoasa(config)# dns domain-lookup outside
ciscoasa(config)# dns domain-lookup inside
ciscoasa(config)# dns name-server 208.67.220.220
```

Step 4 Configure the Umbrella Connector Global Settings, on page 103.

Step 5 Enable Umbrella in the DNS Inspection Policy Map, on page 105.

Step 6 Verify the Umbrella Registration, on page 106.

Install the CA Certificate from the Cisco Umbrella Registration Server

You must import the root certificate to establish the HTTPS connection with the Cisco Umbrella registration server. The system uses the HTTPS connection when registering the device. In Cisco Umbrella choose **Deployments > Configuration > Root Certificate** and download the certificate.

When using the API and secret keys to register to Umbrella, you need to install the the ISRG Root X1 self-signed certificate pem file. You can obtain this certificate from https://letsencrypt.org/certs/isrgrootx1.pem.

Before you begin

When Umbrella updates its certificate, you need to download the new certificate. The root certificate might also change. Ensure that you have the correct root certificate uploaded.

When you update the certificate, you must disable Umbrella, then enable it again, so the system picks up the new certificate and registers correctly with Umbrella.

Procedure

Step 1 Create a trustpoint for the Cisco Umbrella registration server.

crypto ca trustpoint *name*

You can use any name you want for the trustpoint (up to 128 characters), such as ctx1 or umbrella_server.

Example:

```
ciscoasa(config)# crypto ca trustpoint ctx1
ciscoasa(config-ca-trustpoint)#
```

Step 2 Indicate that you want to manually enroll by pasting the certificate.

enrollment terminal

Example:

```
ciscoasa(config-ca-trustpoint)# enrollment terminal
ciscoasa(config-ca-trustpoint)#
```

Step 3 Import the certificate.

crypto ca authenticate *name*

Enter the name of the trustpoint you created for this certificate. Follow the prompts and paste the base-64 encoded certificate. Do not include the BEGIN CERTIFICATE and END CERTIFICATE lines.

```
ciscoasa(config-ca-trustpoint)# crypto ca authenticate ctx1
Enter the base 64 encoded CA certificate.
End with the word "quit" on a line by itself
```

Configure the Umbrella Connector Global Settings

The Umbrella global settings primarily define the API and Secret keys, or the legacy API token, that is needed to register the device with Cisco Umbrella.

The global settings are not sufficient to enable Umbrella. You must also enable Umbrella in your DNS inspection policy map, as described in Enable Umbrella in the DNS Inspection Policy Map, on page 105.

Before you begin

- Log into the Umbrella dashboard and generate an API key and a Secret key. Generate keys from **Admin** > **API Keys** in Umbrella. When generating the key, you must ensure that you copy and save the API and secret keys, as you cannot retrieve the secret later. The scope of the key must be read/write.
- (Legacy, not recommended.) Log into the Cisco Umbrella Network Devices Dashboard (https://login.umbrella.com/) and obtain a legacy network device API token for your organization. A token will be a hexadecimal string, for example, AABBA59A0BDE1485C912AFE. Generate a Legacy Network Devices API key from the Umbrella dashboard.
- Install the certificate for the Cisco Umbrella registration server.

Procedure

Step 1 Enter Umbrella configuration mode.

umbrella-global

Example:

```
ciscoasa(config)# umbrella-global
ciscoasa(config-umbrella)#
```

Step 2 Configure the API key and secret needed to register with Cisco Umbrella.

token-request-credential api-key *key_value* **secret** *secret_value*

Note that keys expire, and you need to disable umbrella, add new key/secret, and reenable umbrella when applying a new key.

Example:

```
ciscoasa(config)# umbrella-global
ciscoasa(config-umbrella)# token-request-credential
api-key f817feb474d94c56b3448bcbd08edc11 secret 4afe58df5c454161a10a172145cb1456
```

Step 3 (Legacy, not recommended.) Configure the API token needed to register with Cisco Umbrella.

You can configure the token only if you do not configure the API key and secret.

token *api_token*

Example:

```
ciscoasa(config)# umbrella-global
```

```
ciscoasa(config-umbrella)# token AABBA59A0BDE1485C912AFE
Please make sure all the Umbrella Connector prerequisites are satisfied:
1. DNS server is configured to resolve api.opendns.com
2. Route to api.opendns.com is configured
3. Root certificate of Umbrella registration is installed
4. Unit has a 3DES license
```

Step 4 (Optional.) If you intend to enable DNScrypt in the DNS inspection policy map, you can optionally configure the DNScrypt provider public key for certificate verification. If you do not configure the key, the default currently distributed public key is used for validation.

public-key *hex_key*

The key is a 32-byte hexadecimal value. Enter the hex value in ASCII with a colon separator for every 2 bytes. The key is 79 bytes long. Obtain this key from Cisco Umbrella.

The default key is:
B735:1140:206F:225D:3E2B:D822:D7FD:691E:A1C3:3CC8:D666:8D0C:BE04:BFAB:CA43:FB79

To revert to using the default public key, enter **no public-key**. You can either omit the key you configured, or include it on the **no** version of the command.

Example:

```
ciscoasa(config-umbrella)# public-key
B735:1140:206F:225D:3E2B:D822:D7FD:691E:A1C3:3CC8:D666:8D0C:BE04:BFAB:CA43:FB79
```

Step 5 (Optional.) Configure the idle timeout after which a connection from a client to the Umbrella server will be removed if there is no response from the server.

timeout edns *hh:mm:ss*

The timeout is in hours:minutes:seconds format, and can be from 0:0:0 to 1193:0:0. The default is 0:02:00 (2 minutes).

Example:

```
ciscoasa(config-umbrella)# timeout edns 00:01:00
```

Step 6 (Optional.) Configure the local domain names for which Umbrella should be bypassed.

You can identify local domains for which DNS requests should bypass Cisco Umbrella and instead go directly to the configured DNS servers. For example, you can have your internal DNS server resolve all names for the organization's domain name on the assumption that all internal connections are allowed.

You can enter your local domain name directly. Optionally, you can create the regular expressions that define the name, then create a regular expression class map and specify it on the following command:

local-domain-bypass {*regular_expression* | **regex class** *regex_classmap*}

Example:

```
ciscoasa(config)# umbrella-global
ciscoasa(config-umbrella)# local-domain-bypass example.com
```

Step 7 (Optional.) Configure the addresses of the non-default Cisco Umbrella DNS servers, which resolve DNS requests, that you want to use.

resolver {**ipv4** | **ipv6**} *ip_address*

You can enter the command separately to define the IPv4 and IPv6 addresses of non-default Umbrella resolvers.

Example:

```
ciscoasa(config-umbrella)# resolver ipv4 208.67.222.222
ciscoasa(config-umbrella)# resolver ipv6 2620:119:35::35
```

Step 8 Select the method to use when registering with Umbrella

- **Token** (Legacy, not recommended.)—Enter the API token in the **Token** field.
- **Token-Request-Credential**—Configure the following:
  - **API-Key**—Enter the API key you generated from Umbrella.
  - **Secret-Key**—Enter the Secret key you were provided for the API key.

Enable Umbrella in the DNS Inspection Policy Map

Configuring the global Umbrella settings is not enough to register the device and enable DNS lookup redirection. You must add Umbrella as part of your active DNS inspection.

You can enable Umbrella globally by adding it to the preset_dns_map DNS inspection policy map.

However, if you have customized DNS inspection and applied different inspection policy maps to different traffic classes, you must enable Umbrella on each class where you want the service.

The following procedure explains how to implement Umbrella globally. If you have customized DNS policy maps, please see Configure DNS Inspection Policy Map, on page 269.

Procedure

Step 1 Edit the preset_dns_map inspection policy map and enter parameter configuration mode.

```
ciscoasa(config)# policy-map type inspect dns preset_dns_map
ciscoasa(config-pmap)# parameters
ciscoasa(config-pmap-p)#
```

Step 2 Enable Umbrella and optionally specify the name of the Cisco Umbrella policy to apply to the device.

umbrella [**tag** *umbrella_policy*] [**fail-open**]

The tag is the name of a policy as defined in Cisco Umbrella. During registration, Cisco Umbrella will assign the policy to the device (if the policy name exists). If you do not specify a policy, the default policy is applied.

Include the **fail-open** keyword if you want DNS resolution to work if the Umbrella DNS server is unavailable. When failing open, if the Cisco Umbrella DNS server is unavailable, Umbrella disables itself on this policy map and allows DNS requests to go to the other DNS servers configured on the system, if any. When the

Umbrella DNS servers are available again, the policy map resumes using them. If you do not include this option, DNS requests continue to go to the unreachable Umbrella resolver, so they will not get a response.

Example:

```
ciscoasa(config-pmap-p)# umbrella fail-open
```

Step 3 (Optional.) Enable DNScrypt to encrypt connections between the device and Cisco Umbrella.

dnscrypt

Enabling DNScrypt starts the key-exchange thread with the Umbrella resolver. The key-exchange thread performs the handshake with the resolver every hour and updates the device with a new secret key. Because DNScrypt uses UDP/443, you must ensure that the class map used for DNS inspection includes that port. Note that the default inspection class already includes UDP/443 for DNS inspection.

Example:

```
ciscoasa(config-pmap-p)# dnscrypt
```

Example

```
ciscoasa(config)# policy-map type inspect dns preset_dns_map
ciscoasa(config-pmap)# parameters
ciscoasa(config-pmap-p)# umbrella fail-open
ciscoasa(config-pmap-p)# dnscrypt
```

Verify the Umbrella Registration

After you configure the global Umbrella settings and enable Umbrella in DNS inspection, the device should contact Cisco Umbrella and register. You can check for successful registration by checking whether Cisco Umbrella provided a device ID.

First, check the service policy statistics, and look for the Umbrella Registration line. This should indicate the policy applied by Cisco Umbrella (the tag), the HTTP status of the connection , and the device ID.

The status should be 200 Success. Error codes indicate the following: 401 indicates that the API token was incorrect, 403 indicates that the scope of the key was not read/write (blocking access), 405 indicates that the API key has expired, and 409 indicates that the device already exists in Cisco Umbrella.

If the status is UNKNOWN, check for syslog messages. Message 339011: "Umbrella API token request received no responses," indicates that there is a routing problem for getting to api.umbrella.com, or you did not upload the required certificate.

Note that the Umbrella Resolver lines should not indicate that the resolvers are unresponsive. If they are, verify that you opened DNS communication to these IP addresses in your access control policy. This might be a temporary situation, or it might indicate a routing problem.

```
asa(config)# show service-policy inspect dns
Interface inside:
```

```
  Service-policy: global_policy
    Class-map: inspection_default
     Inspect: dns preset_dns_map, packet 0, lock fail 0, drop 0, reset-drop 0, 5-min-pkt-rate
0 pkts/sec, v6-fail-close 0 sctp-drop-override 0
       message-length maximum client auto, drop 0
       message-length maximum 512, drop 0
       dns-guard, count 0
       protocol-enforcement, drop 0
       nat-rewrite, count 0
      umbrella registration: mode: fail-open tag: default, status: 200 success, device-id:
010a13b8fbdfc9aa
         Umbrella ipv4 resolver: 208.67.220.220
         Umbrella ipv6 resolver: 2620:119:53::53
      Umbrella: bypass 0, req inject 0 - sent 0, res recv 0 - inject 0 local-domain-bypass
10
       DNScrypt egress: rcvd 402, encrypt 402, bypass 0, inject 402
       DNScrypt ingress: rcvd 804, decrypt 402, bypass 402, inject 402
       DNScrypt: Certificate Update: completion 10, failure 1
```

You can also verify the running configuration (filter on policy-map). The umbrella command in the policy map updates to show the device ID. You cannot directly configure the device ID when you enable this command. The following example edits the output to show the relevant information.

```
ciscoasa(config)# show running-config policy-map
!
policy-map type inspect dns preset_dns_map
parameters
  message-length maximum client auto
  message-length maximum 512
  dnscrypt
  umbrella device-id 010a3e5760fdd6d3
  no tcp-inspection
policy-map global_policy
class inspection_default
  inspect dns preset_dns_map
```

Examples for the Umbrella Connector

The following topics provide examples of configuring the Umbrella Connector.

Example: Enabling Umbrella on the Global DNS Inspection Policy

The following example shows how to enable Umbrella globally. The configuration enables DNScrypt using the default public key. It assigns the default Cisco Umbrella Enterprise Security policy. The key and secret are examples only; you must generate your own valid key/secret pair from Umbrella.

This example assumes that you have uploaded the appropriate certificate needed for the Umbrella registration. Ensure that you download the latest root certificate used by the Umbrella site, as the correct certificate changes over time.

```
ciscoasa(config)# dns domain-lookup outside
ciscoasa(config)# dns domain-lookup inside
ciscoasa(config)# dns name-server 208.67.220.220

ciscoasa(config)# umbrella-global
```

```
ciscoasa(config-umbrella)# token-request-credential
api-key f817feb474d94c56b3448bcbd08edc11 secret 4afe58df5c454161a10a172145cb1456

ciscoasa(config)# policy-map type inspect dns preset_dns_map
ciscoasa(config-pmap)# parameters
ciscoasa(config-pmap-p)# umbrella
ciscoasa(config-pmap-p)# dnscrypt
```

Example: Enabling Umbrella on an Interface with a Custom Inspection Policy

The following example shows how to enable Umbrella for a specific traffic class. Umbrella is enabled on the inside interface only for DNS/UDP traffic. Because we are enabling DNScrypt, UDP/443 must be included in the traffic class. An Enterprise Security policy named mypolicy (defined in Cisco Umbrella) is applied. The key and secret are examples only; you must generate your own valid key/secret pair from Umbrella.

This example assumes that you have uploaded the appropriate certificate needed for the Umbrella registration. Ensure that you download the latest root certificate used by the Umbrella site, as the correct certificate changes over time.

```
ciscoasa(config)# dns domain-lookup outside
ciscoasa(config)# dns domain-lookup inside
ciscoasa(config)# dns name-server 208.67.220.220

ciscoasa(config)# umbrella-global
ciscoasa(config-umbrella)# token-request-credential
api-key f817feb474d94c56b3448bcbd08edc11 secret 4afe58df5c454161a10a172145cb1456


ciscoasa(config)# policy-map type inspect dns umbrella-policy
ciscoasa(config-pmap)# parameters
ciscoasa(config-pmap-p)# umbrella tag mypolicy
ciscoasa(config-pmap-p)# dnscrypt

ciscoasa(config)# object-group service umbrella-service-object
ciscoasa(config-service-object-group)# service-object udp destination eq domain
ciscoasa(config-service-object-group)# service-object udp destination eq 443

ciscoasa(config)# access-list umbrella-acl extended permit
object-group umbrella-service-object any any

ciscoasa(config)# class-map dns-umbrella
ciscoasa(config-cmap)# match access-list umbrella-acl

ciscoasa(config)# policy-map inside-policy
ciscoasa(config-pmap)# class dns-umbrella
ciscoasa(config-pmap-c)# inspect dns umbrella-policy

ciscoasa(config)# service-policy inside-policy interface inside
```

Example: Excluding Certain Hosts or Networks from Umbrella Globally

If you need to exclude certain hosts or networks from using Umbrella, and the choice is global rather than interface-based, you can remove the global DNS inspection, and create separate classes for excluding or including Umbrella inspection.

The following example modifies the global inspection policy to exclude the 192.168.1.0/24 network from using Umbrella.

Before you begin

This example assumes that you have already enabled DNS and configured the Umbrella global settings.

Procedure

Step 1 Remove the global default DNS inspection.

```
ciscoasa(config)# policy-map global_policy
ciscoasa(config-pmap)# class inspection_default
ciscoasa(config-pmap-c)# no inspect dns
```

Step 2 Create a DNS policy map that enables Umbrella.

In this example, the policy map is named umbrella-policy.

```
ciscoasa(config)# policy-map type inspect dns umbrella-policy
ciscoasa(config-pmap)# parameters
ciscoasa(config-pmap-p)# umbrella tag mypolicy
```

Step 3 Create a traffic class for the excluded traffic.

The following example uses an ACL to identify UDP/53 traffic from the 192.168.1.0/24 network.

```
ciscoasa(config)# access-list Umb_Exclude permit udp 192.168.1.0 255.255.255.0 any eq 53
ciscoasa(config)# class-map Umbrella_Exclude
ciscoasa(config-cmap)# match access-list Umb_Exclude
```

Step 4 Create a traffic class for hosts that should use Umbrella.

The following example matches UDP/53 traffic from any source.

```
ciscoasa(config)# class-map Umbrella_Include
ciscoasa(config-cmap)# match port udp eq 53
```

Step 5 Update the global inspection policy to enable DNS inspection for the traffic classes using the appropriate DNS policy map.

```
ciscoasa(config)# policy-map global_policy
ciscoasa(config-pmap)#  class Umbrella_Exclude
ciscoasa(config-pmap-c)# inspect dns
ciscoasa(config-pmap)#  class Umbrella_Include
ciscoasa(config-pmap-c)# inspect dns umbrella-policy
```

Monitoring the Umbrella Connector

The following topics explain how to monitor the Umbrella Connector.

Monitoring the Umbrella Service Policy Statistics

You can view both summarized and detailed statistics for DNS inspection with Umbrella enabled.

show service-policy inspect dns [**detail**]

Without the **detail** keyword, you see all the basic DNS inspection counters plus Umbrella configuration information. The status field provides the HTTP status code for the system's attempt to register with Cisco Umbrella.

The Resolver lines indicate which Umbrella servers are being used. These lines will say whether the server is **unresponsive**, or if the system is currently **probing** the server to determine if it has become available. If the mode is fail-open, the system allows DNS requests to go to other DNS servers (if configured); otherwise, DNS requests will not get a response so long as the Umbrella servers are unreponsive.

```
asa(config)# show service-policy inspect dns
Interface inside:
  Service-policy: global_policy
    Class-map: inspection_default
     Inspect: dns preset_dns_map, packet 0, lock fail 0, drop 0, reset-drop 0, 5-min-pkt-rate
 0 pkts/sec, v6-fail-close 0 sctp-drop-override 0
        message-length maximum client auto, drop 0
        message-length maximum 512, drop 0
        dns-guard, count 0
        protocol-enforcement, drop 0
        nat-rewrite, count 0
       umbrella registration: mode: fail-open tag: default, status: 200 success, device-id:
 010a13b8fbdfc9aa
          Umbrella ipv4 resolver: 208.67.220.220
          Umbrella ipv6 resolver: 2620:119:53::53
       Umbrella: bypass 0, req inject 0 - sent 0, res recv 0 - inject 0 local-domain-bypass
 10
        DNScrypt egress: rcvd 402, encrypt 402, bypass 0, inject 402
        DNScrypt ingress: rcvd 804, decrypt 402, bypass 402, inject 402
        DNScrypt: Certificate Update: completion 10, failure 1
```

The detailed output shows DNScrypt statistics and the keys used.

```
asa(config)# show service-policy inspect dns detail
Global policy:
  Service-policy: global_policy
    Class-map: inspection_default
    Class-map: dnscrypt30000
      Inspect: dns dns_umbrella, packet 12, lock fail 0, drop 0, reset-drop 0,
               5-min-pkt-rate 0 pkts/sec, v6-fail-close 0 sctp-drop-override 0
        message-length maximum client auto, drop 0
        message-length maximum 1500, drop 0
        dns-guard, count 3
        protocol-enforcement, drop 0
        nat-rewrite, count 0
       Umbrella registration: mode: fail-open tag: default, status: 200 SUCCESS, device-id:
 010af97abf89abc3, retry 0
          Umbrella ipv4 resolver: 208.67.220.220
          Umbrella ipv6 resolver: 2620:119:53::53
       Umbrella: bypass 0, req inject 6 - sent 6, res recv 6 - inject 6 local-domain-bypass
 10
          Umbrella app-id fail, count 0
          Umbrella flow alloc fail, count 0
          Umbrella block alloc fail, count 0
          Umbrella client flow expired, count 0
```

```
    Umbrella server flow expired, count 0
    Umbrella request drop, count 0
    Umbrella response drop, count 0
  DNScrypt egress: rcvd 6, encrypt 6, bypass 0, inject 6
  DNScrypt ingress: rcvd 18, decrypt 6, bypass 12, inject 6
    DNScrypt length error, count 0
    DNScrypt add padding error, count 0
    DNScrypt encryption error, count 0
    DNScrypt magic_mismatch error, count 0
    DNScrypt disabled, count 0
    DNScrypt flow error, count 0
    DNScrypt nonce error, count 0
  DNScrypt: Certificate Update: completion 1, failure 1
    DNScrypt Receive internal drop count 0
    DNScrypt Receive on wrong channel drop count 0
    DNScrypt Receive cannot queue drop count 0
    DNScrypt No memory to create channel count 0
    DNScrypt Send no output interface count 1
    DNScrypt Send open channel failed count 0
    DNScrypt Send no handle count 0
    DNScrypt Send dupb failure count 0
    DNScrypt Create cert update no memory count 0
    DNScrypt Store cert no memory count 0
    DNScrypt Certificate invalid length count 0
    DNScrypt Certificate invalid magic count 0
    DNScrypt Certificate invalid major version count 0
    DNScrypt Certificate invalid minor version count 0
    DNScrypt Certificate invalid signature count 0
    Last Successful: 01:42:29 UTC May 2 2018, Last Failed: None
    Magic DNSC, Major Version 0x0001, Minor Version 0x0000,
    Query Magic 0x714e7a696d657555, Serial Number 1517943461,
    Start Time 1517943461 (18:57:41 UTC Feb 6 2018)
    End Time 1549479461 (18:57:41 UTC Feb 6 2019)
    Server Public Key
240B:11B7:AD02:FAC0:6285:1E88:6EAA:44E7:AE5B:AD2F:921F:9577:514D:E226:D552:6836
    Client Secret Key Hash
48DD:E6D3:C058:D063:1098:C6B4:BA6F:D8A7:F0F8:0754:40B0:AFB3:CB31:2B22:A7A4:9CEE
    Client Public key
6CB9:FA4B:4273:E10A:8A67:BA66:76A3:BFF5:2FB9:5004:CD3B:B3F2:86C1:A7EC:A0B6:1A58
    NM key Hash
9182:9F42:6C01:003C:9939:7741:1734:D199:22DF:511E:E8C9:206B:D0A3:8181:CE57:8020
```

Monitoring Umbrella Syslog Messages

You can monitor the following Umbrella-related syslog messages:

- %ASA-3-339001: DNSCRYPT certificate update failed for *number* tries.

  Check that there is a route to the Umbrella server and that the egress interface is up and functioning correctly. Also check that the public key configured for DNScrypt is correct. You might need to obtain a new key from Cisco Umbrella.

- %ASA-3-339002: Umbrella device registration failed with error code *error_code*.

  The error codes have the following meanings:

  - 400—There is a problem with the request format or content. The token is probably too short or corrupted. Verify that the token matches the one on the Umbrella Dashboard.
  - 401—The API token is not authorized. Try reconfiguring the token. If you refreshed the token on the Umbrella Dashboard, then you must ensure that you use the new token.

  - 409—The device ID conflicts with another organization. Please check with the Umbrella Administrator to see what the issue might be.
  - 500—There is an internal server error. Check with the Umbrella Administrator to see what the issue might be.

- %ASA-6-339003: Umbrella device registration was successful.
- %ASA-3-339004: Umbrella device registration failed due to missing token.

  You must obtain an API token from Cisco Umbrella and configure it in the global Umbrella settings.
- %ASA-3-339005: Umbrella device registration failed after *number* retries.

  Check the syslog 339002 messages to identify the errors that you need to fix.
- %ASA-3-339006: Umbrella resolver *IP_address* is reachable, resuming Umbrella redirect.

  This message indicates that the system is functioning normally again. No action is needed.
- %ASA-3-339007: Umbrella resolver *IP_address* is unresponsive and fail-close mode used, starting probe to resolver.

  Because you are using fail-close mode, users will not get responses to their DNS requests until the Umbrella DNS server comes back online. If the problem persists, verify that there is a route from the system to the Umbrella servers, and that you allow DNS traffic to the servers in your access control policy.
- Messages related to API key/secret registration:
  - %ASA-6-339010: Umbrella API token request was successful
  - %ASA-3-339011: Umbrella API token request received no responses
  - %ASA-3-339012: Umbrella API token request failed with error code %d
  - %ASA-3-339013: Umbrella API token request failed in response processing
  - %ASA-3-339014: Umbrella API token request failed after %d retries; abort registration

History for Cisco Umbrella Connector

Feature Name	Platform Releases	Description
Cisco Umbrella support.	9.10(1)	You can configure the device to redirect DNS requests to Cisco Umbrella, so that your Enterprise Security policy defined in Cisco Umbrella can be applied to user connections. You can allow or block connections based on FQDN, or for suspicious FQDNs, you can redirect the user to the Cisco Umbrella intelligent proxy, which can perform URL filtering. The Umbrella configuration is part of the DNS inspection policy. We added or modified the following commands: **umbrella**, **umbrella-global**, **token**, **public-key**, **timeout edns**, **dnscrypt**, **show service-policy inspect dns detail**.

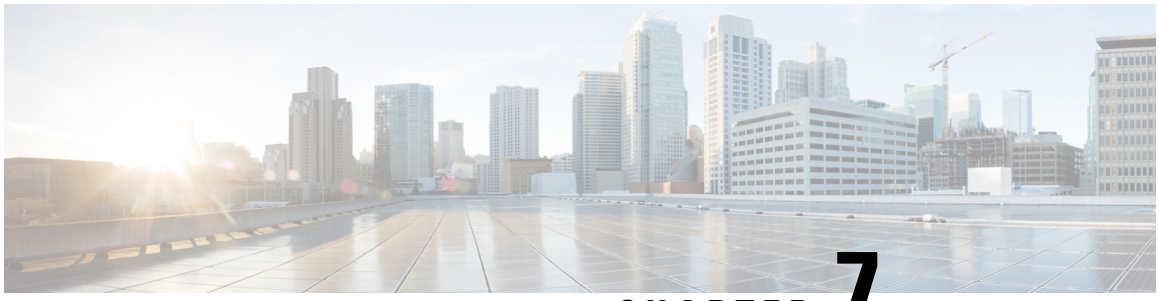
Feature Name	Platform Releases	Description
Cisco Umbrella Enhancements.	9.12(1)	You can now identify local domain names that should bypass Cisco Umbrella. DNS requests for these domains go directly to the DNS servers without Umbrella processing. You can also identify which Umbrella servers to use for resolving DNS requests. Finally, you can define the Umbrella inspection policy to fail open, so that DNS requests are not blocked if the Umbrella server is unavailable. We added or changed the following commands: **local-domain-bypass**, **resolver**, **umbrella fail-open**.
New Umbrella API.	9.23(1)	You can now configure Umbrella using the Umbrella Open API, which uses an API key with a Secret key. We added the following command: **token-request-credential**



PART II

Firewall Services for Virtual Environments

- Attribute-Based Access Control, on page 117



CHAPTER 7

Attribute-Based Access Control

Attributes are customized network objects for use in your configuration. You can define and use them in ASA configurations to filter traffic associated with one or more virtual machines in an VMware ESXi environment managed by VMware vCenter. Attributes allow you to define access control lists (ACLs) to assign policies to traffic from groups of virtual machines sharing one or more attributes. You assign attributes to virtual machines within the ESXi environment and configure an attribute agent, which connects to vCenter or a single ESXi host using HTTPS. The agent then requests and retrieves one or more bindings which correlate specific attributes to the primary IP address of a virtual machine.

Attribute-based access control is supported on all hardware platforms, and on all ASA Virtual platforms running on ESXi, KVM, or HyperV hypervisors. Attributes can only be retrieved from virtual machines running on an ESXi hypervisor.

- Guidelines for Attribute-Based Network Objects, on page 117
- Configure Attribute-Based Access Control, on page 118
- Monitoring Attribute-Based Network Objects , on page 125
- History for Attribute-Based Access Control, on page 126

Guidelines for Attribute-Based Network Objects

IPv6 Guidelines

- IPv6 addresses not supported by vCenter for host credentials.
- IPv6 is supported for virtual machine bindings where the primary IP address of the virtual machine is an IPv6 address.

Additional Guidelines and Limitations

- Multi-context mode is not supported. Attribute-based network objects are supported for single-mode context only.
- Attribute-based network objects support binding to the virtual machine's primary address only. Binding to multiple vNICs on a single virtual machine is not supported.
- Attribute-based network objects may only be configured for objects used for access groups. Network objects for other features (NAT, etc.) are not supported.

- Virtual machines must be running VMware Tools in order to report primary IP addresses to vCenter. The ASA is not notified of attribute changes unless vCenter knows the IP address of the virtual machine. This is a vCenter restriction.
- Attribute-based network objects are not supported in the Amazon Web Services (AWS) or Microsoft Azure public cloud environments.

Configure Attribute-Based Access Control

The following procedure provides a general sequence for implementing attribute-based access control on managed virtual machines in a VMware ESXi environment.

Procedure

Step 1 Assign custom attribute types and values to your managed virtual machines. See Configure Attributes for vCenter Virtual Machines, on page 118.

Step 2 Configure an attribute agent to connect to your vCenter Server or ESXi host. See Configure a VM Attribute Agent, on page 120.

Step 3 Configure attribute-based network objects needed for your deployment scheme. See Configure Attribute-Based Network Objects, on page 122.

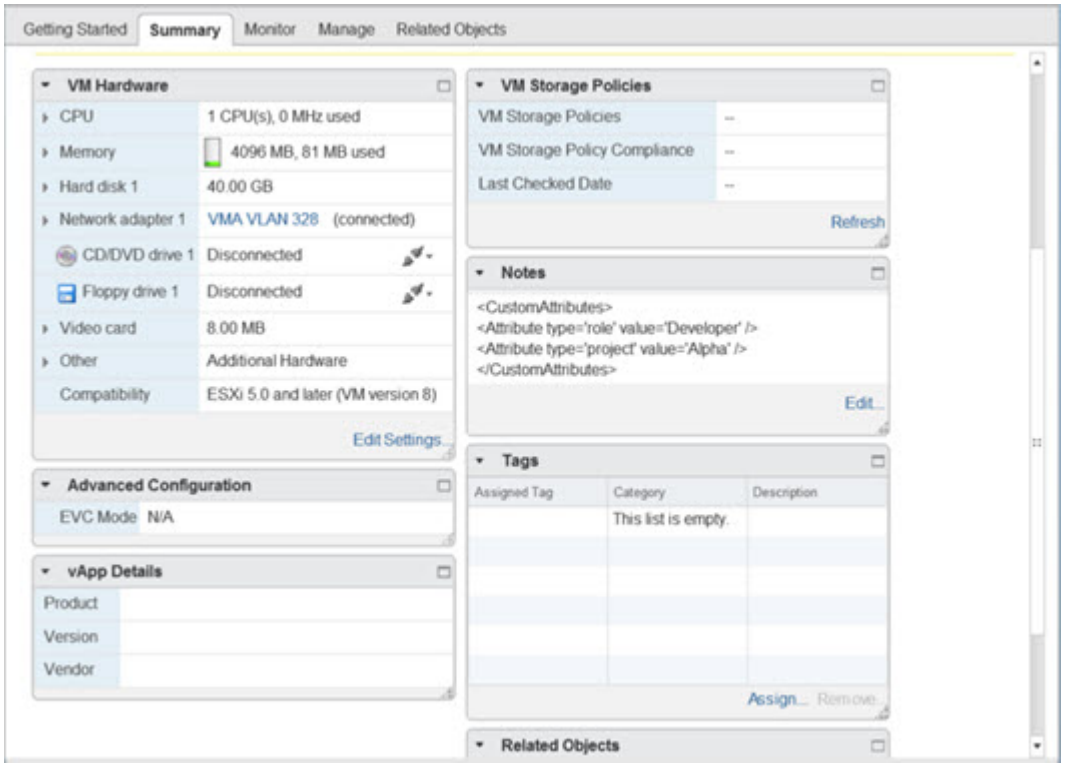
Step 4 Configure the access control lists and rules. See Configure Access Control Using Attribute-Based Network Objects, on page 123 .

Configure Attributes for vCenter Virtual Machines

You assign custom attribute types and values to virtual machines, and associate these attributes to network objects. You can then use these attribute-based network objects to apply ACLs to a set of virtual machines with common user-defined characteristics. For example, you could isolate developer build machines from test machines, or group virtual machines by project and/or location. For the ASA to monitor virtual machines using attributes, you need to make the attributes available to vCenter from the managed virtual machines. You do this by inserting a formatted text file into the Notes field, which is found on the Summary page of virtual machines in vCenter.

You can see the Notes field in the following figure.

Figure 6: Summary Tab of a Virtual Machine in vCenter



To specify custom attributes, you copy a properly formatted XML file into the Notes field for the virtual machine. The format of the file is:

```
<CustomAttributes>
<Attribute type='attribute-type' value='attribute-value'/>
...
</CustomAttributes>
```

A single virtual machine may have multiple attributes defined by repeating the second line above. Note that each line must identify a unique attribute type. If the same attribute type is defined with multiple attribute values, each binding update for that attribute type will overwrite the previous one.

For string attribute values, the value associated with the object definition must be an exact match to the value reported to vCenter by the virtual machine. For example, an attribute value *Build Machine* does not match the annotation value *build machine* on the virtual machine. A binding would not be added to the host-map for this attribute.

You can define multiple unique attribute types in a single file.

Procedure

Step 1 Select the virtual machine from your vCenter inventory.

Step 2 Click the **Summary** tab for the virtual machine.

Step 3 In the **Notes** field, click the **Edit** link.

Step 4 Paste the custom attributes text file into the **Edit Notes** box. The text file should follow the XML template format:

Example:

```
<CustomAttributes>
<Attribute type='attribute-type' value='attribute-value'/>
...
</CustomAttributes>
```

Step 5 Click **OK**.

Example

The following example shows a properly formatted XML text file that defines custom attributes for 'role' and 'project' that you can apply to virtual machines:

```
<CustomAttributes>
<Attribute type='role' value='Developer'/>
<Attribute type='project' value='Alpha'/>
</CustomAttributes>
```

Configure a VM Attribute Agent

You configure a VM attribute agent to communicate with vCenter or a single ESXi host. When you assign attributes to virtual machines within the VMware environment, the attribute agent sends a message to vCenter indicating which attributes have been configured, and vCenter responds with a binding update for every virtual machine where a matching attribute type is configured.

The VM attribute agent and vCenter exchange binding updates as follows:

- If the agent issues a request containing a new attribute type, vCenter responds with a binding update for every virtual machine where the attribute type is configured. After that point, vCenter only issues a new binding when an attribute value is added or changed.
- If a monitored attribute changes for one or more virtual machines, a binding update message is received. Each binding message is identified by the IP address of the virtual machine reporting the attribute value.
- If multiple attributes are being monitored by a single agent, a single binding update contains the current value of all monitored attributes for each virtual machine.
- If a specific attribute being monitored by the agent is not configured on a virtual machine, the binding will contain an empty attribute value for that virtual machine.
- If a virtual machine has not been configured with any monitored attributes, vCenter does not send a binding update.

Each attribute agent communicates with exactly one vCenter or ESXi host. A single ASA may have multiple attribute agents defined, each communicating with a different vCenter, or one or more communicating with the same vCenter.

Procedure

Step 1 Create the VM attribute agent to communicates with vCenter: **attribute source-group** *agent-name* **type** *agent-type*

Example:

```
hostname(config)# attribute source-group VMAgent type esxi
```

The *agent-name* argument specifies the VM attribute agent name. The *type* argument is the type of attribute agent.

Note
Currently ESXi is the only supported agent type.

Step 2 Configure your vCenter host credentials: **host** *ip-address* **username** *ESXi-username* **password** *ESXi-password*

Example:

```
hostname(config-attr)# host 10.122.202.217 user admin password Cisco123
```

Step 3 Configure keepalive settings for vCenter communication: **keepalive retry-interval** *interval* **retry-count** *count*

Example:

```
hostname(config-attr)# keepalive retry-timer 10 retry-count 3
```

The default keepalive timer values are 3 retries at 30-second intervals.

Step 4 Examine the VM attribute agent configuration: **show attribute source-group** *agent-name*

Example:

```
hostname(config-attr)# sh attribute source-group VMAgent

   Attribute agent VMAgent
   Agent type: ESXi
   Agent state: Inactive
   Connection state: Connected
   Host Address: 10.122.202.217
   Retry interval: 30 seconds
   Retry count: 3
```

The *Agent State* remains inactive until you configure a network object and specify attributes to associate with the object.

Step 5 Exit from the attribute configuration mode: **exit**

Example:

```
hostname(config-attr)# exit
```

Configure Attribute-Based Network Objects

Attribute-based network objects filter traffic according to attributes associated with one or more virtual machines in a VMware ESXi environment. You can define access control lists (ACLs) to assign policies to traffic from groups of virtual machines sharing one or more attributes.

For example, you can configure access rules that permit machines with an *engineering* attribute to access machines with a *eng_lab* attribute. A network admininstrator can add or remove engineering machines and lab servers while the security policy managed by the security administrator continues to work automatically without manual updates to the access rules.

Procedure

Step 1 Enable object group search: **object-group-search access-control**

Example:

```
hostname(config)# object-group-search access-control
```

You must enable object-group-search to configure attribute-based network objects.

Step 2 Create or edit an attribute-based network object using the object name: **object network** *object-id*

Example:

```
hostname(config)# object network dev
```

Step 3 Specify an agent, attribute type, and attribute value to associate with the object: **attribute** *agent-name attribute-type attribute-value*

Example:

```
hostname(config-network-object)# attribute VMAgent custom.role Developer
```

The *agent-name* specifies the VM attribute agent; see <XREF>. If you configure an attribute-based network object to use an attribute agent which has not been configured, a placeholder agent is automatically created with no credentials and default keepalive values. This agent remains in the "No credentials available" state until host credentials are supplied using the **host** subcommand

Together, the *attribute-type* and *attribute-value* pair define a unique attribute. The *attribute-type* is an arbitrary string and must include the **custom.** prefix. If you define the same attribute type more than once with multiple attribute values, the last value defined overwrites the previous one.

Examples

The following example creates the attribute-based network object *dev* for a development group, with a role of 'Developer'. The VM attribute agent communicates with vCenter and returns all of the virtual machine bindings that match the attribute *custom.role*:

```
hostname(config)# object network dev
hostname(config-network-object)# attribute VMAgent custom.role Developer
```

The following example creates the attribute-based network object *test* for a test group, with a role of 'Automation'. The VM attribute agent communicates with vCenter and returns all of the virtual machine bindings that match the attribute *custom.role*. Note that this is the same list of virtual machines as the previous example:

```
hostname(config)# object network test
hostname(config-network-object)# attribute VMAgent custom.role Automation
```

The following example creates the attribute-based network object *project* for a project group, with a role of 'Alpha'. The VM attribute agent communicates with vCenter and returns all of the virtual machine bindings that match the attribute *custom.project*. Note that some machines overlap more than one attribute:

```
hostname(config)# object network project
hostname(config-network-object)# attribute VMAgent custom.project Alpha
```

The following example shows a VM attribute agent in active status with pending attribute requests:

```
hostname(config-attr)# show attribute source-group VMAgent

   Attribute agent VMAgent
   Agent type: ESXi
   Agent state: Active
   Connection state: Connected
   Host Address: 10.122.202.217
   Retry interval: 30 seconds
   Retry count: 3
   Attribute requests pending:
       'custom.project'
       'custom.role'
```

Configure Access Control Using Attribute-Based Network Objects

You can use attribute-based network objects when you define access control lists (ACLs) to traffic from groups of virtual machines sharing one or more attributes. Access lists are made up of one or more access control entries (ACEs). An ACE is a single entry in an access list that specifies a permit or deny rule (to forward or drop the packet). Typically a permit or deny rule is applied to a protocol, to a source and destination IP address or network, and, optionally, to the source and destination ports.

When you use attribute-based network objects, you can replace source and/or destination IP addresses with these objects. As virtual machines are deployed, moved, or retired, attributes can be updated on the virtual machines while the assigned access control policies can remain in effect without configuration changes.

For complete information on all of the available options for ACLs, see Configure ACLs, on page 32.

Procedure

Step 1 Create and configure an extended ACL entry (ACE) using attribute-based network objects: **access-list** *access_list_name* **extended** {**deny | permit**} *protocol_argument* **object** *source_object_name* **object** *dest_object_name*

Example:

```
hostname(config)# access-list lab-access extended permit ip object dev object test
```

Note
Repeat as needed for your policies.

The options are:

- *access_list_name*—The name of the new or existing ACL.
- Permit or Deny—The **deny** keyword denies or exempts a packet if the conditions are matched. The **permit** keyword permits or includes a packet if the conditions are matched.
- Protocol—The *protocol_argument* specifies the IP protocol:
  - *name* or *number*—Specifies the protocol name or number. Specify **ip** to apply to all protocols.
  - **object-group** *protocol_grp_id*—Specifies a protocol object group created using the **object-group protocol** command.
- Source Object—**object** specifies an attribute-based network object created using the **object network** command. The *source_object_name* specifies the object from which the packet is being sent.
- Destination Object—**object** specifies an attribute-based network object created using the **object network** command. The *dest_object_name* specifies the object to which the packet is being sent.

Step 2 Bind the ACL to an interface or apply it globally: **access-group** *access_list_name* {**in interface** *interface_name* | **global**}

Example:

```
hostname(config)# access-group lab-access in interface inside
```

For an interface-specific access group:

- Specify the extended ACL name. You can configure one **access-group** command per ACL type per interface.
- The **in** keyword applies the ACL to inbound traffic.
- Specify the **interface** name.

For a global access group, specify the **global** keyword to apply the extended ACL to the inbound direction of all interfaces.

Example

The following example shows how to apply an attribute-based extended ACL globally:

```
hostname(config)# access-list lab-access extended permit ip object dev object test
hostname(config)# access-group lab-access global
hostname(config)# show access-list
access-list cached ACL log flows: total 0, denied 0 (deny-flow-max 4096)
            alert-interval 300
access-list lab-access; 1 elements; name hash: 0x62b4790b
access-list lab-access line 1 extended permit ip object dev object test (hitcnt=0) 0x64a1be76

  access-list lab-access line 1 extended permit ip object dev(2) object test(3) (hitcnt=0)
 0x64a1be76
```

Monitoring Attribute-Based Network Objects

To monitor attribute-based network objects, enter the following commands:

- **show attribute host-map**

  Displays attribute bindings for a given attribute's agent, type, and value.

- **show attribute object-map**

  Displays the object-to-attribute bindings.

- **show attribute source-group**

  Displays the configured VM attribute agents.

Examples

The following example shows a map of the host-to-attribute bindings:

```
hostname# show attribute host-map /all
IP Address-Attribute Bindings Information

         Source/Attribute                          Value
=========================================================================
VMAgent.custom.project                         'Alpha'
     10.15.28.34
     10.15.28.32
     10.15.28.31
     10.15.28.33
VMAgent.custom.role                            'Automation'
     10.15.27.133
     10.15.27.135
     10.15.27.134
VMAgent.custom.role                            'Developer'
     10.15.28.34
     10.15.28.12
     10.15.28.31
```

```
     10.15.28.13
```

The following example shows the object-to-attribute bindings:

```
hostname# show attribute object-map /all
Network Object-Attribute Bindings Information

Object
         Source/Attribute                          Value
=========================================================================
dev
     VMAgent.custom.role                           'Developer'
test
     VMAgent.custom.role                           'Automation'
project
     VMAgent.custom.project                        'Alpha'
```

The following example shows the attribute agent configuration:

```
hostname# show attribute source-group
   Attribute agent VMAgent
   Agent type: ESXi
   Agent state: Active
   Connection state: Connected
   Host Address: 10.122.202.217
   Retry interval: 30 seconds
   Retry count: 3
   Attributes being monitored:
      'custom.role' (2)
```

History for Attribute-Based Access Control

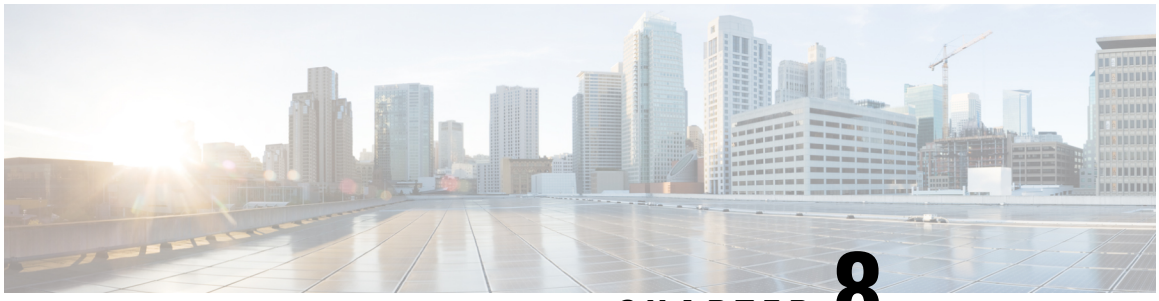
Feature Name	Platform Releases	Description
Support for Attribute-Based Network Objects	9.7.(1)	You can now control network access using virtual machine attributes in addition to traditional network charateristics such as IP addresses, protocols, and ports. The virtual machines must be in a VMware ESXi environment. We introduced the following commands: **object network** *attribute* **attribute** *agent-name attribute-type attribute-value* **attribute source-group** *agent-name* **type** *agent-type* **host** *ip-address* **username** *ESXi-username* **password** *ESXi-password* **keepalive retry-interval** *interval* **retry-count** *count*
Remove support for VM attribute-based network objects from ASA 5506-X (all models), 5508-X, 5512-X, 5516-X.	9.10(1)	You can no longer use VM-attribute based network objects on the following platforms: ASA 5506-X (all models), 5508-X, 5512-X, 5516-X.



PART III

Network Address Translation

- Network Address Translation (NAT), on page 129
- NAT Examples and Reference, on page 181
- Mapping Address and Port (MAP), on page 213



CHAPTER 8

Network Address Translation (NAT)

The following topics explain Network Address Translation (NAT) and how to configure it.

- Why use NAT?, on page 129
- NAT basics, on page 130
- Guidelines for NAT, on page 135
- Dynamic NAT, on page 142
- Dynamic PAT, on page 149
- Static NAT, on page 160
- Identity NAT, on page 170
- Monitoring NAT, on page 174
- History for NAT, on page 175

Why use NAT?

Each computer and device within an IP network is assigned a unique IP address that identifies the host. Because of a shortage of public IPv4 addresses, most of these IP addresses are private, not routable anywhere outside of the private company network. RFC 1918 defines the private IP addresses you can use internally that should not be advertised:

- 10.0.0.0 through 10.255.255.255
- 172.16.0.0 through 172.31.255.255
- 192.168.0.0 through 192.168.255.255

One of the main functions of NAT is to enable private IP networks to connect to the Internet. NAT replaces a private IP address with a public IP address, translating the private addresses in the internal private network into legal, routable addresses that can be used on the public Internet. In this way, NAT conserves public addresses because it can be configured to advertise at a minimum only one public address for the entire network to the outside world.

Other functions of NAT include:

- Security—Keeping internal IP addresses hidden discourages direct attacks.
- IP routing solutions—Overlapping IP addresses are not a problem when you use NAT.

- Flexibility—You can change internal IP addressing schemes without affecting the public addresses available externally; for example, for a server accessible to the Internet, you can maintain a fixed IP address for Internet use, but internally, you can change the server address.
- Translating between IPv4 and IPv6 (Routed mode only)—If you want to connect an IPv6 network to an IPv4 network, NAT lets you translate between the two types of addresses.



Note NAT is not required. If you do not configure NAT for a given set of traffic, that traffic will not be translated, but will have all of the security policies applied as normal.

NAT basics

The following topics explain some of the basics of NAT.

NAT terminology

This document uses the following terminology:

- Real address/host/network/interface—The real address is the address that is defined on the host, before it is translated. In a typical NAT scenario where you want to translate the inside network when it accesses the outside, the inside network would be the "real" network. Note that you can translate any network connected to the device, not just an inside network. Therefore if you configure NAT to translate outside addresses, "real" can refer to the outside network when it accesses the inside network.
- Mapped address/host/network/interface—The mapped address is the address that the real address is translated to. In a typical NAT scenario where you want to translate the inside network when it accesses the outside, the outside network would be the "mapped" network.



Note During address translation, IP addresses configured for the device interfaces are not translated.

- Bidirectional initiation—Static NAT allows connections to be initiated *bidirectionally*, meaning both to the host and from the host.
- Source and destination NAT—For any given packet, both the source and destination IP addresses are compared to the NAT rules, and one or both can be translated/untranslated. For static NAT, the rule is bidirectional, so be aware that "source" and "destination" are used in commands and descriptions throughout this guide even though a given connection might originate at the "destination" address.

NAT types

You can implement NAT using the following methods:

- Dynamic NAT—A group of real IP addresses are mapped to a (usually smaller) group of mapped IP addresses, on a first come, first served basis. Only the real host can initiate traffic. See Dynamic NAT, on page 142.
- Dynamic Port Address Translation (PAT)—A group of real IP addresses are mapped to a single IP address using a unique source port of that IP address. See Dynamic PAT, on page 149.
- Static NAT—A consistent mapping between a real and mapped IP address. Allows bidirectional traffic initiation. See Static NAT, on page 160.
- Identity NAT—A real address is statically translated to itself, essentially bypassing NAT. You might want to configure NAT this way when you want to translate a large group of addresses, but then want to exempt a smaller subset of addresses. See Identity NAT, on page 170.

Network object NAT and Twice NAT

You can implement address translation in two ways: *network object NAT* and *twice NAT*.

We recommend using network object NAT unless you need the extra features that twice NAT provides. It is easier to configure network object NAT, and it might be more reliable for applications such as Voice over IP (VoIP). (For VoIP, you might see a failure in the translation of indirect addresses that do not belong to either of the objects used in the rule.)

Network object NAT

All NAT rules that are configured as a parameter of a network object are considered to be *network object NAT* rules. This is a quick and easy way to configure NAT for a network object. You cannot create these rules for a group object, however.

After you configure the network object, you can then identify the mapped address for that object, either as an inline address or as another network object or network object group.

When a packet enters an interface, both the source and destination IP addresses are checked against the network object NAT rules. The source and destination address in the packet can be translated by separate rules if separate matches are made. These rules are not tied to each other; different combinations of rules can be used depending on the traffic.

Because the rules are never paired, you cannot specify that sourceA/destinationA should have a different translation than sourceA/destinationB. Use twice NAT for that kind of functionality, where you can identify the source and destination address in a single rule.

Twice NAT

Twice NAT lets you identify both the source and destination address in a single rule. Specifying both the source and destination addresses lets you specify that sourceA/destinationA can have a different translation than sourceA/destinationB.



Note For static NAT, the rule is bidirectional, so be aware that "source" and "destination" are used in commands and descriptions throughout this guide even though a given connection might originate at the "destination" address. For example, if you configure static NAT with port address translation, and specify the source address as a Telnet server, and you want all traffic going to that Telnet server to have the port translated from 2323 to 23, then you must specify the *source* ports to be translated (real: 23, mapped: 2323). You specify the source ports because you specified the Telnet server address as the source address.

The destination address is optional. If you specify the destination address, you can either map it to itself (identity NAT), or you can map it to a different address. The destination mapping is always a static mapping.

Comparing Network object NAT and Twice NAT

The main differences between these two NAT types are:

- How you define the real address.
  - Network object NAT—You define NAT as a parameter for a network object. A network object names an IP host, range, or subnet so you can then use the object in the NAT configuration instead of the actual IP addresses. The network object IP address serves as the real address. This method lets you easily add NAT to network objects that might already be used in other parts of your configuration.
  - Twice NAT—You identify a network object or network object group for both the real and mapped addresses. In this case, NAT is not a parameter of the network object; the network object or group is a parameter of the NAT configuration. The ability to use a network object *group* for the real address means that twice NAT is more scalable.
- How source and destination NAT is implemented.
  - Network object NAT— Each rule can apply to either the source or destination of a packet. So two rules might be used, one for the source IP address, and one for the destination IP address. These two rules cannot be tied together to enforce a specific translation for a source/destination combination.
  - Twice NAT—A single rule translates both the source and destination. A packet matches one rule only, and further rules are not checked. Even if you do not configure the optional destination address, a matching packet still matches one twice NAT rule only. The source and destination are tied together, so you can enforce different translations depending on the source/destination combination. For example, sourceA/destinationA can have a different translation than sourceA/destinationB.
- Order of NAT Rules.
  - Network object NAT—Automatically ordered in the NAT table.
  - Twice NAT—Manually ordered in the NAT table (before or after network object NAT rules).

NAT rule order

Network object NAT and twice NAT rules are stored in a single table that is divided into three sections. Section 1 rules are applied first, then section 2, and finally section 3, until a match is found. For example, if

a match is found in section 1, sections 2 and 3 are not evaluated. The following table shows the order of rules within each section.



Note There is also a Section 0, which contains any NAT rules that the system creates for its own use. These rules have priority over all others. The system automatically creates these rules and clears xlates as needed. You cannot add, edit, or modify rules in Section 0.

Table 7: NAT Rule Table

Table Section	Rule Type	Order of Rules within the Section
Section 1	Twice NAT	Applied on a first match basis, in the order they appear in the configuration. Because the first match is applied, you must ensure that specific rules come before more general rules, or the specific rules might not be applied as desired. By default, twice NAT rules are added to section 1. By "specific rules first," we mean: • Static rules should come before dynamic rules. • Rules that include destination translation should come before rules with source translation only. If you cannot eliminate overlapping rules, where more than one rule might apply based on the source or destination address, be especially careful to follow these recommendations. If you do not follow the above recommendation, ensure that accurate routes are configured for the mapped addresses to match the correct NAT rule.
Section 2	Network object NAT	If a match in section 1 is not found, section 2 rules are applied in the following order: **1.** Static rules. **2.** Dynamic rules. **Within each rule type, the following ordering guidelines are used:** **1.** Quantity of real IP addresses—From smallest to largest. For example, an object with one address will be assessed before an object with 10 addresses. **2.** For quantities that are the same, then the IP address number is used, from lowest to highest. For example, 10.1.1.0 is assessed before 11.1.1.0. **3.** If the same IP address is used, then the name of the network object is used, in alphabetical order. For example, abracadabra is assessed before catwoman.

Table Section	Rule Type	Order of Rules within the Section
Section 3	Twice NAT	If a match is still not found, section 3 rules are applied on a first match basis, in the order they appear in the configuration. This section should contain your most general rules. You must also ensure that any specific rules in this section come before general rules that would otherwise apply.

For section 2 rules, for example, you have the following IP addresses defined within network objects:

- 192.168.1.0/24 (static)
- 192.168.1.0/24 (dynamic)
- 10.1.1.0/24 (static)
- 192.168.1.1/32 (static)
- 172.16.1.0/24 (dynamic) (object def)
- 172.16.1.0/24 (dynamic) (object abc)

The resultant ordering would be:

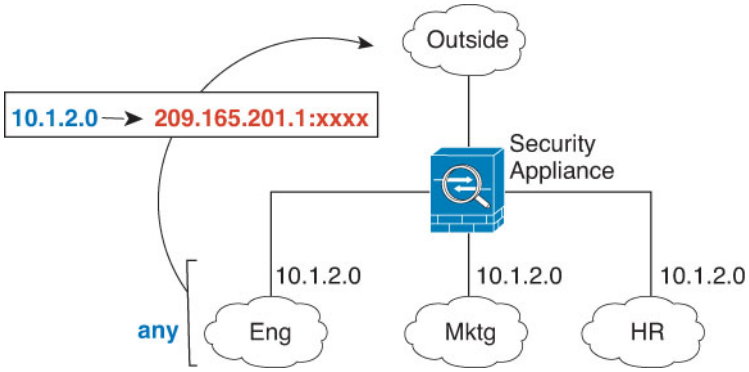
- 192.168.1.1/32 (static)
- 10.1.1.0/24 (static)
- 192.168.1.0/24 (static)
- 172.16.1.0/24 (dynamic) (object abc)
- 172.16.1.0/24 (dynamic) (object def)
- 192.168.1.0/24 (dynamic)

NAT interfaces

Except for bridge group member interfaces, you can configure a NAT rule to apply to any interface (in other words, all interfaces), or you can identify specific real and mapped interfaces. You can also specify any interface for the real address, and a specific interface for the mapped address, or vice versa.

For example, you might want to specify any interface for the real address and specify the outside interface for the mapped address if you use the same private addresses on multiple interfaces, and you want to translate them all to the same global pool when accessing the outside.

Figure 7: Specifying Any Interface



However, the concept of "any" interface does not apply to bridge group member interfaces. When you specify "any" interface, all bridge group member interfaces are excluded. Thus, to apply NAT to bridge group members, you must specify the member interface. This could result in many similar rules where only one interface is different. You cannot configure NAT for the Bridge Virtual Interface (BVI) itself, you can configure NAT for member interfaces only.

Guidelines for NAT

The following topics provide detailed guidelines for implementing NAT.

Firewall mode guidelines for NAT

NAT is supported in routed and transparent firewall mode.

However, configuring NAT on bridge group member interfaces (interfaces that are part of a Bridge Group Virtual Interface, or BVI) has the following restrictions:

- When configuring NAT for the members of a bridge group, you specify the member interface. You cannot configure NAT for the bridge group interface (BVI) itself.
- When doing NAT between bridge group member interfaces, you must specify the real and mapped addresses. You cannot specify "any" as the interface.
- You cannot configure interface PAT when the mapped address is a bridge group member interface, because there is no IP address attached to the interface.
- You cannot translate between IPv4 and IPv6 networks (NAT64/46) when the source and destination interfaces are members of the same bridge group. Static NAT/PAT 44/66, dynamic NAT44/66, and dynamic PAT44 are the only allowed methods; dynamic PAT66 is not supported. However, you can do NAT64/46 between members of different bridge groups, or between a bridge group member (source) and standard routed interface (destination).

IPv6 NAT guidelines

NAT supports IPv6 with the following guidelines and restrictions.

- For standard routed mode interfaces, you can also translate between IPv4 and IPv6.

- You cannot translate between IPv4 and IPv6 for interfaces that are members of the same bridge group. You can translate between two IPv6 or two IPv4 networks only. This restriction does not apply when the interfaces are members of different bridge groups, or between a bridge group member and a standard routed interface.
- You cannot use dynamic PAT for IPv6 (NAT66) when translating between interfaces in the same bridge group. This restriction does not apply when the interfaces are members of different bridge groups, or between a bridge group member and a standard routed interface.
- For static NAT, you can specify an IPv6 subnet up to /64. Larger subnets are not supported.
- When using FTP with NAT46, when an IPv4 FTP client connects to an IPv6 FTP server, the client must use either the extended passive mode (EPSV) or extended port mode (EPRT); PASV and PORT commands are not supported with IPv6.

IPv6 NAT best practices

You can use NAT to translate between IPv6 networks, and also to translate between IPv4 and IPv6 networks (routed mode only). We recommend the following best practices:

- NAT66 (IPv6-to-IPv6)—We recommend using static NAT. Although you can use dynamic NAT or PAT, IPv6 addresses are in such large supply, you do not have to use dynamic NAT. If you do not want to allow returning traffic, you can make the static NAT rule unidirectional (twice NAT only).
- NAT46 (IPv4-to-IPv6)—We recommend using static NAT. Because the IPv6 address space is so much larger than the IPv4 address space, you can easily accommodate a static translation. If you do not want to allow returning traffic, you can make the static NAT rule unidirectional (twice NAT only). When translating to an IPv6 subnet (/96 or lower), the resulting mapped address is by default an IPv4-embedded IPv6 address, where the 32-bits of the IPv4 address is embedded after the IPv6 prefix. For example, if the IPv6 prefix is a /96 prefix, then the IPv4 address is appended in the last 32-bits of the address. For example, if you map 192.168.1.0/24 to 201b::0/96, then 192.168.1.4 will be mapped to 201b::0.192.168.1.4 (shown with mixed notation). If the prefix is smaller, such as /64, then the IPv4 address is appended after the prefix, and a suffix of 0s is appended after the IPv4 address. You can also optionally translate the addresses net-to-net, where the first IPv4 address maps to the first IPv6 address, the second to the second, and so on.
- NAT64 (IPv6-to-IPv4)—You may not have enough IPv4 addresses to accommodate the number of IPv6 addresses. We recommend using a dynamic PAT pool to provide a large number of IPv4 translations.

Additional guidelines for NAT

- NAT rules apply to through the device traffic only, they do not apply to traffic initiated by the device, such as a RADIUS authentication.
- For interfaces that are members of a bridge group, you write NAT rules for the member interfaces. You cannot write NAT rules for the Bridge Virtual Interface (BVI) itself.
- You cannot write NAT rules for a Virtual Tunnel Interface (VTI), which are used in site-to-site VPN. Writing rules for the VTI's source interface will not apply NAT to the VPN tunnel. To write NAT rules that will apply to VPN traffic tunneled on a VTI, you must use "any" as the interface; you cannot explicitly specify interface names.

- (Network object NAT only.) You can only define a single NAT rule for a given object; if you want to configure multiple NAT rules for an object, you need to create multiple objects with different names that specify the same IP address. For example, **object network obj-10.10.10.1-01**, **object network obj-10.10.10.1-02**, and so on.
- If a VPN is defined on an interface, inbound ESP traffic on the interface is not subject to the NAT rules. The system allows the ESP traffic for established VPN tunnels only, dropping traffic not associated with an existing tunnel. This restriction applies to ESP and UDP ports 500 and 4500.
- If you define a site-to-site VPN on a device that is behind a device that is applying dynamic PAT, so that UDP ports 500 and 4500 are not the ones actually used, you must initiate the connection from the device that is behind the PAT device. The responder cannot initiate the security association (SA) because it does not know the correct port numbers.
- If you change the NAT configuration, and you do not want to wait for existing translations to time out before the new NAT configuration is used, you can clear the translation table using the **clear xlate** command in the device CLI. However, clearing the translation table disconnects all current connections that use translations.

  If you create a new NAT rule that should apply to an existing connection (such as a VPN tunnel), you need to use **clear conn** to end the connection. Then, the attempt to re-establish the connection should hit the NAT rule and the connection should be NAT'ed correctly.



Note If you remove a dynamic NAT or PAT rule, and then add a new rule with mapped addresses that overlap the addresses in the removed rule, then the new rule will not be used until all connections associated with the removed rule time out or are cleared using the **clear xlate** or **clear conn** commands. This safeguard ensures that the same address is not assigned to multiple hosts.

- When translating SCTP traffic, use static network object NAT only. Dynamic NAT/PAT is not allowed. Although you can configure static twice NAT, this is not recommended because the topology of the destination part of the SCTP association is unknown.
- Objects and object groups used in NAT cannot be undefined; they must include IP addresses.
- You cannot use an object group with both IPv4 and IPv6 addresses; the object group must include only one type of address.
- (Twice NAT only.) When using **any** as the source address in a NAT rule, the definition of “any” traffic (IPv4 vs. IPv6) depends on the rule. Before the ASA performs NAT on a packet, the packet must be IPv6-to-IPv6 or IPv4-to-IPv4; with this prerequisite, the ASA can determine the value of **any** in a NAT rule. For example, if you configure a rule from “any” to an IPv6 server, and that server was mapped from an IPv4 address, then **any** means “any IPv6 traffic.” If you configure a rule from “any” to “any,” and you map the source to the interface IPv4 address, then **any** means “any IPv4 traffic” because the mapped interface address implies that the destination is also IPv4.
- You can use the same mapped object or group in multiple NAT rules.
- The mapped IP address pool cannot include:
  - The mapped interface IP address. If you specify “any” interface for the rule, then all interface IP addresses are disallowed. For interface PAT (routed mode only), specify the interface name instead of the interface address.

  - The failover interface IP address.
  - (Transparent mode.) The management IP address.
  - (Dynamic NAT.) The standby interface IP address when VPN is enabled.
  - Existing VPN pool addresses.

- Avoid using overlapping addresses in static and dynamic NAT policies. For example, with overlapping addresses, a PPTP connection can fail to get established if the secondary connection for PPTP hits the static instead of dynamic xlate.
- You cannot use overlapping addresses in the source address of a NAT rule and a remote access VPN address pool.
- For application inspection limitations with NAT or PAT, see Default Inspections and NAT Limitations, on page 249.
- The default behavior for identity NAT has proxy ARP enabled, matching other static NAT rules. You can disable proxy ARP if desired. See Routing NAT Packets, on page 192 for more information.
- If you enable the **arp permit-nonconnected** command, the system does not respond to ARP requests if the mapped address is not part of any connected subnet and you also do not specify the mapped interface in the NAT rule (that is, you specify "any" interface). To resolve this problem, specify the mapped interface.
- If you specify a destination interface in a rule, then that interface is used as the egress interface rather than looking up the route in the routing table. However, for identity NAT, you have the option to use a route lookup instead.
- If you use PAT on Sun RPC traffic, which is used to connect to NFS servers, be aware that the NFS server might reject connections if the PAT'ed port is above 1024. The default configuration of NFS servers is to reject connections from ports higher than 1024. The error is typically "Permission Denied." Mapping ports above 1024 might happen if you use the "flat range" option to use the higher port numbers if a port in the lower range is not available, especially if you do not select the option to include the lower range in the flat range.Mapping ports above 1024 happens if you do not select the option to include the reserved ports (1-1023) in the port range of a PAT pool. You can avoid this problem by changing the NFS server configuration to allow all port numbers.
- NAT applies to through traffic only. Traffic generated by the system is not subject to NAT.
- You can improve system performance and reliability by using the transactional commit model for NAT. See the basic settings chapter in the general operations configuration guide for more information. Use the **asp rule-engine transactional-commit nat** command.
- Do not name a network object or group pat-pool, using any combination of upper- or lower-case letters.
- The unidirectional option is mainly useful for testing purposes and might not work with all protocols. For example, SIP requires protocol inspection to translate SIP headers using NAT, but this will not occur if you make the translation unidirectional.
- You cannot use NAT on the internal payload of Protocol Independent Multicast (PIM) registers.
- (Twice NAT) When writing NAT rules for a dual ISP interface setup (primary and backup interfaces using service level agreements in the routing configuration), do not specify destination criteria in the rule. Ensure the rule for the primary interface comes before the rule for the backup interface. This allows the device to choose the correct NAT destination interface based on the current routing state when the

primary ISP is unavailable. If you specify destination objects, the NAT rule will always select the primary interface for the otherwise duplicate rules.

- If you get the ASP drop reason nat-no-xlate-to-pat-pool for traffic that should not match the NAT rules defined for the interface, configure identity NAT rules for the affected traffic so the traffic can pass untranslated.
- If you configure NAT for GRE tunnel endpoints, you must disable keepalives on the endpoints or the tunnel cannot be established. The endpoints send keepalives to the original addresses.
- DHCP and BOOTP share ports UDP/67-68. Because BOOTP is obsolete, writing NAT rules for the bootps port can cause port allocation problems when also running DHCP. Consider using DHCP relay instead for transmitting DHCP requests between network segments.
- In rare cases, return traffic (server to client) with an existing translation (xlate) might be logged as a new flow in Connection Events. This can occur when the client has already terminated the connection and the server sends another packet that reaches the device during the brief interval between connection closure and xlate removal, often due to application behavior or TCP stack cleanup. Because the device removes the xlate only after deleting the connection, a server packet can arrive while the xlate still exists. If no valid connection entry is found, the device logs a separate connection event based on the matched Access Control Policy rule.

Network Object NAT Guidelines for Mapped Address Objects

For dynamic NAT, you must use an object or group for the mapped addresses. For the other NAT types, you can use an object or group, or you have the option of using inline addresses. Network object groups are particularly useful for creating a mapped address pool with discontinuous IP address ranges or multiple hosts or subnets. Use the **object network** and **object-group network** commands to create the objects.

Consider the following guidelines when creating objects for mapped addresses.

- A network object group can contain objects or inline addresses of either IPv4 or IPv6 addresses. The group cannot contain both IPv4 and IPv6 addresses; it must contain one type only.
- See Additional guidelines for NAT, on page 136 for information about disallowed mapped IP addresses.
- Do not name a network object or group pat-pool, using any combination of upper- or lower-case letters.
- Dynamic NAT:
  - You cannot use an inline address; you must configure a network object or group.
  - The object or group cannot contain a subnet; the object must define a range; the group can include hosts and ranges.
  - If a mapped network object contains both ranges and host IP addresses, then the ranges are used for dynamic NAT, and then the host IP addresses are used as a PAT fallback. If the object contains one host address only, it is used for PAT.
- Dynamic PAT (Hide):
  - Instead of using an object, you can optionally configure an inline host address or specify the interface address.
  - If you use an object, the object or group cannot contain a subnet. The object must define a host, or for a PAT pool, a range. The group (for a PAT pool) can include hosts and ranges.

- Static NAT or Static NAT with port translation:
  - Instead of using an object, you can configure an inline address or specify the interface address (for static NAT-with-port-translation).
  - If you use an object, the object or group can contain a host, range, or subnet.
- Identity NAT
  - Instead of using an object, you can configure an inline address.
  - If you use an object, the object must match the real addresses you want to translate.

Twice NAT Guidelines for Real and Mapped Address Objects

For each NAT rule, configure up to four network objects or groups for:

- Source real address
- Source mapped address
- Destination real address
- Destination mapped address

Objects are required unless you specify the **any** keyword inline to represent all traffic, or for some types of NAT, the **interface** keyword to represent the interface address. Network object groups are particularly useful for creating a mapped address pool with discontinuous IP address ranges or multiple hosts or subnets. Use the **object network** and **object-group network** commands to create the objects.

Consider the following guidelines when creating objects for twice NAT.

- A network object group can contain objects or inline addresses of either IPv4 or IPv6 addresses. The group cannot contain both IPv4 and IPv6 addresses; it must contain one type only.
- See Additional guidelines for NAT, on page 136 for information about disallowed mapped IP addresses.
- Do not name a network object or group pat-pool, using any combination of upper- or lower-case letters.
- Source Dynamic NAT:
  - You typically configure a larger group of real addresses to be mapped to a smaller group.
  - The mapped object or group cannot contain a subnet; the object must define a range; the group can include hosts and ranges.
  - If a mapped network object contains both ranges and host IP addresses, then the ranges are used for dynamic NAT, and the host IP addresses are used as a PAT fallback. If the object contains one host address only, it is used for PAT.
- Source Dynamic PAT (Hide):
  - If you use an object, the object or group cannot contain a subnet. The object must define a host, or for a PAT pool, a range. The group (for a PAT pool) can include hosts and ranges.
- Source Static NAT or Static NAT with port translation:

- The mapped object or group can contain a host, range, or subnet.
- The static mapping is typically one-to-one, so the real addresses have the same quantity as the mapped addresses. You can, however, have different quantities if desired.

- Source Identity NAT
  - The real and mapped objects must match. You can use the same object for both, or you can create separate objects that contain the same IP addresses.

- Destination Static NAT or Static NAT with port translation (the destination translation is always static):
  - Although the main feature of twice NAT is the inclusion of the destination IP address, the destination address is optional. If you do specify the destination address, you can configure static translation for that address or just use identity NAT for it. You might want to configure twice NAT without a destination address to take advantage of some of the other qualities of twice NAT, including the use of network object groups for real addresses, or manually ordering of rules. For more information, see Comparing Network object NAT and Twice NAT, on page 132.
  - For identity NAT, the real and mapped objects must match. You can use the same object for both, or you can create separate objects that contain the same IP addresses.
  - The static mapping is typically one-to-one, so the real addresses have the same quantity as the mapped addresses. You can, however, have different quantities if desired.
  - For static interface NAT with port translation (routed mode only), you can specify the **interface** keyword instead of a network object/group for the mapped address.
  - You can use a fully-qualified domain name, such as www.example.com, as the translated (mapped) destination. For details, see FQDN destination guidelines, on page 141.

FQDN destination guidelines

You can specify the translated (mapped) destination in a twice NAT rule using a fully-qualified domain name (FQDN) network object instead of an IP address. For example, you can create a rule based on traffic that is destined for the www.example.com web server.

When using an FQDN, the system obtains the DNS resolution and writes the NAT rule based on the returned address. If you are using multiple DNS server groups, the filter domains are honored and the address is requested from the appropriate group based on the filters. If more than one address is obtained from the DNS server, the address used is based on the following:

- If there is an address on the same subnet as the specified interface, that address is used. If there isn't one on the same subnet, the first address returned is used.
- The IP type for the translated source and translated destination must match. For example, if the translated source address is IPv6, the FQDN object must specify IPv6 as the address type. If the translated source is IPv4, the FQDN object must specify IPv4 as the address type.

You cannot include an FQDN object in a network group that is used for manual NAT destination. In NAT, an FQDN object must be used alone, as only a single destination host makes sense for this type of NAT rule.

If the FQDN cannot be resolved to an IP address, the rule is not functional until a DNS resolution is obtained.

Twice NAT Guidelines for Service Objects for Real and Mapped Ports

You can optionally configure service objects for:

- **Source real port (Static only)** *or* **Destination real port**
- **Source mapped port (Static only)** *or* **Destination mapped port**

Use the **object service** command to create the objects.

Consider the following guidelines when creating objects for twice NAT.

- NAT supports TCP, UDP, and SCTP only. When translating a port, be sure the protocols in the real and mapped service objects are identical (for example, both TCP). Although you can configure static twice NAT rules with SCTP port specifications, this is not recommended, because the topology of the destination part of the SCTP association is unknown. Use static object NAT instead for SCTP.
- The "not equal" (**neq**) operator is not supported.
- For identity port translation, you can use the same service object for both the real and mapped ports.
- Source Dynamic NAT—Source Dynamic NAT does not support port translation.
- Source Dynamic PAT (Hide)—Source Dynamic PAT does not support port translation.
- Source Static NAT, Static NAT with port translation, or Identity NAT—A service object can contain both a source and destination port; however, you should specify *either* the source *or* the destination port for both service objects. You should only specify *both* the source and destination ports if your application uses a fixed source port (such as some DNS servers); but fixed source ports are rare. For example, if you want to translate the port for the source host, then configure the source service.
- Destination Static NAT or Static NAT with port translation (the destination translation is always static)—For non-static source NAT, you can only perform port translation on the destination. A service object can contain both a source and destination port, but only the destination port is used in this case. If you specify the source port, it will be ignored.

Dynamic NAT

The following topics explain dynamic NAT and how to configure it.

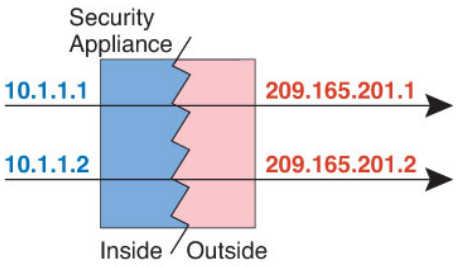
About dynamic NAT

Dynamic NAT translates a group of real addresses to a pool of mapped addresses that are routable on the destination network. The mapped pool typically includes fewer addresses than the real group. When a host you want to translate accesses the destination network, NAT assigns the host an IP address from the mapped pool. The translation is created only when the real host initiates the connection. The translation is in place only for the duration of the connection, and a given user does not keep the same IP address after the translation times out. Users on the destination network, therefore, cannot initiate a reliable connection to a host that uses dynamic NAT, even if the connection is allowed by an access rule.

Note For the duration of the translation, a remote host can initiate a connection to the translated host if an access rule allows it. Because the address is unpredictable, a connection to the host is unlikely. Nevertheless, in this case you can rely on the security of the access rule. A successful connection from a remote host can reset the idle timer for the connection.

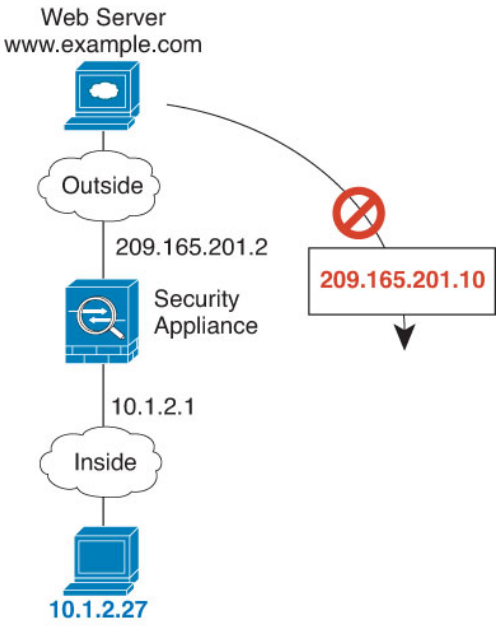
The following figure shows a typical dynamic NAT scenario. Only real hosts can create a NAT session, and responding traffic is allowed back.

Figure 8: Dynamic NAT



The following figure shows a remote host attempting to initiate a connection to a mapped address. This address is not currently in the translation table; therefore, the packet is dropped.

Figure 9: Remote Host Attempts to Initiate a Connection to a Mapped Address



Dynamic NAT disadvantages and advantages

Dynamic NAT has these disadvantages:

- If the mapped pool has fewer addresses than the real group, you could run out of addresses if the amount of traffic is more than expected.

Use PAT or a PAT fall-back method if this event occurs often because PAT provides over 64,000 translations using ports of a single address.

- You have to use a large number of routable addresses in the mapped pool, and routable addresses may not be available in large quantities.

The advantage of dynamic NAT is that some protocols cannot use PAT. PAT does not work with the following:

- IP protocols that do not have a port to overload, such as GRE version 0.
- Some multimedia applications that have a data stream on one port, the control path on another port, and are not open standard.

Configure Dynamic Network Object NAT

This section describes how to configure network object NAT for dynamic NAT.

Procedure

Step 1 Create a host or range network object (**object network** command), or a network object group (**object-group network** command), for the mapped addresses.

- The object or group cannot contain a subnet; the object must define a range; the group can include hosts and ranges.
- If a mapped network object contains both ranges and host IP addresses, then the ranges are used for dynamic NAT, and then the host IP addresses are used as a PAT fallback. If the object contains one host address only, it is used for PAT.

Step 2 Create or edit the network object for which you want to configure NAT: **object network** *obj_name*

Example:

```
hostname(config)# object network my-host-obj1
```

Step 3 (Skip when editing an object that has the right address.) Define the real IPv4 or IPv6 addresses that you want to translate.

- **host** {*IPv4_address* | *IPv6_address*}—The IPv4 or IPv6 address of a single host. For example, 10.1.1.1 or 2001:DB8::0DB8:800:200C:417A.
- **subnet** {*IPv4_address IPv4_mask* | *IPv6_address*/*IPv6_prefix*}—The address of a network. For IPv4 subnets, include the mask after a space, for example, 10.0.0.0 255.0.0.0. For IPv6, include the address and prefix as a single unit (no spaces), such as 2001:DB8:0:CD30::/60.
- **range** *start_address end_address*—A range of addresses. You can specify IPv4 or IPv6 ranges. Do not include masks or prefixes.

Example:

```
hostname(config-network-object)# host 10.2.2.2
```

Step 4 Configure **dynamic NAT** for the object IP addresses. You can only define a single NAT rule for a given object.

nat [**(***real_ifc***,***mapped_ifc***)**] **dynamic** *mapped_obj* [**interface** [**ipv6**]] [**dns**]

Where:

- Interfaces—(Required for bridge group member interfaces.) Specify the real (*real_ifc*) and mapped (*mapped_ifc*) interfaces. Be sure to include the parentheses. In routed mode, if you do not specify the real and mapped interfaces, all interfaces are used. You can also specify the keyword **any** for one or both of the interfaces, for example (any,outside), but **any** does not apply to bridge group member interfaces.
- Mapped IP address—Specify the network object or network object group that includes the mapped IP addresses.
- Interface PAT fallback—(Optional) The **interface** keyword enables interface PAT fallback. After the mapped IP addresses are used up, then the IP address of the mapped interface is used. If you specify **ipv6**, then the IPv6 address of the interface is used. For this option, you must configure a specific interface for the *mapped_ifc*. (You cannot specify **interface** when the mapped interface is a bridge group member.)
- DNS—(Optional) The **dns** keyword translates DNS replies. Be sure DNS inspection is enabled (it is enabled by default). See Rewriting DNS queries and responses using NAT, on page 207 for more information.

Example:

```
hostname(config-network-object)# nat (inside,outside) dynamic MAPPED_IPS interface
```

Examples

The following example configures dynamic NAT that hides the 192.168.2.0 network behind a range of outside addresses 10.2.2.1 through 10.2.2.10:

```
hostname(config)# object network my-range-obj
hostname(config-network-object)# range 10.2.2.1 10.2.2.10
hostname(config)# object network my-inside-net
hostname(config-network-object)# subnet 192.168.2.0 255.255.255.0
hostname(config-network-object)# nat (inside,outside) dynamic my-range-obj
```

The following example configures dynamic NAT with dynamic PAT backup. Hosts on inside network 10.76.11.0 are mapped first to the nat-range1 pool (10.10.10.10-10.10.10.20). After all addresses in the nat-range1 pool are allocated, dynamic PAT is performed using the pat-ip1 address (10.10.10.21). In the unlikely event that the PAT translations are also used up, dynamic PAT is performed using the outside interface address.

```
hostname(config)# object network nat-range1
hostname(config-network-object)# range 10.10.10.10 10.10.10.20

hostname(config-network-object)# object network pat-ip1
hostname(config-network-object)# host 10.10.10.21

hostname(config-network-object)# object-group network nat-pat-grp
hostname(config-network-object)# network-object object nat-range1
```

```
hostname(config-network-object)# network-object object pat-ip1

hostname(config-network-object)# object network my_net_obj5
hostname(config-network-object)# subnet 10.76.11.0 255.255.255.0
hostname(config-network-object)# nat (inside,outside) dynamic nat-pat-grp interface
```

The following example configures dynamic NAT with dynamic PAT backup to translate IPv6 hosts to IPv4. Hosts on inside network 2001:DB8::/96 are mapped first to the IPv4_NAT_RANGE pool (209.165.201.1 to 209.165.201.30). After all addresses in the IPv4_NAT_RANGE pool are allocated, dynamic PAT is performed using the IPv4_PAT address (209.165.201.31). In the event that the PAT translations are also used up, dynamic PAT is performed using the outside interface address.

```
hostname(config)# object network IPv4_NAT_RANGE
hostname(config-network-object)# range 209.165.201.1 209.165.201.30

hostname(config-network-object)# object network IPv4_PAT
hostname(config-network-object)# host 209.165.201.31

hostname(config-network-object)# object-group network IPv4_GROUP
hostname(config-network-object)# network-object object IPv4_NAT_RANGE
hostname(config-network-object)# network-object object IPv4_PAT

hostname(config-network-object)# object network my_net_obj5
hostname(config-network-object)# subnet 2001:DB8::/96
hostname(config-network-object)# nat (inside,outside) dynamic IPv4_GROUP interface
```

Configure Dynamic Twice NAT

This section describes how to configure twice NAT for dynamic NAT.

Procedure

Step 1 Create host or range network objects (**object network** command), or network object groups (**object-group network** command), for the source real addresses, the source mapped addresses, the destination real addresses, and the destination mapped addresses. You can also use an FQDN network object for the destination mapped address.

- If you want to translate all source traffic, you can skip adding an object for the source real addresses, and instead specify the **any** keyword in the **nat** command.
- If you want to configure destination static interface NAT with port translation only, you can skip adding an object for the destination mapped addresses, and instead specify the **interface** keyword in the **nat** command.

If you do create objects, consider the following guidelines:

- You typically configure a larger group of real addresses to be mapped to a smaller group.
- The object or group cannot contain a subnet; the object must define a range; the group can include hosts and ranges.

- If a mapped network object contains both ranges and host IP addresses, then the ranges are used for dynamic NAT, and then the host IP addresses are used as a PAT fallback. If the object contains one host address only, it is used for PAT.

Step 2 (Optional.) Create service objects for the destination real ports and the destination mapped ports.

For dynamic NAT, you can only perform port translation on the destination. A service object can contain both a source and destination port, but only the destination port is used in this case. If you specify the source port, it will be ignored.

Step 3 Configure **dynamic NAT**.

nat [**(***real_ifc***,***mapped_ifc***)**] [*line* | {**after-auto** [*line*]}] **source dynamic** {*real_obj* | **any**} {*mapped_obj* [**interface** [**ipv6**]]} [**destination static** {*mapped_obj* | **interface** [**ipv6**]} *real_obj*] [**service** *mapped_dest_svc_obj real_dest_svc_obj*] [**dns**] [**unidirectional**] [**inactive**] [**description** *desc*]

Where:

- Interfaces—(Required for bridge group member interfaces.) Specify the real (*real_ifc*) and mapped (*mapped_ifc*) interfaces. Be sure to include the parentheses. In routed mode, if you do not specify the real and mapped interfaces, all interfaces are used. You can also specify the keyword **any** for one or both of the interfaces, for example (any,outside), but **any** does not apply to bridge group member interfaces.
- Section and Line—(Optional.) By default, the NAT rule is added to the end of section 1 of the NAT table (see NAT rule order, on page 132). If you want to add the rule into section 3 instead (after the network object NAT rules), then use the **after-auto** keyword. You can insert a rule anywhere in the applicable section using the *line* argument.
- Source addresses:
  - Real—Specify a network object, group, or the **any** keyword.
  - Mapped—Specify a different network object or group. You can optionally configure the following fallback method:
    - Interface PAT fallback—(Optional.) The **interface** keyword enables interface PAT fallback. After the mapped IP addresses are used up, then the IP address of the mapped interface is used. If you specify **ipv6**, then the IPv6 address of the interface is used. For this option, you must configure a specific interface for the *mapped_ifc*. (You cannot specify **interface** when the mapped interface is a bridge group member).
- Destination addresses (Optional):
  - Mapped—Specify a network object or group, or for static interface NAT with port translation only, specify the **interface** keyword. If you specify **ipv6**, then the IPv6 address of the interface is used. If you specify **interface**, be sure to also configure the **service** keyword. For this option, you must configure a specific interface for the *real_ifc*. See Static NAT with port translation, on page 161 for more information.
  - Real—Specify a network object or group. For identity NAT, simply use the same object or group for both the real and mapped addresses.
- Destination port—(Optional.) Specify the **service** keyword along with the mapped and real service objects. For identity port translation, simply use the same service object for both the real and mapped ports.

- DNS—(Optional; for a source-only rule.) The **dns** keyword translates DNS replies. Be sure DNS inspection is enabled (it is enabled by default). You cannot configure the **dns** keyword if you configure a **destination** address. See Rewriting DNS queries and responses using NAT, on page 207 for more information.
- Unidirectional—(Optional.) Specify **unidirectional** so the destination addresses cannot initiate traffic to the source addresses.
- Inactive—(Optional.) To make this rule inactive without having to remove the command, use the **inactive** keyword. To reactivate it, reenter the whole command without the **inactive** keyword.
- Description—(Optional.) Provide a description up to 200 characters using the **description** keyword.

Example:

```
hostname(config)# nat (inside,outside) source dynamic MyInsNet NAT_POOL
destination static Server1_mapped Server1 service MAPPED_SVC REAL_SVC
```

Examples

The following example configures dynamic NAT for inside network 10.1.1.0/24 when accessing servers on the 209.165.201.1/27 network as well as servers on the 203.0.113.0/24 network:

```
hostname(config)# object network INSIDE_NW
hostname(config-network-object)# subnet 10.1.1.0 255.255.255.0

hostname(config)# object network MAPPED_1
hostname(config-network-object)# range 209.165.200.225 209.165.200.254

hostname(config)# object network MAPPED_2
hostname(config-network-object)# range 209.165.202.129 209.165.200.158

hostname(config)# object network SERVERS_1
hostname(config-network-object)# subnet 209.165.201.0 255.255.255.224

hostname(config)# object network SERVERS_2
hostname(config-network-object)# subnet 203.0.113.0 255.255.255.0

hostname(config)# nat (inside,outside) source dynamic INSIDE_NW MAPPED_1
destination static SERVERS_1 SERVERS_1
hostname(config)# nat (inside,outside) source dynamic INSIDE_NW MAPPED_2
destination static SERVERS_2 SERVERS_2
```

The following example configures dynamic NAT for an IPv6 inside network 2001:DB8:AAAA::/96 when accessing servers on the IPv4 209.165.201.1/27 network as well as servers on the 203.0.113.0/24 network:

```
hostname(config)# object network INSIDE_NW
hostname(config-network-object)# subnet 2001:DB8:AAAA::/96

hostname(config)# object network MAPPED_1
hostname(config-network-object)# range 209.165.200.225 209.165.200.254

hostname(config)# object network MAPPED_2
hostname(config-network-object)# range 209.165.202.129 209.165.200.158
```

```
hostname(config)# object network SERVERS_1
hostname(config-network-object)# subnet 209.165.201.0 255.255.255.224

hostname(config)# object network SERVERS_2
hostname(config-network-object)# subnet 203.0.113.0 255.255.255.0

hostname(config)# nat (inside,outside) source dynamic INSIDE_NW MAPPED_1
destination static SERVERS_1 SERVERS_1
hostname(config)# nat (inside,outside) source dynamic INSIDE_NW MAPPED_2
destination static SERVERS_2 SERVERS_2
```

Dynamic PAT

The following topics describe dynamic PAT.

About dynamic PAT

Dynamic PAT translates multiple real addresses to a single mapped IP address by translating the real address and source port to the mapped address and a unique port.

Each connection requires a separate translation session because the source port differs for each connection. For example, 10.1.1.1:1025 requires a separate translation from 10.1.1.1:1026.

The following figure shows a typical dynamic PAT scenario. Only real hosts can create a NAT session, and responding traffic is allowed back. The mapped address is the same for each translation, but the port is dynamically assigned.

Figure 10: Dynamic PAT



For the duration of the translation, a remote host on the destination network can initiate a connection to the translated host if an access rule allows it. Because the port address (both real and mapped) is unpredictable, a connection to the host is unlikely. Nevertheless, in this case you can rely on the security of the access rule.

After the connection expires, the port translation also expires. For multi-session PAT, the PAT timeout is used, 30 seconds by default. For per-session PAT, the xlate is immediately removed.

Note We recommend that you use different PAT pools for each interface. If you use the same pool for multiple interfaces, especially if you use it for "any" interface, the pool can be quickly exhausted, with no ports available for new translations.

Dynamic PAT disadvantages and advantages

Dynamic PAT lets you use a single mapped address, thus conserving routable addresses. You can even use the ASA interface IP address as the PAT address.

You cannot use dynamic PAT for IPv6 (NAT66) when translating between interfaces in the same bridge group. This restriction does not apply when the interfaces are members of different bridge groups, or between a bridge group member and a standard routed interface.

Dynamic PAT does not work with some multimedia applications that have a data stream that is different from the control path. See Default Inspections and NAT Limitations, on page 249 for more information about NAT and PAT support.

Dynamic PAT might also create a large number of connections appearing to come from a single IP address, and servers might interpret the traffic as a DoS attack. You can configure a PAT pool of addresses and use a round-robin assignment of PAT addresses to mitigate this situation.

PAT pool object guidelines

When creating network objects for a PAT pool, follow these guidelines.

For a PAT pool

- Ports are mapped to an available port in the 1024 to 65535 range. You can optionally include the reserved ports, those below 1024, to make the entire port range available for translations.

  When operating in a cluster, blocks of 512 ports per address are allocated to the members of the cluster, and mappings are made within these port blocks. If you also enable block allocation, the ports are distributed according to the block allocation size, whose default is also 512. If you change the cluster unit limit (the size of the cluster), ensure that you clear xlates or reboot the devices so that PAT pools can be appropriately reallocated to the cluster units.

- If you enable block allocation for a PAT pool, port blocks are allocated in the 1024-65535 range only. Thus, if an application requires a low port number (1-1023), it might not work. For example, an application requesting port 22 (SSH) will get a mapped port within the range of 1024-65535 and within the block allocated to the host.

- If you use the same PAT pool object in two separate rules, then be sure to specify the same options for each rule. For example, if one rule specifies extended PAT, then the other rule must also specify extended PAT.

- If a host has an existing connection, then subsequent connections from that host use the same PAT IP address. If no ports are available, this can prevent the connection. Use the round robin option to avoid this problem.

- For best performance, limit the number of IP addresses within a PAT pool to 10,000.

For extended PAT for a PAT pool

- Many application inspections do not support extended PAT. See Default Inspections and NAT Limitations, on page 249 for a complete list of unsupported inspections.

- If you enable extended PAT for a dynamic PAT rule, then you cannot also use an address in the PAT pool as the PAT address in a separate static NAT with port translation rule. For example, if the PAT pool includes 10.1.1.1, then you cannot create a static NAT-with-port-translation rule using 10.1.1.1 as the PAT address.

- If you use a PAT pool and specify an interface for fallback, you cannot specify extended PAT.
- For VoIP deployments that use ICE or TURN, do not use extended PAT. ICE and TURN rely on the PAT binding to be the same for all destinations.
- You cannot use extended PAT on units in a cluster.
- Extended PAT increases memory usage on the device.

For round robin for a PAT pool

- If a host has an existing connection, then subsequent connections from that host will use the same PAT IP address if ports are available. However, this "stickiness" does not survive a failover. If the device fails over, then subsequent connections from a host might not use the initial IP address.
- IP address "stickiness" is also impacted if you mix PAT pool/round robin rules with interface PAT rules on the same interface. For any given interface, choose either a PAT pool or interface PAT; do not create competing PAT rules.
- Round robin, especially when combined with extended PAT, can consume a large amount of memory. Because NAT pools are created for every mapped protocol/IP address/port range, round robin results in a large number of concurrent NAT pools, which use memory. Extended PAT results in an even larger number of concurrent NAT pools.

Configure Dynamic Network Object PAT

This section describes how to configure network object NAT for dynamic PAT.

Procedure

Step 1 (Optional.) Create a host or range network object (**object network** command), or a network object group (**object-group network** command), for the mapped addresses.

- Instead of using an object, you can optionally configure an inline host address or specify the interface address.
- If you use an object, the object or group cannot contain a subnet; the object must define a host, or for a PAT pool, a range; the group (for a PAT pool) can include hosts and ranges.

Step 2 Create or edit the network object for which you want to configure NAT: **object network** *obj_name*

Example:

```
hostname(config)# object network my-host-obj1
```

Step 3 (Skip when editing an object that has the right address.) Define the real IPv4 or IPv6 addresses that you want to translate.

- **host** {*IPv4_address* | *IPv6_address*}—The IPv4 or IPv6 address of a single host. For example, 10.1.1.1 or 2001:DB8::0DB8:800:200C:417A.

- **subnet** {*IPv4_address IPv4_mask* | *IPv6_address*/*IPv6_prefix*}—The address of a network. For IPv4 subnets, include the mask after a space, for example, 10.0.0.0 255.0.0.0. For IPv6, include the address and prefix as a single unit (no spaces), such as 2001:DB8:0:CD30::/60.
- **range** *start_address end_address*—A range of addresses. You can specify IPv4 or IPv6 ranges. Do not include masks or prefixes.

Example:

```
hostname(config-network-object)# range 10.1.1.1 10.1.1.90
```

Step 4 Configure **dynamic PAT** for the object IP addresses. You can only define a single NAT rule for a given object.

nat [(*real_ifc*,*mapped_ifc*)] **dynamic** {*mapped_inline_host_ip* | *mapped_obj* | **pat-pool** *mapped-obj* [**round-robin**] [**extended**] [**include-reserve**] [**block-allocation**] | **interface** [**ipv6**]} [**interface** [**ipv6**]]

Where:

- Interfaces—(Required for bridge group member interfaces.) Specify the real (*real_ifc*) and mapped (*mapped_ifc*) interfaces. Be sure to include the parentheses. In routed mode, if you do not specify the real and mapped interfaces, all interfaces are used. You can also specify the keyword **any** for one or both of the interfaces, for example (any,outside), but **any** does not apply to bridge group member interfaces.
- Mapped IP address—You can specify the mapped IP address as:
  - *mapped_inline_host_ip*—An inline host address.
  - *mapped_obj*—A network object that is defined as a host address.
  - **pat-pool** *mapped-obj*—A network object or group that contains multiple addresses.
  - **interface** [**ipv6**]—The IP address of the mapped interface is used as the mapped address. If you specify **ipv6**, then the IPv6 address of the interface is used. For this option, you must configure a specific interface for the *mapped_ifc*. (You cannot specify **interface** when the mapped interface is a bridge group member.) You must use this keyword when you want to use the interface IP address; you cannot enter it inline or as an object.
- For a PAT pool, you can specify one or more of the following options:
  - **round-robin**—Enables round-robin address allocation for a PAT pool. Without round robin, by default all ports for a PAT address will be allocated before the next PAT address is used. The round-robin method assigns an address/port from each PAT address in the pool before returning to use the first address again, and then the second address, and so on.
  - **extended**—Enables extended PAT. Extended PAT uses 65535 ports per *service*, as opposed to per IP address, by including the destination address and port in the translation information. Normally, the destination port and address are not considered when creating PAT translations, so you are limited to 65535 ports per PAT address. For example, with extended PAT, you can create a translation of 10.1.1.1:1027 when going to 192.168.1.7:23 as well as a translation of 10.1.1.1:1027 when going to 192.168.1.7:80.
  - **include-reserve**—Includes the reserved ports, 1-1023, in the range of ports that are available for address translation. If you do not specify this option, addresses are translated to ports in the 1024-65535 range only.

- **block-allocation**—Enables port block allocation. For carrier-grade or large-scale PAT, you can allocate a block of ports for each host, rather than have NAT allocate one port translation at a time. If you allocate a block of ports, subsequent connections from the host use new randomly-selected ports within the block. If necessary, additional blocks are allocated if the host has active connections for all ports in the original block. Port blocks are allocated in the 1024-65535 range only. Port block allocation is compatible with **round-robin**, but you cannot use the **extended** option. You also cannot use interface PAT fallback.

- Interface PAT fallback—(Optional.) The **interface** [**ipv6**] keyword enables interface PAT fallback when entered after a primary PAT address. After the primary PAT addresses are used up, then the IP address of the mapped interface is used. If you specify **ipv6**, then the IPv6 address of the interface is used. For this option, you must configure a specific interface for the *mapped_ifc*. (You cannot specify **interface** when the mapped interface is a bridge group member.)

Example:

```
hostname(config-network-object)# nat (any,outside) dynamic interface
```

Examples

The following example configures dynamic PAT that hides the 192.168.2.0 network behind address 10.2.2.2:

```
hostname(config)# object network my-inside-net
hostname(config-network-object)# subnet 192.168.2.0 255.255.255.0
hostname(config-network-object)# nat (inside,outside) dynamic 10.2.2.2
```

The following example configures dynamic PAT that hides the 192.168.2.0 network behind the outside interface address:

```
hostname(config)# object network my-inside-net
hostname(config-network-object)# subnet 192.168.2.0 255.255.255.0
hostname(config-network-object)# nat (inside,outside) dynamic interface
```

The following example configures dynamic PAT with a PAT pool to translate the inside IPv6 network to an outside IPv4 network:

```
hostname(config)# object network IPv4_POOL
hostname(config-network-object)# range 203.0.113.1 203.0.113.254
hostname(config)# object network IPv6_INSIDE
hostname(config-network-object)# subnet 2001:DB8::/96
hostname(config-network-object)# nat (inside,outside) dynamic pat-pool IPv4_POOL
```

Configure Dynamic Twice PAT

This section describes how to configure twice NAT for dynamic PAT.

Procedure

Step 1 Create host or range network objects (**object network** command), or network object groups (**object-group network** command), for the source real addresses, the source mapped addresses, the destination real addresses, and the destination mapped addresses. You can also use an FQDN network object for the destination mapped address.

- If you want to translate all source traffic, you can skip adding an object for the source real addresses, and instead specify the **any** keyword in the **nat** command.
- If you want to use the interface address as the mapped address, you can skip adding an object for the source mapped addresses, and instead specify the **interface** keyword in the **nat** command.
- If you want to configure destination static interface NAT with port translation only, you can skip adding an object for the destination mapped addresses, and instead specify the **interface** keyword in the **nat** command.

If you use an object, the object or group cannot contain a subnet. The object must define a host, or for a PAT pool, a range. The group (for a PAT pool) can include hosts and ranges.

Step 2 (Optional.) Create service objects for the destination real ports and the destination mapped ports.

For dynamic NAT, you can only perform port translation on the destination. A service object can contain both a source and destination port, but only the destination port is used in this case. If you specify the source port, it will be ignored.

Step 3 Configure **dynamic PAT**.

nat [(*real_ifc*,*mapped_ifc*)] [*line* | **after-auto** [*line*]] **source dynamic** {*real-obj* | **any**} {*mapped_obj* [**interface** [**ipv6**]] | **pat-pool** *mapped-obj* [**round-robin**] [**extended**] [**include-reserve**] [**block-allocation**] [**interface** [**ipv6**]] | **interface** [**ipv6**]} [**destination static** {*mapped_obj* | **interface** [**ipv6**]} *real_obj*] [**service** *mapped_dest_svc_obj real_dest_svc_obj*] [**unidirectional**] [**inactive**] [**description** *description*]

Where:

- Interfaces—(Required for bridge group member interfaces.) Specify the real (*real_ifc*) and mapped (*mapped_ifc*) interfaces. Be sure to include the parentheses. In routed mode, if you do not specify the real and mapped interfaces, all interfaces are used. You can also specify the keyword **any** for one or both of the interfaces, for example (any,outside), but **any** does not apply to bridge group member interfaces..
- Section and Line—(Optional.) By default, the NAT rule is added to the end of section 1 of the NAT table (see NAT rule order, on page 132). If you want to add the rule into section 3 instead (after the network object NAT rules), then use the **after-auto** keyword. You can insert a rule anywhere in the applicable section using the *line* argument.
- Source addresses:
  - Real—A network object, group, or the **any** keyword. Use the **any** keyword if you want to translate all traffic from the real interface to the mapped interface.
  - Mapped—Configure one of the following:
    - Network object—A network object that contains a host address.
    - **pat-pool** *mapped-obj*—A network object or group that contains multiple addresses.

- **interface** [**ipv6**]—(Routed mode only.) The IP address of the mapped interface is used as the mapped address (interface PAT). If you specify **ipv6**, then the IPv6 address of the interface is used. For this option, you must configure a specific interface for the *mapped_ifc*. (You cannot specify **interface** when the mapped interface is a bridge group member.) If you specify this keyword with a PAT pool or network object, you are enabling interface PAT fallback. After the PAT IP addresses are used up, then the IP address of the mapped interface is used.

For a PAT pool, you can specify one or more of the following options:

- **round-robin**—Enables round-robin address allocation for a PAT pool. Without round robin, by default all ports for a PAT address will be allocated before the next PAT address is used. The round-robin method assigns an address/port from each PAT address in the pool before returning to use the first address again, and then the second address, and so on.
- **extended**—Enables extended PAT. Extended PAT uses 65535 ports per *service*, as opposed to per IP address, by including the destination address and port in the translation information. Normally, the destination port and address are not considered when creating PAT translations, so you are limited to 65535 ports per PAT address. For example, with extended PAT, you can create a translation of 10.1.1.1:1027 when going to 192.168.1.7:23 as well as a translation of 10.1.1.1:1027 when going to 192.168.1.7:80.
- **include-reserve**—Includes the reserved ports, 1-1023, in the range of ports that are available for address translation. If you do not specify this option, addresses are translated to ports in the 1024-65535 range only.
- **block-allocation**—Enables port block allocation. For carrier-grade or large-scale PAT, you can allocate a block of ports for each host, rather than have NAT allocate one port translation at a time. If you allocate a block of ports, subsequent connections from the host use new randomly-selected ports within the block. If necessary, additional blocks are allocated if the host has active connections for all ports in the original block. Port blocks are allocated in the 1024-65535 range only. Port block allocation is compatible with **round-robin**, but you cannot use the **extended** option. You also cannot use interface PAT fallback.

- Destination addresses (Optional):
  - Mapped—Specify a network object or group, or for static interface NAT with port translation only (non-bridge group member interfaces only), specify the **interface** keyword. If you specify **ipv6**, then the IPv6 address of the interface is used. If you specify **interface**, be sure to also configure the **service** keyword. For this option, you must configure a specific interface for the *real_ifc*. See Static NAT with port translation, on page 161 for more information.
  - Real—Specify a network object or group. For identity NAT, simply use the same object or group for both the real and mapped addresses.
- Destination port—(Optional.) Specify the **service** keyword along with the mapped and real service objects. For identity port translation, simply use the same service object for both the real and mapped ports.
- Unidirectional—(Optional.) Specify **unidirectional** so the destination addresses cannot initiate traffic to the source addresses.
- Inactive—(Optional.) To make this rule inactive without having to remove the command, use the **inactive** keyword. To reactivate it, reenter the whole command without the **inactive** keyword.

• Description—(Optional.) Provide a description up to 200 characters using the **description** keyword.

Example:

```
hostname(config)# nat (inside,outside) source dynamic MyInsNet interface
destination static Server1 Server1
description Interface PAT for inside addresses when going to server 1
```

Examples

The following example configures interface PAT for inside network 192.168.1.0/24 when accessing outside Telnet server 209.165.201.23, and Dynamic PAT using a PAT pool when accessing any server on the 203.0.113.0/24 network.

```
hostname(config)# object network INSIDE_NW
hostname(config-network-object)# subnet 192.168.1.0 255.255.255.0

hostname(config)# object network PAT_POOL
hostname(config-network-object)# range 209.165.200.225 209.165.200.254

hostname(config)# object network TELNET_SVR
hostname(config-network-object)# host 209.165.201.23

hostname(config)# object service TELNET
hostname(config-service-object)# service tcp destination eq 23

hostname(config)# object network SERVERS
hostname(config-network-object)# subnet 203.0.113.0 255.255.255.0

hostname(config)# nat (inside,outside) source dynamic INSIDE_NW interface
destination static TELNET_SVR TELNET_SVR service TELNET TELNET
hostname(config)# nat (inside,outside) source dynamic INSIDE_NW pat-pool PAT_POOL
destination static SERVERS SERVERS
```

The following example configures interface PAT for inside network 192.168.1.0/24 when accessing outside IPv6 Telnet server 2001:DB8::23, and Dynamic PAT using a PAT pool when accessing any server on the 2001:DB8:AAAA::/96 network.

```
hostname(config)# object network INSIDE_NW
hostname(config-network-object)# subnet 192.168.1.0 255.255.255.0

hostname(config)# object network PAT_POOL
hostname(config-network-object)# range 2001:DB8:AAAA::1 2001:DB8:AAAA::200

hostname(config)# object network TELNET_SVR
hostname(config-network-object)# host 2001:DB8::23

hostname(config)# object service TELNET
hostname(config-service-object)# service tcp destination eq 23

hostname(config)# object network SERVERS
hostname(config-network-object)# subnet 2001:DB8:AAAA::/96

hostname(config)# nat (inside,outside) source dynamic INSIDE_NW interface ipv6
destination static TELNET_SVR TELNET_SVR service TELNET TELNET
```

```
hostname(config)# nat (inside,outside) source dynamic INSIDE_NW pat-pool PAT_POOL
destination static SERVERS SERVERS
```

Configure PAT with Port Block Allocation

For carrier-grade or large-scale PAT, you can allocate a block of ports for each host, rather than have NAT allocate one port translation at a time (see RFC 6888). If you allocate a block of ports, subsequent connections from the host use new randomly-selected ports within the block. If necessary, additional blocks are allocated if the host has active connections for all ports in the original block. Blocks are freed when the last xlate that uses a port in the block is removed.

The main reason for allocating port blocks is reduced logging. The port block allocation is logged, connections are logged, but xlates created within the port block are not logged. On the other hand, this makes log analysis more difficult.

Port blocks are allocated in the 1024-65535 range only. There are separate blocks for TCP, UDP, and ICMP connections, and these blocks can overlap. Thus, if an application requires a low port number (1-1023), it might not work. For example, an application requesting port 22 (SSH) will get a mapped port within the range of 1024-65535 and within the block allocated to the host. You can create a separate NAT rule that does not use block allocation for applications that use low port numbers; for twice NAT, ensure the rule comes before the block allocation rule.

Before you begin

Usage notes for NAT rules:

- You can include the **round-robin** keyword, but you cannot include **extended**, **include-reserve**, or **interface** (for interface PAT fallback). Other source/destination address and port information is also allowed.
- As with all NAT changes, if you replace an existing rule, you must clear xlates related to the replaced rule to have the new rule take effect. You can clear them explicitly or simply wait for them to time out. However, when operating in a cluster, you must reload the cluster so that the units can distribute blocks correctly.



Note If you are switching between a regular PAT and block allocation PAT rule, for object NAT, you must first delete the rule, then clear xlates. You can then create the new object NAT rule. Otherwise, you will see pat-port-block-state-mismatch drops in the **show asp drop** output.

- For a given PAT pool, you must specify (or not specify) block allocation for all rules that use the pool. You cannot allocate blocks in one rule and not in another. PAT pools that overlap also cannot mix block allocation settings. You also cannot overlap static NAT with port translation rules with the pool.

Procedure

Step 1 (Optional.) Configure the block allocation size, which is the number of ports in each block.

xlate block-allocation size *value*

The range is 32-4096. The default is 512. Use the "no" form to return to the default.

If you do not use the default, ensure that the size you choose divides evenly into 64,512 (the number of ports in the 1024-65535 range). Otherwise, there will be ports that cannot be used. For example, if you specify 100, there will be 12 unused ports.

Step 2 (Optional.) Configure the maximum blocks that can be allocated per host.

xlate block-allocation maximum-per-host *number*

The limit is per protocol, so a limit of 4 means at most 4 UDP blocks, 4 TCP blocks, and 4 ICMP blocks per host. The range is 1-8, the default is 4. Use the "no" form to return to the default.

Step 3 (Optional.) Enable interim syslog generation.

xlate block-allocation pba-interim-logging *seconds*

By default, the system generates syslog messages during port block creation and deletion. If you enable interim logging, the system generates the following message at the interval you specify. The messages report all active port blocks allocated at that time, including the protocol (ICMP, TCP, UDP) and source and destination interface and IP address, and the port block. You can specify an interval from 21600-604800 seconds (6 hours to 7 days).

%ASA-6-305017: Pba-interim-logging: Active *protocol* block of ports for translation from *real_interface*:*real_host_ip* to *mapped_interface*:*mapped_ip_address*/*start_port_num-end_port_num*

Example:

```
ciscoasa(config)# xlate block-allocation pba-interim-logging 21600
```

Step 4 Add NAT rules that use PAT pool block allocation.

- **Object PAT.**

  nat [(*real_ifc*,*mapped_ifc*)] **dynamic pat-pool** *mapped-obj* **block-allocation**

  Example:

```
object network mapped-pat-pool
   range 10.100.10.1 10.100.10.2
object network src_host
   host 10.111.10.15
object network src_host
   nat (inside,outside) dynamic
pat-pool mapped-pat-pool block-allocation
```

- **Twice PAT.**

  nat [(*real_ifc*,*mapped_ifc*)] [*line* | **after-auto** [*line*]] **source dynamic** *real_obj* **pat-pool***mapped-obj* **block-allocation**

  Example:

```
object network mapped-pat-pool
   range 10.100.10.1 10.100.10.2
object network src_network
   subnet 10.100.10.0 255.255.255.0
nat (inside,outside) 1 source dynamic src_network
```

```
pat-pool mapped-pat-pool block-allocation
```

Configure Per-Session PAT or Multi-Session PAT

By default, all TCP PAT traffic and all UDP DNS traffic uses per-session PAT. To use multi-session PAT for traffic, you can configure per-session PAT rules: a permit rule uses per-session PAT, and a deny rule uses multi-session PAT.

Per-session PAT improves the scalability of PAT and, for clustering, allows each member unit to own PAT connections; multi-session PAT connections have to be forwarded to and owned by the control unit. At the end of a per-session PAT session, the ASA sends a reset and immediately removes the xlate. This reset causes the end node to immediately release the connection, avoiding the TIME_WAIT state. Multi-session PAT, on the other hand, uses the PAT timeout, by default 30 seconds.

For "hit-and-run" traffic, such as HTTP or HTTPS, per-session PAT can dramatically increase the connection rate supported by one address. Without per-session PAT, the maximum connection rate for one address for an IP protocol is approximately 2000 per second. With per-session PAT, the connection rate for one address for an IP protocol is 65535/*average-lifetime*.

For traffic that can benefit from multi-session PAT, such as H.323, SIP, or Skinny, you can disable per-session PAT by creating a per-session deny rule. However, if you also want to use per-session PAT for the UDP ports used by these protocols, you must create the permit rules for them.

Before you begin

By default, the following rules are installed:

```
xlate per-session permit tcp any4 any4
xlate per-session permit tcp any4 any6
xlate per-session permit tcp any6 any4
xlate per-session permit tcp any6 any6
xlate per-session permit udp any4 any4 eq domain
xlate per-session permit udp any4 any6 eq domain
xlate per-session permit udp any6 any4 eq domain
xlate per-session permit udp any6 any6 eq domain
```

You cannot remove these rules, and they always exist after any manually-created rules. Because rules are evaluated in order, you can override the default rules. For example, to completely negate these rules, you could add the following:

```
xlate per-session deny tcp any4 any4
xlate per-session deny tcp any4 any6
xlate per-session deny tcp any6 any4
xlate per-session deny tcp any6 any6
xlate per-session deny udp any4 any4 eq domain
xlate per-session deny udp any4 any6 eq domain
xlate per-session deny udp any6 any4 eq domain
xlate per-session deny udp any6 any6 eq domain
```

Procedure

Create a permit or deny per-session PAT rule. This rule is placed above the default rules, but below any other manually-created rules. Be sure to create your rules in the order you want them applied.

xlate per-session {**permit** | **deny**} {**tcp** | **udp**} *source_ip* [*operator src_port*] *destination_ip* [*operator dest_port*]

For the source and destination IP addresses, you can configure the following:

- **host** *ip_address*—Specifies an IPv4 or IPv6 host address.
- *ip_address mask*—Specifies an IPv4 network address and subnet mask.
- *ipv6-address*/*prefix-length*—Specifies an IPv6 network address and prefix.
- **any4** and **any6**—**any4** specifies only IPv4 traffic; and **any6** specifies any6 traffic.

The *operator* matches the port numbers used by the source or destination. The default is all ports. The permitted operators are:

- **lt**—less than
- **gt**—greater than
- **eq**—equal to
- **neq**—not equal to
- **range**—an inclusive range of values. When you use this operator, specify two port numbers, for example, **range 100 200**.

Example

The following example creates a deny rule for H.323 traffic, so that it uses multi-session PAT:

```
hostname(config)# xlate per-session deny tcp any4 209.165.201.7 eq 1720
hostname(config)# xlate per-session deny udp any4 209.165.201.7 range 1718 1719
```

The following example enables the distribution of SIP across the members of a cluster by permitting per-session PAT for the SIP UDP port. Per-session PAT is the default for SIP TCP ports, so you do not need a rule for TCP unless you altered the default rules.

```
hostname(config)# xlate per-session permit udp any4 any4 eq sip
```

Static NAT

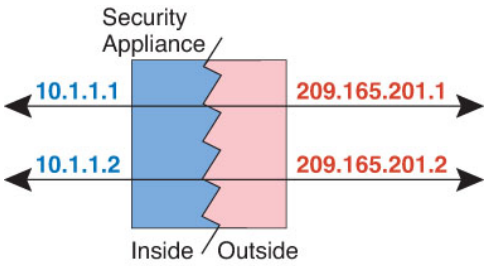
The following topics explain static NAT and how to implement it.

About static NAT

Static NAT creates a fixed translation of a real address to a mapped address. Because the mapped address is the same for each consecutive connection, static NAT allows bidirectional connection initiation, both to and from the host (if an access rule exists that allows it). With dynamic NAT and PAT, on the other hand, each host uses a different address or port for each subsequent translation, so bidirectional initiation is not supported.

The following figure shows a typical static NAT scenario. The translation is always active so both real and remote hosts can initiate connections.

Figure 11: Static NAT



Note You can disable bidirectionality if desired.

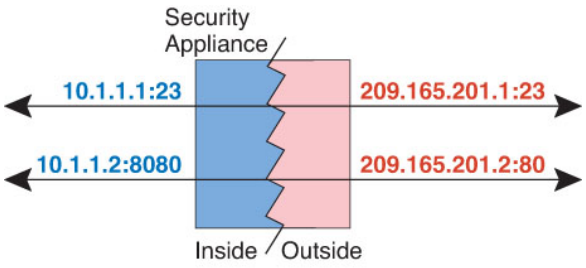
Static NAT with port translation

Static NAT with port translation lets you specify a real and mapped protocol and port.

When you specify the port with static NAT, you can choose to map the port and/or the IP address to the same value or to a different value.

The following figure shows a typical static NAT with port translation scenario showing both a port that is mapped to itself and a port that is mapped to a different value; the IP address is mapped to a different value in both cases. The translation is always active so both translated and remote hosts can initiate connections.

Figure 12: Typical Static NAT with Port Translation Scenario



Static NAT-with-port-translation rules limit access to the destination IP address for the specified port only. If you try to access the destination IP address on a different port not covered by a NAT rule, then the connection is blocked. In addition, for twice NAT, traffic that does not match the source IP address of the NAT rule will be dropped if it matches the destination IP address, regardless of the destination port. Therefore, you must add additional rules for all other traffic allowed to the destination IP address. For example, you can configure a static NAT rule for the IP address, without port specification, and place it after the port translation rule.

Note For applications that require application inspection for secondary channels (for example, FTP and VoIP), NAT automatically translates the secondary ports.

Following are some other uses of static NAT with port translation.

Static NAT with Identity Port Translation

You can simplify external access to internal resources. For example, if you have three separate servers that provide services on different ports (such as FTP, HTTP, and SMTP), you can give external users a single IP address to access those services. You can then configure static NAT with identity port translation to map the single external IP address to the correct IP addresses of the real servers based on the port they are trying to access. You do not need to change the port, because the servers are using the standard ones (21, 80, and 25 respectively). For details on how to configure this example, see Single Address for FTP, HTTP, and SMTP (Static NAT-with-Port-Translation), on page 185.

Static NAT with Port Translation for Non-Standard Ports

You can also use static NAT with port translation to translate a well-known port to a non-standard port or vice versa. For example, if inside web servers use port 8080, you can allow outside users to connect to port 80, and then undo translation to the original port 8080. Similarly, to provide extra security, you can tell web users to connect to non-standard port 6785, and then undo translation to port 80.

Static Interface NAT with Port Translation

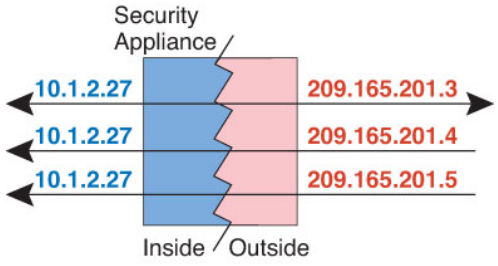
You can configure static NAT to map a real address to an interface address/port combination. For example, if you want to redirect Telnet access for the device's outside interface to an inside host, then you can map the inside host IP address/port 23 to the outside interface address/port 23.

One-to-many static NAT

Typically, you configure static NAT with a one-to-one mapping. However, in some cases, you might want to configure a single real address to several mapped addresses (one-to-many). When you configure one-to-many static NAT, when the real host initiates traffic, it always uses the first mapped address. However, for traffic initiated to the host, you can initiate traffic to any of the mapped addresses, and they will be untranslated to the single real address.

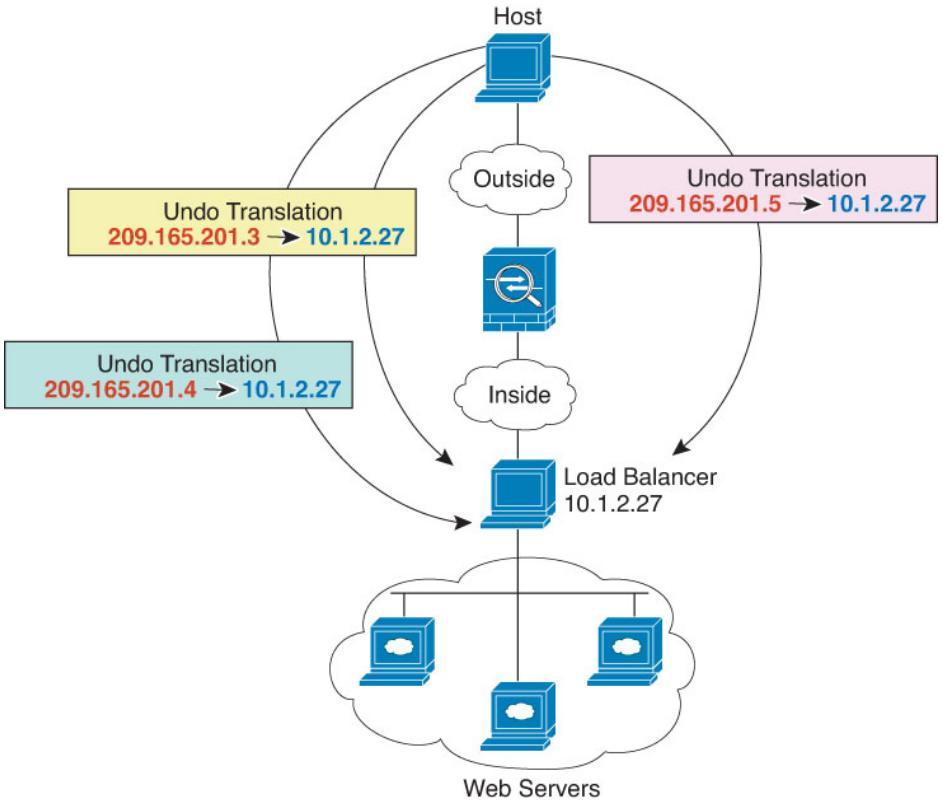
The following figure shows a typical one-to-many static NAT scenario. Because initiation by the real host always uses the first mapped address, the translation of real host IP/first mapped IP is technically the only bidirectional translation.

Figure 13: One-to-Many Static NAT



For example, you have a load balancer at 10.1.2.27. Depending on the URL requested, it redirects traffic to the correct web server. For details on how to configure this example, see Inside Load Balancer with Multiple Mapped Addresses (Static NAT, One-to-Many), on page 184.

Figure 14: One-to-Many Static NAT Example



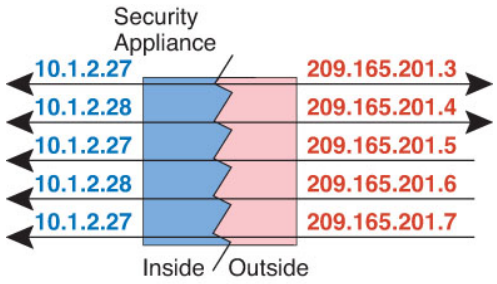
Other mapping scenarios (not recommended)

NAT has the flexibility to allow any kind of static mapping scenario: one-to-one, one-to-many, but also few-to-many, many-to-few, and many-to-one mappings. We recommend using only one-to-one or one-to-many mappings. These other mapping options might result in unintended consequences.

Functionally, few-to-many is the same as one-to-many; but because the configuration is more complicated and the actual mappings may not be obvious at a glance, we recommend creating a one-to-many configuration for each real address that requires it. For example, for a few-to-many scenario, the few real addresses are mapped to the many mapped addresses in order (A to 1, B to 2, C to 3). When all real addresses are mapped, the next mapped address is mapped to the first real address, and so on until all mapped addresses are mapped (A to 4, B to 5, C to 6). This results in multiple mapped addresses for each real address. Just like a one-to-many configuration, only the first mappings are bidirectional; subsequent mappings allow traffic to be initiated *to* the real host, but all traffic *from* the real host uses only the first mapped address for the source.

The following figure shows a typical few-to-many static NAT scenario.

Figure 15: Few-to-Many Static NAT

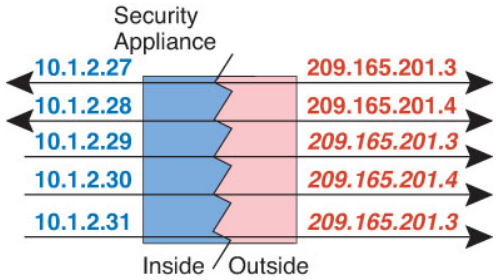


For a many-to-few or many-to-one configuration, where you have more real addresses than mapped addresses, you run out of mapped addresses before you run out of real addresses. Only the mappings between the lowest real IP addresses and the mapped pool result in bidirectional initiation. The remaining higher real addresses can initiate traffic, but traffic cannot be initiated to them (returning traffic for a connection is directed to the correct real address because of the unique 5-tuple (source IP, destination IP, source port, destination port, protocol) for the connection).

Note Many-to-few or many-to-one NAT is not PAT. If two real hosts use the same source port number and go to the same outside server and the same TCP destination port, and both hosts are translated to the same IP address, then both connections will be reset because of an address conflict (the 5-tuple is not unique).

The following figure shows a typical many-to-few static NAT scenario.

Figure 16: Many-to-Few Static NAT



Instead of using a static rule this way, we suggest that you create a one-to-one rule for the traffic that needs bidirectional initiation, and then create a dynamic rule for the rest of your addresses.

Configure Static Network Object NAT or Static NAT-with-Port-Translation

This section describes how to configure a static NAT rule using network object NAT.

Procedure

Step 1 (Optional.) Create a network object (**object network** command), or a network object group (**object-group network** command), for the mapped addresses.

- Instead of using an object, you can configure an inline address or specify the interface address (for static NAT-with-port-translation).
- If you use an object, the object or group can contain a host, range, or subnet.

Step 2 Create or edit the network object for which you want to configure NAT: **object network** *obj_name*

Example:

```
hostname(config)# object network my-host-obj1
```

Step 3 (Skip when editing an object that has the right address.) Define the real IPv4 or IPv6 addresses that you want to translate.

- **host** {*IPv4_address* | *IPv6_address*}—The IPv4 or IPv6 address of a single host. For example, 10.1.1.1 or 2001:DB8::0DB8:800:200C:417A.
- **subnet** {*IPv4_address IPv4_mask* | *IPv6_address/IPv6_prefix*}—The address of a network. For IPv4 subnets, include the mask after a space, for example, 10.0.0.0 255.0.0.0. For IPv6, include the address and prefix as a single unit (no spaces), such as 2001:DB8:0:CD30::/60.
- **range** *start_address end_address*—A range of addresses. You can specify IPv4 or IPv6 ranges. Do not include masks or prefixes.

Example:

```
hostname(config-network-object)# subnet 10.2.1.0 255.255.255.0
```

Step 4 Configure **static NAT** for the object IP addresses. You can only define a single NAT rule for a given object.

nat [**(***real_ifc***,***mapped_ifc***)**] **static** {*mapped_inline_host_ip* | *mapped_obj* | **interface** [**ipv6**]} [**net-to-net**] [**dns** | **service** {**tcp** | **udp** | **sctp**} *real_port mapped_port*] [**no-proxy-arp**]

Where:

- Interfaces—(Required for bridge group member interfaces.) Specify the real (*real_ifc*) and mapped (*mapped_ifc*) interfaces. Be sure to include the parentheses. In routed mode, if you do not specify the real and mapped interfaces, all interfaces are used. You can also specify the keyword **any** for one or both of the interfaces, for example (any,outside), but **any** does not apply to bridge group member interfaces..
- Mapped IP address—You can specify the mapped IP address as one of the following. Typically, you configure the same number of mapped addresses as real addresses for a one-to-one mapping. You can, however, have a mismatched number of addresses. See Static NAT, on page 160.
  - *mapped_inline_host_ip*—An inline host IP address. This provides a one-to-one mapping for host objects. For subnet objects, the same netmask is used for the inline host address, and you get one-to-one translations for addresses in the mapped inline host's subnet. For range objects, the mapped address includes the same number of hosts that are in the range object, starting with the mapped host address. For example, if the real address is defined as a range from 10.1.1.1 through 10.1.1.6, and you specify 172.20.1.1 as the mapped address, then the mapped range will include 172.20.1.1 through 172.20.1.6. For NAT46 or NAT66 translations, this can be an IPv6 network address.
  - *mapped_obj*—An existing network object or group. To do a one-to-one mapping for a range of IP addresses, select an object that contains a range with the same number of addresses.

- **interface**—(Static NAT-with-port-translation only.) The IP address of the mapped interface is used as the mapped address. If you specify **ipv6**, then the IPv6 address of the interface is used. For this option, you must configure a specific interface for the *mapped_ifc*. (You cannot specify **interface** when the mapped interface is a bridge group member.) You must use this keyword when you want to use the interface IP address; you cannot enter it inline or as an object. Be sure to also configure the **service** keyword.

- Net-to-net—(Optional.) For NAT 46, specify **net-to-net** to translate the first IPv4 address to the first IPv6 address, the second to the second, and so on. Without this option, the IPv4-embedded method is used. For a one-to-one translation, you must use this keyword.
- DNS—(Optional.) The **dns** keyword translates DNS replies. Be sure DNS inspection is enabled (it is enabled by default). See Rewriting DNS queries and responses using NAT, on page 207 for more information.
- Port translation—(Static NAT-with-port-translation only.) Specify **service** with the desired protocol keyword and the real and mapped ports. You can enter either a port number or a well-known port name (such as **http**).
- No Proxy ARP—(Optional.) Specify **no-proxy-arp** to disable proxy ARP for incoming packets to the mapped IP addresses. For information on the conditions which might require the disabling of proxy ARP, see Mapped Addresses and Routing, on page 192.

Example:

```
hostname(config-network-object)#
nat (inside,outside) static MAPPED_IPS service tcp 80 8080
```

Examples

The following example configures static NAT for the real host 10.1.1.1 on the inside to 10.2.2.2 on the outside with DNS rewrite enabled.

```
hostname(config)# object network my-host-obj1
hostname(config-network-object)# host 10.1.1.1
hostname(config-network-object)# nat (inside,outside) static 10.2.2.2 dns
```

The following example configures static NAT for the real host 10.1.1.1 on the inside to 10.2.2.2 on the outside using a mapped object.

```
hostname(config)# object network my-mapped-obj
hostname(config-network-object)# host 10.2.2.2

hostname(config-network-object)# object network my-host-obj1
hostname(config-network-object)# host 10.1.1.1
hostname(config-network-object)# nat (inside,outside) static my-mapped-obj
```

The following example configures static NAT-with-port-translation for 10.1.1.1 at TCP port 21 to the outside interface at port 2121.

```
hostname(config)# object network my-ftp-server
hostname(config-network-object)# host 10.1.1.1
hostname(config-network-object)# nat (inside,outside) static interface service tcp 21 2121
```

The following example maps an inside IPv4 network to an outside IPv6 network.

```
hostname(config)# object network inside_v4_v6
hostname(config-network-object)# subnet 10.1.1.0 255.255.255.0
hostname(config-network-object)# nat (inside,outside) static 2001:DB8::/96
```

The following example maps an inside IPv6 network to an outside IPv6 network.

```
hostname(config)# object network inside_v6
hostname(config-network-object)# subnet 2001:DB8:AAAA::/96
hostname(config-network-object)# nat (inside,outside) static 2001:DB8:BBBB::/96
```

Configure Static Twice NAT or Static NAT-with-Port-Translation

This section describes how to configure a static NAT rule using twice NAT.

Procedure

Step 1 Create host or range network objects (**object network** command), or network object groups (**object-group network** command), for the source real addresses, the source mapped addresses, the destination real addresses, and the destination mapped addresses. You can also use an FQDN network object for the destination mapped address.

- If you want to configure source static interface NAT with port translation only, you can skip adding an object for the source mapped addresses, and instead specify the **interface** keyword in the **nat** command.
- If you want to configure destination static interface NAT with port translation only, you can skip adding an object for the destination mapped addresses, and instead specify the **interface** keyword in the **nat** command.

If you do create objects, consider the following guidelines:

- The mapped object or group can contain a host, range, or subnet.
- The static mapping is typically one-to-one, so the real addresses have the same quantity as the mapped addresses. You can, however, have different quantities if desired. For more information, see Static NAT, on page 160.

Step 2 (Optional.) Create service objects for the:

- Source *or* Destination real ports
- Source *or* Destination mapped ports

A service object can contain both a source and destination port; however, you should specify *either* the source *or* the destination port for both service objects. You should only specify *both* the source and destination ports

if your application uses a fixed source port (such as some DNS servers); but fixed source ports are rare. For example, if you want to translate the port for the source host, then configure the source service.

Step 3 Configure **static NAT**.

nat [**(***real_ifc***,***mapped_ifc***)**] [*line* | {**after-object** [*line*]}] **source static** *real_ob* [*mapped_obj* | **interface** [**ipv6**]] [**destination static** {*mapped_obj* | **interface** [**ipv6**]} *real_obj*] [**service** *real_src_mapped_dest_svc_obj mapped_src_real_dest_svc_obj*] [**net-to-net**] [**dns**] [**unidirectional** | **no-proxy-arp**] [**inactive**] [**description** *desc*]

Where:

- Interfaces—(Required for bridge group member interfaces.) Specify the real (*real_ifc*) and mapped (*mapped_ifc*) interfaces. Be sure to include the parentheses. In routed mode, if you do not specify the real and mapped interfaces, all interfaces are used. You can also specify the keyword **any** for one or both of the interfaces, for example (any,outside), but **any** does not apply to bridge group member interfaces.
- Section and Line—(Optional.) By default, the NAT rule is added to the end of section 1 of the NAT table (see NAT rule order, on page 132). If you want to add the rule into section 3 instead (after the network object NAT rules), then use the **after-auto** keyword. You can insert a rule anywhere in the applicable section using the *line* argument.
- Source addresses:
  - Real—Specify a network object or group. Do not use the **any** keyword, which would be used for identity NAT.
  - Mapped—Specify a different network object or group. For static interface NAT with port translation only, you can specify the **interface** keyword. If you specify **ipv6**, then the IPv6 address of the interface is used. If you specify **interface**, be sure to also configure the **service** keyword (in this case, the service objects should include only the source port). For this option, you must configure a specific interface for the *mapped_ifc*. (You cannot specify **interface** when the mapped interface is a bridge group member.) See Static NAT with port translation, on page 161 for more information.
- Destination addresses (Optional):
  - Mapped—Specify a network object or group, or for static interface NAT with port translation only, specify the **interface** keyword. If you specify **ipv6**, then the IPv6 address of the interface is used. If you specify **interface**, be sure to also configure the **service** keyword (in this case, the service objects should include only the destination port). For this option, you must configure a specific interface for the *real_ifc*. (You cannot specify **interface** when the mapped interface is a bridge group member.)
  - Real—Specify a network object or group. For identity NAT, simply use the same object or group for both the real and mapped addresses.
- Ports—(Optional.) Specify the **service** keyword along with the real and mapped service objects. For source port translation, the objects must specify the source service. The order of the service objects in the command for source port translation is **service** *real_obj mapped_obj*. For destination port translation, the objects must specify the destination service. The order of the service objects for destination port translation is **service** *mapped_obj real_obj*. In the rare case where you specify both the source and destination ports in the object, the first service object contains the real source port/mapped destination port; the second service object contains the mapped source port/real destination port. For identity port translation, simply use the same service object for both the real and mapped ports (source and/or destination ports, depending on your configuration).

- Net-to-net—(Optional.) For NAT 46, specify **net-to-net** to translate the first IPv4 address to the first IPv6 address, the second to the second, and so on. Without this option, the IPv4-embedded method is used. For a one-to-one translation, you must use this keyword.
- DNS—(Optional; for a source-only rule.) The **dns** keyword translates DNS replies. Be sure DNS inspection is enabled (it is enabled by default). You cannot configure the **dns** keyword if you configure a **destination** address. See Rewriting DNS queries and responses using NAT, on page 207 for more information.
- Unidirectional—(Optional.) Specify **unidirectional** so the destination addresses cannot initiate traffic to the source addresses.
- No Proxy ARP—(Optional.) Specify **no-proxy-arp** to disable proxy ARP for incoming packets to the mapped IP addresses. See Mapped Addresses and Routing, on page 192 for more information.
- Inactive—(Optional.) To make this rule inactive without having to remove the command, use the **inactive** keyword. To reactivate it, reenter the whole command without the **inactive** keyword.
- Description—(Optional.) Provide a description up to 200 characters using the **description** keyword.

Example:

```
hostname(config)# nat (inside,dmz) source static MyInsNet MyInsNet_mapped
destination static Server1 Server1 service REAL_SRC_SVC MAPPED_SRC_SVC
```

Examples

The following example shows the use of static interface NAT with port translation. Hosts on the outside access an FTP server on the inside by connecting to the outside interface IP address with destination port 65000 through 65004. The traffic is untranslated to the internal FTP server at 192.168.10.100:6500 through 65004. Note that you specify the source port range in the service object (and not the destination port) because you want to translate the source address and port as identified in the command; the destination port is “any.” Because static NAT is bidirectional, “source” and “destination” refers primarily to the command keywords; the actual source and destination address and port in a packet depends on which host sent the packet. In this example, connections are originated from outside to inside, so the “source” address and port of the FTP server is actually the destination address and port in the originating packet.

```
hostname(config)# object service FTP_PASV_PORT_RANGE
hostname(config-service-object)# service tcp source range 65000 65004

hostname(config)# object network HOST_FTP_SERVER
hostname(config-network-object)# host 192.168.10.100

hostname(config)# nat (inside,outside) source static HOST_FTP_SERVER interface
service FTP_PASV_PORT_RANGE FTP_PASV_PORT_RANGE
```

The following example shows a static translation of one IPv6 network to another IPv6 when accessing an IPv6 network, and the dynamic PAT translation to an IPv4 PAT pool when accessing the IPv4 network:

```
hostname(config)# object network INSIDE_NW
```

```
hostname(config-network-object)# subnet 2001:DB8:AAAA::/96

hostname(config)# object network MAPPED_IPv6_NW
hostname(config-network-object)# subnet 2001:DB8:BBBB::/96

hostname(config)# object network OUTSIDE_IPv6_NW
hostname(config-network-object)# subnet 2001:DB8:CCCC::/96

hostname(config)# object network OUTSIDE_IPv4_NW
hostname(config-network-object)# subnet 10.1.1.0 255.255.255.0

hostname(config)# object network MAPPED_IPv4_POOL
hostname(config-network-object)# range 10.1.2.1 10.1.2.254

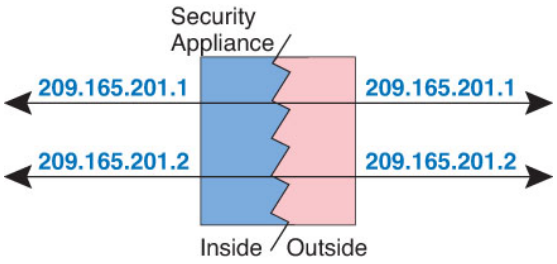
hostname(config)# nat (inside,outside) source static INSIDE_NW MAPPED_IPv6_NW
destination static OUTSIDE_IPv6_NW OUTSIDE_IPv6_NW
hostname(config)# nat (inside,outside) source dynamic INSIDE_NW pat-pool MAPPED_IPv4_POOL
destination static OUTSIDE_IPv4_NW OUTSIDE_IPv4_NW
```

Identity NAT

You might have a NAT configuration in which you need to translate an IP address to itself. For example, if you create a broad rule that applies NAT to every network, but want to exclude one network from NAT, you can create a static NAT rule to translate that network to itself. Identity NAT is necessary for remote access VPN, where you need to exempt the client traffic from NAT.

The following figure shows a typical identity NAT scenario.

Figure 17: Identity NAT



The following topics explain how to configure identity NAT.

Configure Identity Network Object NAT

This section describes how to configure an identity NAT rule using network object NAT.

Procedure

Step 1 (Optional.) Create a network object (**object network** command), or a network object group (**object-group network** command), for the mapped addresses.

- Instead of using an object, you can configure an inline address.

- If you use an object, the object must match the real addresses you want to translate.

Step 2 Create or edit the network object for which you want to configure NAT: **object network** *obj_name*

The object must be a different one than what you use for the mapped addresses, even though the contents must be the same in each object.

Example:

```
hostname(config)# object network my-host-obj1
```

Step 3 (Skip when editing an object that has the right address.) Define the real IPv4 or IPv6 addresses that you want to translate.

- **host** {*IPv4_address* | *IPv6_address*}—The IPv4 or IPv6 address of a single host. For example, 10.1.1.1 or 2001:DB8::0DB8:800:200C:417A.
- **subnet** {*IPv4_address IPv4_mask* | *IPv6_address/IPv6_prefix*}—The address of a network. For IPv4 subnets, include the mask after a space, for example, 10.0.0.0 255.0.0.0. For IPv6, include the address and prefix as a single unit (no spaces), such as 2001:DB8:0:CD30::/60.
- **range** *start_address end_address*—A range of addresses. You can specify IPv4 or IPv6 ranges. Do not include masks or prefixes.

Example:

```
hostname(config-network-object)# subnet 10.2.1.0 255.255.255.0
```

Step 4 Configure **identity NAT** for the object IP addresses. You can only define a single NAT rule for a given object.

nat [**(***real_ifc***,***mapped_ifc***)**] **static** {*mapped_inline_host_ip* | *mapped_obj*} [**no-proxy-arp**] [**route-lookup**]

Where:

- Interfaces—(Required for bridge group member interfaces.) Specify the real (*real_ifc*) and mapped (*mapped_ifc*) interfaces. Be sure to include the parentheses. In routed mode, if you do not specify the real and mapped interfaces, all interfaces are used. You can also specify the keyword **any** for one or both of the interfaces, for example (any,outside), but **any** does not apply to bridge group member interfaces.
- Mapped IP addresses—Be sure to configure the same IP address for both the mapped and real address. Use one of the following:
  - *mapped_inline_host_ip*—An inline host IP address. For host objects, specify the same address. For range objects, specify the first address in the real range (the same number of addresses in the range will be used). For subnet objects, specify any address within the real subnet (all addresses in the subnet will be used).
  - *mapped_obj*—A network object or group that includes the same addresses as the real object.
- No Proxy ARP—(Optional.) Specify **no-proxy-arp** to disable proxy ARP for incoming packets to the mapped IP addresses. For information on the conditions which might require the disabling of proxy ARP, see Mapped Addresses and Routing, on page 192.

- Route lookup—(Routed mode only; interfaces specified.) Specify **route-lookup** to determine the egress interface using a route lookup instead of using the interface specified in the NAT command. See Determining the Egress Interface, on page 194 for more information.

Example:

```
hostname(config-network-object)# nat (inside,outside) static MAPPED_IPS
```

Example

The following example maps a host address to itself using an inline mapped address:

```
hostname(config)# object network my-host-obj1
hostname(config-network-object)# host 10.1.1.1
hostname(config-network-object)# nat (inside,outside) static 10.1.1.1
```

The following example maps a host address to itself using a network object:

```
hostname(config)# object network my-host-obj1-identity
hostname(config-network-object)# host 10.1.1.1

hostname(config-network-object)# object network my-host-obj1
hostname(config-network-object)# host 10.1.1.1
hostname(config-network-object)# nat (inside,outside) static my-host-obj1-identity
```

Configure Identity Twice NAT

This section describes how to configure an identity NAT rule using twice NAT.

Procedure

Step 1 Create host or range network objects (**object network** command), or network object groups (**object-group network** command), for the source real addresses (you will typically use the same object for the source mapped addresses), the destination real addresses, and the destination mapped addresses. You can also use an FQDN network object for the destination mapped address.

- If you want to perform identity NAT for all addresses, you can skip creating an object for the source real addresses and instead use the keywords **any any** in the **nat** command.
- If you want to configure destination static interface NAT with port translation only, you can skip adding an object for the destination mapped addresses, and instead specify the **interface** keyword in the **nat** command.

If you do create objects, consider the following guidelines:

- The mapped object or group can contain a host, range, or subnet.

- The real and mapped source objects must match. You can use the same object for both, or you can create separate objects that contain the same IP addresses.

Step 2 (Optional.) Create service objects for the:

- Source *or* Destination real ports
- Source *or* Destination mapped ports

A service object can contain both a source and destination port; however, you should specify *either* the source *or* the destination port for both service objects. You should only specify *both* the source and destination ports if your application uses a fixed source port (such as some DNS servers); but fixed source ports are rare. For example, if you want to translate the port for the source host, then configure the source service.

Step 3 Configure **identity NAT**.

nat [**(***real_ifc***,***mapped_ifc***)**] [*line* | {**after-object** [*line*]}] **source static** {*nw_obj nw_obj* | **any any**} [**destination static** {*mapped_obj* | **interface** [**ipv6**]} *real_obj*] [**service** *real_src_mapped_dest_svc_obj mapped_src_real_dest_svc_obj*] [**no-proxy-arp**] [**route-lookup**] [**inactive**] [**description** *desc*]

Where:

- Interfaces—(Required for bridge group member interfaces.) Specify the real (*real_ifc*) and mapped (*mapped_ifc*) interfaces. Be sure to include the parentheses. In routed mode, if you do not specify the real and mapped interfaces, all interfaces are used. You can also specify the keyword **any** for one or both of the interfaces, for example (any,outside), but **any** does not apply to bridge group member interfaces.
- Section and Line—(Optional.) By default, the NAT rule is added to the end of section 1 of the NAT table (see NAT rule order, on page 132). If you want to add the rule into section 3 instead (after the network object NAT rules), then use the **after-auto** keyword. You can insert a rule anywhere in the applicable section using the *line* argument.
- Source addresses—Specify a network object, group, or the **any** keyword for both the real and mapped addresses.
- Destination addresses (Optional):
  - Mapped—Specify a network object or group, or for static interface NAT with port translation only, specify the **interface** keyword. If you specify **ipv6**, then the IPv6 address of the interface is used. If you specify **interface**, be sure to also configure the **service** keyword (in this case, the service objects should include only the destination port). For this option, you must configure a specific interface for the *real_ifc*. (You cannot specify **interface** when the real interface is a bridge group member).
  - Real—Specify a network object or group. For identity NAT, simply use the same object or group for both the real and mapped addresses.
- Ports—(Optional.) Specify the **service** keyword along with the real and mapped service objects. For source port translation, the objects must specify the source service. The order of the service objects in the command for source port translation is **service** *real_obj mapped_obj*. For destination port translation, the objects must specify the destination service. The order of the service objects for destination port translation is **service** *mapped_obj real_obj*. In the rare case where you specify both the source and destination ports in the object, the first service object contains the real source port/mapped destination port; the second service object contains the mapped source port/real destination port. For identity port

translation, simply use the same service object for both the real and mapped ports (source and/or destination ports, depending on your configuration).

- No Proxy ARP—(Optional.) Specify **no-proxy-arp** to disable proxy ARP for incoming packets to the mapped IP addresses. See Mapped Addresses and Routing, on page 192 for more information.
- Route lookup—(Optional; routed mode only; interfaces specified.) Specify **route-lookup** to determine the egress interface using a route lookup instead of using the interface specified in the NAT command. See Determining the Egress Interface, on page 194 for more information.
- Inactive—(Optional.) To make this rule inactive without having to remove the command, use the **inactive** keyword. To reactivate it, reenter the whole command without the **inactive** keyword.
- Description—Optional.) Provide a description up to 200 characters using the **description** keyword.

Example:

```
hostname(config)# nat (inside,outside) source static MyInsNet MyInsNet
destination static Server1 Server1
```

Monitoring NAT

To monitor NAT, use the following commands:

- **show nat**

  Shows NAT statistics, including hits for each NAT rule.

- **show nat pool**

  Shows NAT pool statistics, including the addresses and ports allocated, and how many times they were allocated.

- **show running-config nat**

  Shows the NAT configuration. You cannot see object NAT rules using **show running-config object**. When you use show running-config without modifiers, objects that include NAT rules are shown twice, first with the basic address configuration, then later in the configuration, the object with the NAT rule. The complete object, with the address and NAT rule, is not shown as a unit.

- **show xlate**

  Shows current NAT session information.

History for NAT

Feature Name	Platform Releases	Description
Network Object NAT	8.3(1)	Configures NAT for a network object IP address(es). We introduced or modified the following commands: **nat** (object network configuration mode), **show nat**, **show xlate**, **show nat pool**.
Twice NAT	8.3(1)	Twice NAT lets you identify both the source and destination address in a single rule. We modified or introduced the following commands: **nat**, **show nat**, **show xlate**, **show nat pool**.
Identity NAT configurable proxy ARP and route lookup	8.4(2)/8.5(1)	In earlier releases for identity NAT, proxy ARP was disabled, and a route lookup was always used to determine the egress interface. You could not configure these settings. In 8.4(2) and later, the default behavior for identity NAT was changed to match the behavior of other static NAT configurations: proxy ARP is enabled, and the NAT configuration determines the egress interface (if specified) by default. You can leave these settings as is, or you can enable or disable them discretely. Note that you can now also disable proxy ARP for regular static NAT. For pre-8.3 configurations, the migration of NAT exempt rules (the **nat 0 access-list** command) to 8.4(2) and later now includes the following keywords to disable proxy ARP and to use a route lookup: **no-proxy-arp** and **route-lookup**. The **unidirectional** keyword that was used for migrating to 8.3(2) and 8.4(1) is no longer used for migration. When upgrading to 8.4(2) from 8.3(1), 8.3(2), and 8.4(1), all identity NAT configurations will now include the **no-proxy-arp** and **route-lookup** keywords, to maintain existing functionality. The **unidirectional** keyword is removed. We modified the following command: **nat static** [**no-proxy-arp**] [**route-lookup**].

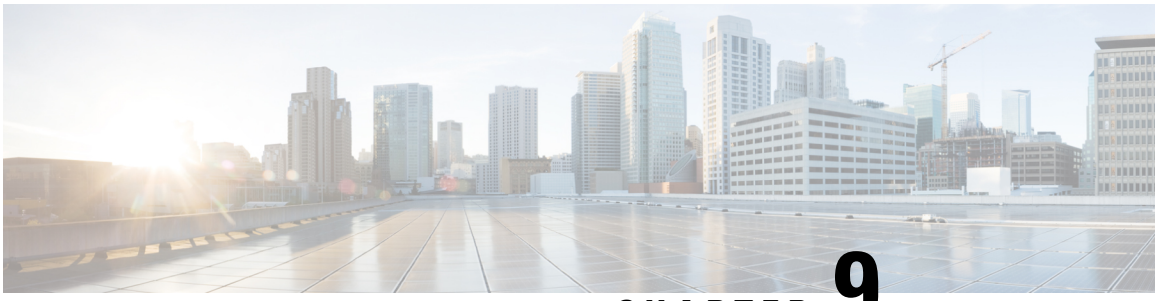
Feature Name	Platform Releases	Description
PAT pool and round robin address assignment	8.4(2)/8.5(1)	You can now specify a pool of PAT addresses instead of a single address. You can also optionally enable round-robin assignment of PAT addresses instead of first using all ports on a PAT address before using the next address in the pool. These features help prevent a large number of connections from a single PAT address from appearing to be part of a DoS attack and makes configuration of large numbers of PAT addresses easy. We modifed the following commands: **nat dynamic** [**pat-pool** *mapped_object* [**round-robin**]] and **nat source dynamic** [**pat-pool** *mapped_object* [**round-robin**]].
Round robin PAT pool allocation uses the same IP address for existing hosts	8.4(3)	When using a PAT pool with round robin allocation, if a host has an existing connection, then subsequent connections from that host will use the same PAT IP address if ports are available. We did not modify any commands. *This feature is not available in 8.5(1) or 8.6(1).*
Flat range of PAT ports for a PAT pool	8.4(3)	If available, the real source port number is used for the mapped port. However, if the real port is *not* available, by default the mapped ports are chosen from the same range of ports as the real port number: 0 to 511, 512 to 1023, and 1024 to 65535. Therefore, ports below 1024 have only a small PAT pool. If you have a lot of traffic that uses the lower port ranges, when using a PAT pool, you can now specify a flat range of ports to be used instead of the three unequal-sized tiers: either 1024 to 65535, or 1 to 65535. We modifed the following commands: **nat dynamic** [**pat-pool** *mapped_object* [**flat** [**include-reserve**]]] and **nat source dynamic** [**pat-pool** *mapped_object* [**flat** [**include-reserve**]]]. *This feature is not available in 8.5(1) or 8.6(1).*
Extended PAT for a PAT pool	8.4(3)	Each PAT IP address allows up to 65535 ports. If 65535 ports do not provide enough translations, you can now enable extended PAT for a PAT pool. Extended PAT uses 65535 ports per *service*, as opposed to per IP address, by including the destination address and port in the translation information. We modifed the following command: **nat dynamic** [**pat-pool** *mapped_object* [**extended**]] and **nat source dynamic** [**pat-pool** *mapped_object* [**extended**]]. *This feature is not available in 8.5(1) or 8.6(1).*

Feature Name	Platform Releases	Description
Automatic NAT rules to translate a VPN peer's local IP address back to the peer's real IP address	8.4(3)	In rare situations, you might want to use a VPN peer's real IP address on the inside network instead of an assigned local IP address. Normally with VPN, the peer is given an assigned local IP address to access the inside network. However, you might want to translate the local IP address back to the peer's real public IP address if, for example, your inside servers and network security is based on the peer's real IP address. You can enable this feature on one interface per tunnel group. Object NAT rules are dynamically added and deleted when the VPN session is established or disconnected. You can view the rules using the **show nat** command. Because of routing issues, we do not recommend using this feature unless you know you need it; contact Cisco TAC to confirm feature compatibility with your network. See the following limitations: • Only supports Cisco IPsec and Secure Client. • Return traffic to the public IP addresses must be routed back to the ASA so the NAT policy and VPN policy can be applied. • Does not support load-balancing (because of routing issues). • Does not support roaming (public IP changing). We introduced the following command: **nat-assigned-to-public-ip** *interface* (tunnel-group general-attributes configuration mode).
NAT support for IPv6	9.0(1)	NAT now supports IPv6 traffic, as well as translating between IPv4 and IPv6. Translating between IPv4 and IPv6 is not supported in transparent mode. We modified the following commands: **nat** (global and object network configuration modes), **show nat**, **show nat pool**, **show xlate**.
NAT support for reverse DNS lookups	9.0(1)	NAT now supports translation of the DNS PTR record for reverse DNS lookups when using IPv4 NAT, IPv6 NAT, and NAT64 with DNS inspection enabled for the NAT rule.

Feature Name	Platform Releases	Description
Per-session PAT	9.0(1)	The per-session PAT feature improves the scalability of PAT and, for clustering, allows each member unit to own PAT connections; multi-session PAT connections have to be forwarded to and owned by the control unit. At the end of a per-session PAT session, the ASA sends a reset and immediately removes the xlate. This reset causes the end node to immediately release the connection, avoiding the TIME_WAIT state. Multi-session PAT, on the other hand, uses the PAT timeout, by default 30 seconds. For "hit-and-run" traffic, such as HTTP or HTTPS, the per-session feature can dramatically increase the connection rate supported by one address. Without the per-session feature, the maximum connection rate for one address for an IP protocol is approximately 2000 per second. With the per-session feature, the connection rate for one address for an IP protocol is 65535/*average-lifetime*. By default, all TCP traffic and UDP DNS traffic use a per-session PAT xlate. For traffic that requires multi-session PAT, such as H.323, SIP, or Skinny, you can disable per-session PAT by creating a per-session deny rule. We introduced the following commands: **xlate per-session**, **show nat pool**.
Transactional Commit Model on NAT Rule Engine	9.3(1)	When enabled, a NAT rule update is applied after the rule compilation is completed; without affecting the rule matching performance. We added the **nat** keyword to the following commands: **asp rule-engine transactional-commit**, **show running-config asp rule-engine transactional-commit**, **clear configure asp rule-engine transactional-commit**. We added NAT to the following screen: Configuration > Device Management > Advanced > Rule Engine.
Carrier Grade NAT enhancements	9.5(1)	For carrier-grade or large-scale PAT, you can allocate a block of ports for each host, rather than have NAT allocate one port translation at a time (see RFC 6888). We added the following commands: **xlate block-allocation size**, **xlate block-allocation maximum-per-host**. We added the **block-allocation** keyword to the **nat** command.
NAT support for SCTP	9.5(2)	You can now specify SCTP ports in static network object NAT rules. Using SCTP in static twice NAT is not recommended. Dynamic NAT/PAT does not support SCTP. We modified the following commands: **nat static** (object).

Feature Name	Platform Releases	Description
Interim logging for NAT port block allocation.	9.12(1)	When you enable port block allocation for NAT, the system generates syslog messages during port block creation and deletion. If you enable interim logging, the system generates message 305017 at the interval you specify. The messages report all active port blocks allocated at that time, including the protocol (ICMP, TCP, UDP) and source and destination interface and IP address, and the port block. We added the following command: **xlate block-allocation pba-interim-logging** *seconds*.
Changes to PAT address allocation in clustering. The PAT pool **flat** option is now enabled by default and it is not configurable.	9.15(1)	The way PAT addresses are distributed to the members of a cluster is changed. Previously, addresses were distributed to the members of the cluster, so your PAT pool would need a minimum of one address per cluster member. Now, the control unit instead divides each PAT pool address into equal-sized port blocks and distributes them across cluster members. Each member has port blocks for the same PAT addresses. Thus, you can reduce the size of the PAT pool, even to as few as one IP address, depending on the amount of connections you typically need to PAT. Port blocks are allocated in 512-port blocks from the 1024-65535 range. You can optionally included the reserved ports, 1-1023, in this block allocation when you configure PAT pool rules. For example, in a 4-node cluster, each node gets 32 blocks with which it will be able to handle 16384 connections per PAT pool IP address compared to a single node handling all 65535 connections per PAT pool IP address. As part of this change, PAT pools for all systems, whether standalone or operating in a cluster, now use a flat port range of 1023 - 65535. Previously, you could optionally use a flat range by including the **flat** keyword in a PAT pool rule. The **flat** keyword is no longer supported: the PAT pool is now always flat. The **include-reserve** keyword, which was previously a sub-keyword to **flat**, is now an independent keyword within the PAT pool configuration. With this option, you can include the 1 - 1023 port range within the PAT pool. Note that if you configure port block allocation (the **block-allocation** PAT pool option), your block allocation size is used rather than the default 512-port block. In addition, you cannot configure extended PAT for a PAT pool for systems in a cluster. New/Modified commands: **nat**, **show nat pool**

Feature Name	Platform Releases	Description
New Section 0 for system-defined NAT rules.	9.16(1)	A new Section 0 has been added to the NAT rule table. This section is exclusively for the use of the system. Any NAT rules that the system needs for normal functioning are added to this section, and these rules take priority over any rules you create. Previously, system-defined rules were added to Section 1, and user-defined rules could interfere with proper system functioning. You cannot add, edit, or delete Section 0 rules, but you will see them in **show nat detail** command output.
Twice NAT support for fully-qualified domain name (FQDN) objects as the translated (mapped) destination.	9.17(1)	You can use an FQDN network object, such as one specifying www.example.com, as the translated (mapped) destination address in twice NAT rules. The system configures the rule based on the IP address returned from the DNS server.



CHAPTER 9

NAT Examples and Reference

The following topics provide examples for configuring NAT, plus information on advanced configuration and troubleshooting.

- Examples for Network Object NAT, on page 181
- Examples for Twice NAT, on page 186
- NAT in routed and transparent mode, on page 190
- Routing NAT Packets, on page 192
- NAT for VPN, on page 195
- Translating IPv6 networks, on page 201
- Rewriting DNS queries and responses using NAT, on page 207

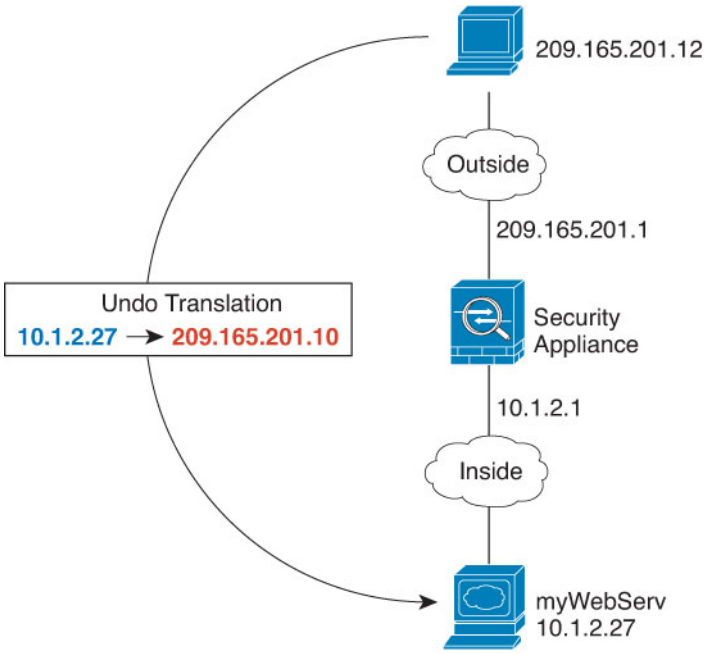
Examples for Network Object NAT

Following are some configuration examples for network object NAT.

Providing Access to an Inside Web Server (Static NAT)

The following example performs static NAT for an inside web server. The real address is on a private network, so a public address is required. Static NAT is necessary so hosts can initiate traffic to the web server at a fixed address.

Figure 18: Static NAT for an Inside Web Server



Procedure

Step 1 Create a network object for the internal web server.

```
hostname(config)# object network myWebServ
hostname(config-network-object)# host 10.1.2.27
```

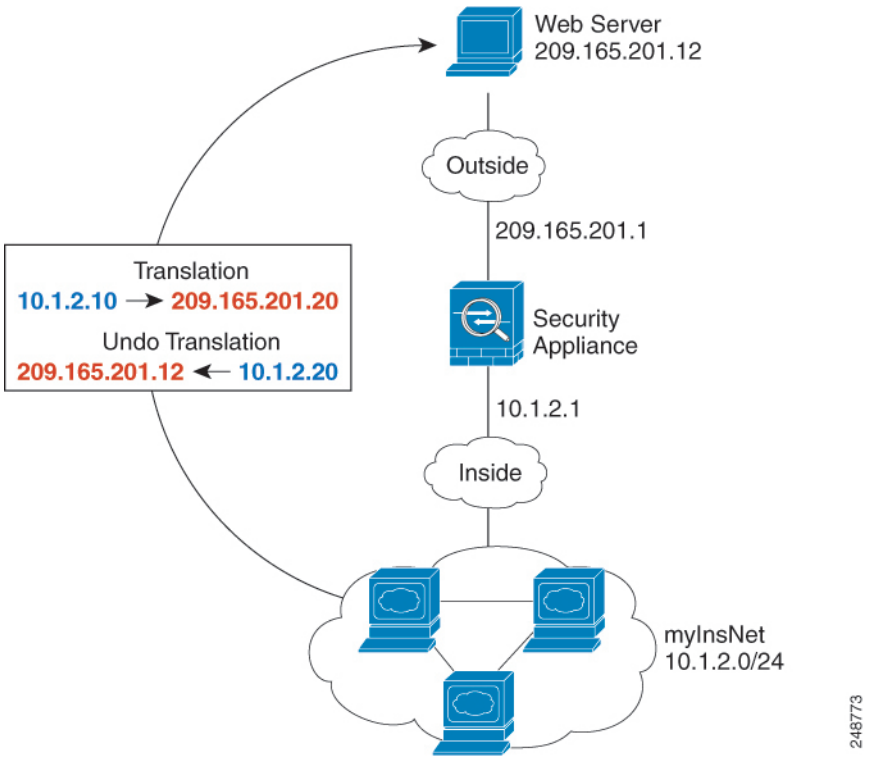
Step 2 Configure static NAT for the object:

```
hostname(config-network-object)# nat (inside,outside) static 209.165.201.10
```

NAT for Inside Hosts (Dynamic NAT) and NAT for an Outside Web Server (Static NAT)

The following example configures dynamic NAT for inside users on a private network when they access the outside. Also, when inside users connect to an outside web server, that web server address is translated to an address that appears to be on the inside network.

Figure 19: Dynamic NAT for Inside, Static NAT for Outside Web Server



Procedure

Step 1 Create a network object for the dynamic NAT pool to which you want to translate the inside addresses.

```
hostname(config)# object network myNatPool
hostname(config-network-object)# range 209.165.201.20 209.165.201.30
```

Step 2 Create a network object for the inside network.

```
hostname(config)# object network myInsNet
hostname(config-network-object)# subnet 10.1.2.0 255.255.255.0
```

Step 3 Enable dynamic NAT for the inside network using the dynamic NAT pool object.

```
hostname(config-network-object)# nat (inside,outside) dynamic myNatPool
```

Step 4 Create a network object for the outside web server.

```
hostname(config)# object network myWebServ
hostname(config-network-object)# host 209.165.201.12
```

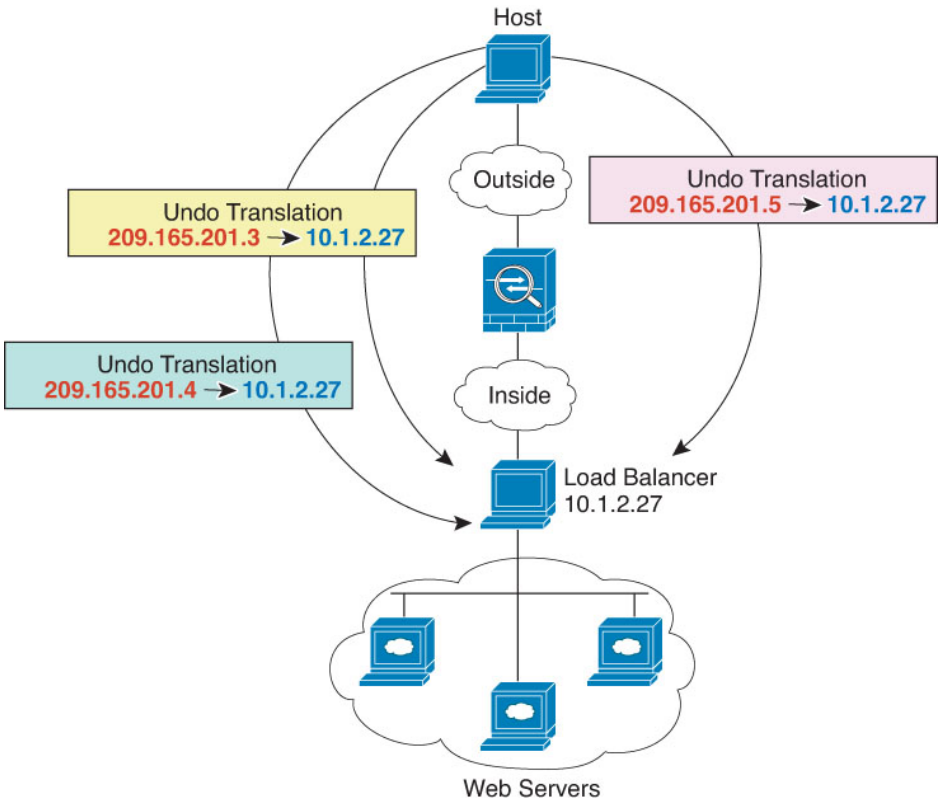
Step 5 Configure static NAT for the web server.

```
hostname(config-network-object)# nat (outside,inside) static 10.1.2.20
```

Inside Load Balancer with Multiple Mapped Addresses (Static NAT, One-to-Many)

The following example shows an inside load balancer that is translated to multiple IP addresses. When an outside host accesses one of the mapped IP addresses, it is untranslated to the single load balancer address. Depending on the URL requested, it redirects traffic to the correct web server.

Figure 20: Static NAT with One-to-Many for an Inside Load Balancer



Procedure

Step 1 Create a network object for the addresses to which you want to map the load balancer.

```
hostname(config)# object network myPublicIPs
hostname(config-network-object)# range 209.165.201.3 209.265.201.8
```

Step 2 Create a network object for the load balancer.

```
hostname(config)# object network myLBHost
hostname(config-network-object)# host 10.1.2.27
```

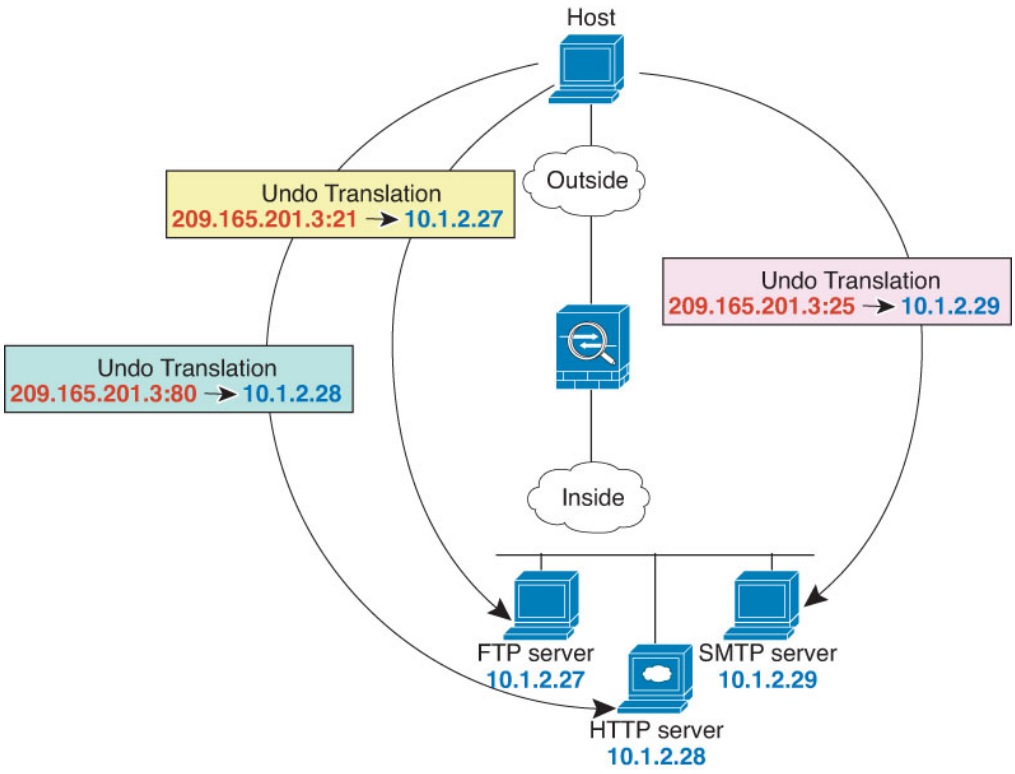
Step 3 Configure static NAT for the load balancer applying the range object.

```
hostname(config-network-object)# nat (inside,outside) static myPublicIPs
```

Single Address for FTP, HTTP, and SMTP (Static NAT-with-Port-Translation)

The following static NAT-with-port-translation example provides a single address for remote users to access FTP, HTTP, and SMTP. These servers are actually different devices on the real network, but for each server, you can specify static NAT-with-port-translation rules that use the same mapped IP address, but different ports.

Figure 21: Static NAT-with-Port-Translation



Procedure

Step 1 Create a network object for the FTP server and configure static NAT with port translation, mapping the FTP port to itself.

```
hostname(config)# object network FTP_SERVER
hostname(config-network-object)# host 10.1.2.27
hostname(config-network-object)# nat (inside,outside) static 209.165.201.3 service tcp ftp
 ftp
```

Step 2 Create a network object for the HTTP server and configure static NAT with port translation, mapping the HTTP port to itself.

```
hostname(config)# object network HTTP_SERVER
hostname(config-network-object)# host 10.1.2.28
hostname(config-network-object)# nat (inside,outside) static 209.165.201.3 service tcp http
 http
```

Step 3 Create a network object for the SMTP server and configure static NAT with port translation, mapping the SMTP port to itself.

```
hostname(config)# object network SMTP_SERVER
hostname(config-network-object)# host 10.1.2.29
hostname(config-network-object)# nat (inside,outside) static 209.165.201.3 service tcp smtp
 smtp
```

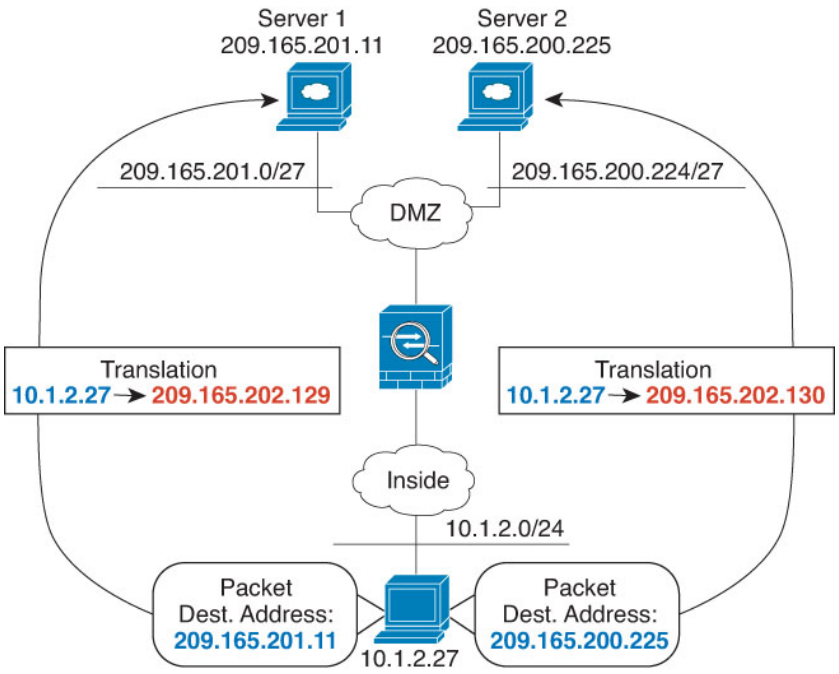
Examples for Twice NAT

This section includes the following configuration examples:

Different Translation Depending on the Destination (Dynamic Twice PAT)

The following figure shows a host on the 10.1.2.0/24 network accessing two different servers. When the host accesses the server at 209.165.201.11, the real address is translated to 209.165.202.129:*port*. When the host accesses the server at 209.165.200.225, the real address is translated to 209.165.202.130:*port*.

Figure 22: Twice NAT with Different Destination Addresses



Procedure

Step 1 Add a network object for the inside network:

```
hostname(config)# object network myInsideNetwork
hostname(config-network-object)# subnet 10.1.2.0 255.255.255.0
```

Step 2 Add a network object for the DMZ network 1:

```
hostname(config)# object network DMZnetwork1
hostname(config-network-object)# subnet 209.165.201.0 255.255.255.224
```

Step 3 Add a network object for the PAT address:

```
hostname(config)# object network PATaddress1
hostname(config-network-object)# host 209.165.202.129
```

Step 4 Configure the first twice NAT rule:

```
hostname(config)# nat (inside,dmz) source dynamic myInsideNetwork PATaddress1
destination static DMZnetwork1 DMZnetwork1
```

Because you do not want to translate the destination address, you need to configure identity NAT for it by specifying the same address for the real and mapped destination addresses.

Step 5 Add a network object for the DMZ network 2:

```
hostname(config)# object network DMZnetwork2
hostname(config-network-object)# subnet 209.165.200.224 255.255.255.224
```

Step 6 Add a network object for the PAT address:

```
hostname(config)# object network PATaddress2
hostname(config-network-object)# host 209.165.202.130
```

Step 7 Configure the second twice NAT rule:

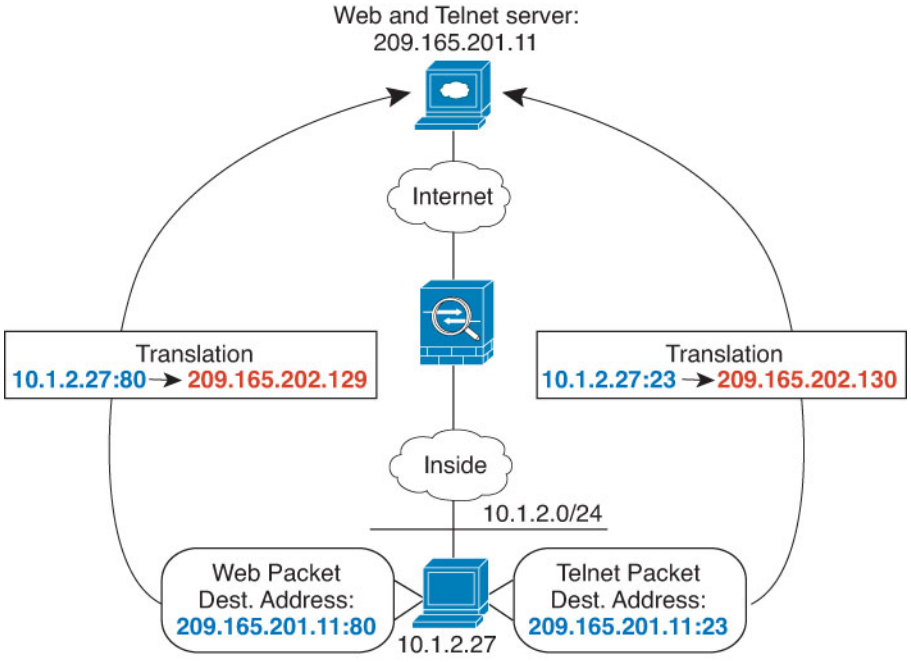
Example:

```
hostname(config)# nat (inside,dmz) source dynamic myInsideNetwork PATaddress2
destination static DMZnetwork2 DMZnetwork2
```

Different Translation Depending on the Destination Address and Port (Dynamic PAT)

The following figure shows the use of source and destination ports. The host on the 10.1.2.0/24 network accesses a single host for both web services and Telnet services. When the host accesses the server for Telnet services, the real address is translated to 209.165.202.129:*port*. When the host accesses the same server for web services, the real address is translated to 209.165.202.130:*port*.

Figure 23: Twice NAT with Different Destination Ports



Procedure

Step 1 Add a network object for the inside network:

```
hostname(config)# object network myInsideNetwork
hostname(config-network-object)# subnet 10.1.2.0 255.255.255.0
```

Step 2 Add a network object for the Telnet/Web server:

```
hostname(config)# object network TelnetWebServer
hostname(config-network-object)# host 209.165.201.11
```

Step 3 Add a network object for the PAT address when using Telnet:

```
hostname(config)# object network PATaddress1
hostname(config-network-object)# host 209.165.202.129
```

Step 4 Add a service object for Telnet:

```
hostname(config)# object service TelnetObj
hostname(config-network-object)# service tcp destination eq telnet
```

Step 5 Configure the first twice NAT rule:

```
hostname(config)# nat (inside,outside) source dynamic myInsideNetwork PATaddress1
destination static TelnetWebServer TelnetWebServer service TelnetObj TelnetObj
```

Because you do not want to translate the destination address or port, you need to configure identity NAT for them by specifying the same address for the real and mapped destination addresses, and the same port for the real and mapped service.

Step 6 Add a network object for the PAT address when using HTTP:

```
hostname(config)# object network PATaddress2
hostname(config-network-object)# host 209.165.202.130
```

Step 7 Add a service object for HTTP:

```
hostname(config)# object service HTTPObj
hostname(config-network-object)# service tcp destination eq http
```

Step 8 Configure the second twice NAT rule:

```
hostname(config)# nat (inside,outside) source dynamic myInsideNetwork PATaddress2
```

```
destination static TelnetWebServer TelnetWebServer service HTTPObj HTTPObj
```

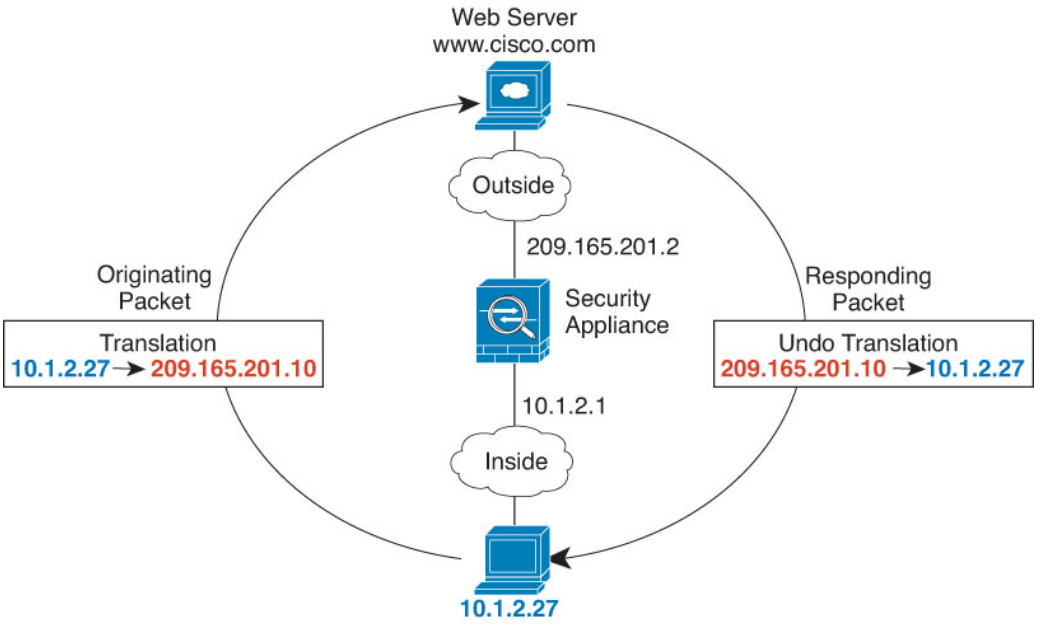
NAT in routed and transparent mode

You can configure NAT in both routed and transparent firewall mode. The following sections describe typical usage for each firewall mode.

NAT in routed mode

The following figure shows a typical NAT example in routed mode, with a private network on the inside.

Figure 24: NAT Example: Routed Mode



1. When the inside host at 10.1.2.27 sends a packet to a web server, the real source address of the packet, 10.1.2.27, is translated to a mapped address, 209.165.201.10.
2. When the server responds, it sends the response to the mapped address, 209.165.201.10, and the ASA receives the packet because the ASA performs proxy ARP to claim the packet.
3. The ASA then changes the translation of the mapped address, 209.165.201.10, back to the real address, 10.1.2.27, before sending it to the host.

NAT in transparent mode or within a bridge group

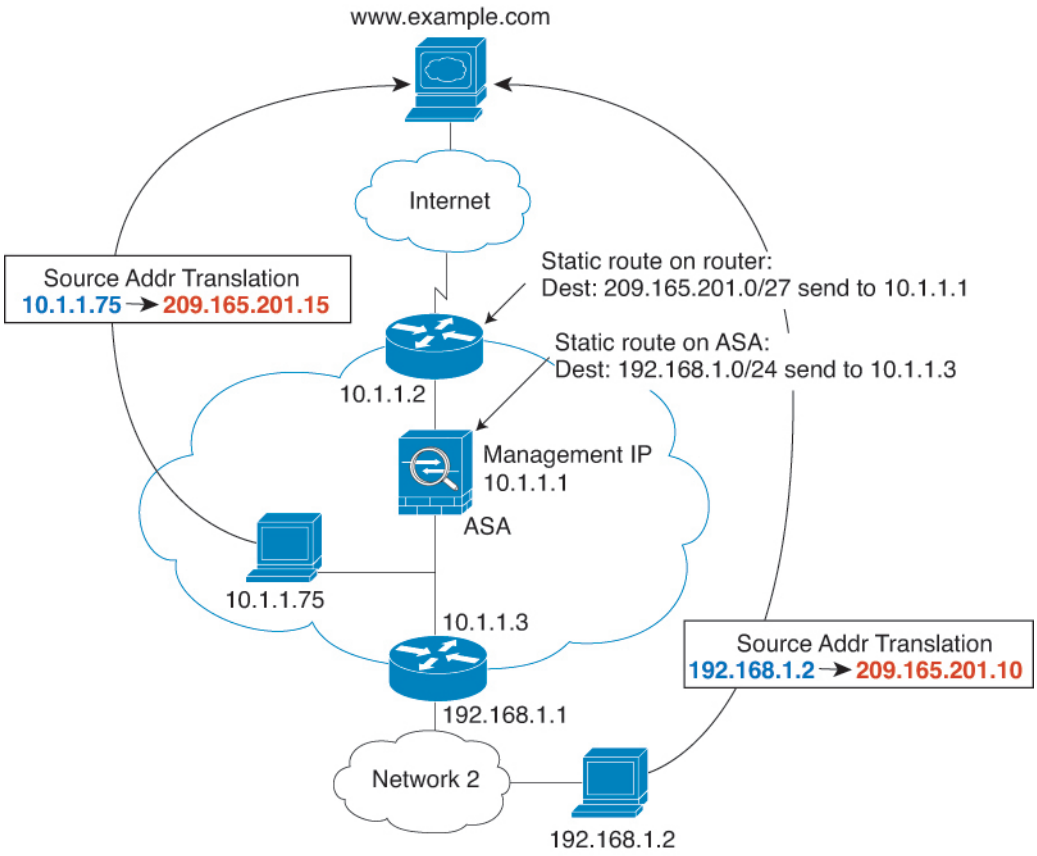
Using NAT in transparent mode eliminates the need for the upstream or downstream routers to perform NAT for their networks. It can perform a similar function within a bridge group in routed mode.

NAT in transparent mode, or in routed mode between members of the same bridge group, has the following requirements and limitations:

- You cannot configure interface PAT when the mapped address is a bridge group member interface, because there is no IP address attached to the interface.
- ARP inspection is not supported. Moreover, if for some reason a host on one side of the ASA sends an ARP request to a host on the other side of the ASA, and the initiating host real address is mapped to a different address on the same subnet, then the real address remains visible in the ARP request.
- Translating between IPv4 and IPv6 networks is not supported. Translating between two IPv6 networks, or between two IPv4 networks is supported.

The following figure shows a typical NAT scenario in transparent mode, with the same network on the inside and outside interfaces. The transparent firewall in this scenario is performing the NAT service so that the upstream router does not have to perform NAT.

Figure 25: NAT Example: Transparent Mode



1. When the inside host at 10.1.1.75 sends a packet to a web server, the real source address of the packet, 10.1.1.75, is changed to a mapped address, 209.165.201.15.
2. When the server responds, it sends the response to the mapped address, 209.165.201.15, and the ASA receives the packet because the upstream router includes this mapped network in a static route directed to the ASA management IP address.

3. The ASA then undoes the translation of the mapped address, 209.165.201.15, back to the real address, 10.1.1.1.75. Because the real address is directly-connected, the ASA sends it directly to the host.
4. For host 192.168.1.2, the same process occurs, except for returning traffic, the ASA looks up the route in its routing table and sends the packet to the downstream router at 10.1.1.3 based on the ASA static route for 192.168.1.0/24.

Routing NAT Packets

The ASA needs to be the destination for any packets sent to the mapped address. The ASA also needs to determine the egress interface for any packets it receives destined for mapped addresses. This section describes how the ASA handles accepting and delivering packets with NAT.

Mapped Addresses and Routing

When you translate the real address to a mapped address, the mapped address you choose determines how to configure routing, if necessary, for the mapped address.

See additional guidelines about mapped IP addresses in Additional guidelines for NAT, on page 136.

The following topics explain the mapped address types.

Addresses on the same network as the mapped interface

If you use addresses on the same network as the destination (mapped) interface, the ASA uses proxy ARP to answer any ARP requests for the mapped addresses, thus intercepting traffic destined for a mapped address. This solution simplifies routing because the ASA does not have to be the gateway for any additional networks. This solution is ideal if the outside network contains an adequate number of free addresses, a consideration if you are using a 1:1 translation like dynamic NAT or static NAT. Dynamic PAT greatly extends the number of translations you can use with a small number of addresses, so even if the available addresses on the outside network is small, this method can be used. For PAT, you can even use the IP address of the mapped interface.



Note If you configure the mapped interface to be any interface, and you specify a mapped address on the same network as one of the mapped interfaces, then if an ARP request for that mapped address comes in on a *different* interface, then you need to manually configure an ARP entry for that network on the ingress interface, specifying its MAC address. Typically, if you specify any interface for the mapped interface, then you use a unique network for the mapped addresses, so this situation would not occur. Configure ARP using the **arp** command.

Addresses on a unique network

If you need more addresses than are available on the destination (mapped) interface network, you can identify addresses on a different subnet. The upstream router needs a static route for the mapped addresses that points to the ASA.

Alternatively for routed mode, you can configure a static route on the ASA for the mapped addresses using any IP address on the destination network as the gateway, and then redistribute the route using your routing protocol. For example, if you use NAT for the inside network (10.1.1.0/24) and use the mapped IP address

209.165.201.5, then you can configure a static route for 209.165.201.5 255.255.255.255 (host address) to the 10.1.1.99 gateway that can be redistributed.

```
route inside 209.165.201.5 255.255.255.255 10.1.1.99
```

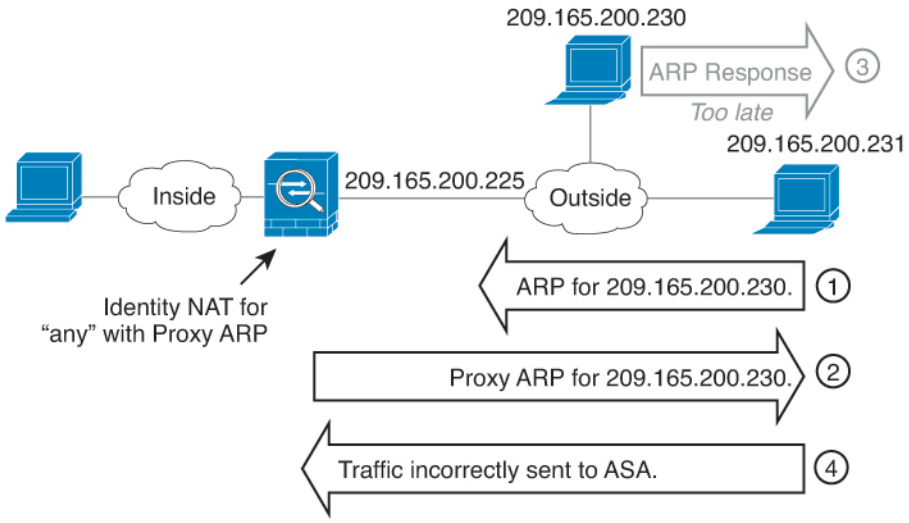
For transparent mode, if the real host is directly-connected, configure the static route on the upstream router to point to the ASA: specify the bridge group IP address. For remote hosts in transparent mode, in the static route on the upstream router, you can alternatively specify the downstream router IP address.

The same address as the real address (identity NAT)

The default behavior for identity NAT has proxy ARP enabled, matching other static NAT rules. You can disable proxy ARP if desired. You can also disable proxy ARP for regular static NAT if desired, in which case you need to be sure to have proper routes on the upstream router.

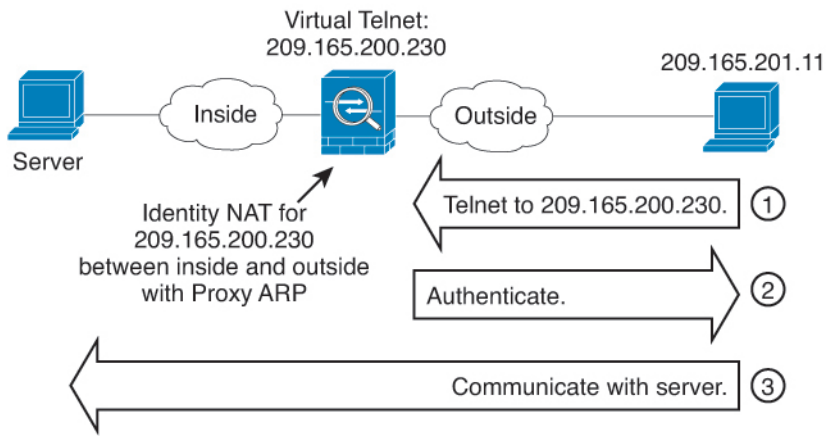
Normally for identity NAT, proxy ARP is not required, and in some cases can cause connectivity issues. For example, if you configure a broad identity NAT rule for "any" IP address, then leaving proxy ARP enabled can cause problems for hosts on the network directly connected to the mapped interface. In this case, when a host on the mapped network wants to communicate with another host on the same network, then the address in the ARP request matches the NAT rule (which matches "any" address). The ASA will then proxy ARP for the address, even though the packet is not actually destined for the ASA. (Note that this problem occurs even if you have a twice NAT rule; although the NAT rule must match both the source and destination addresses, the proxy ARP decision is made only on the "source" address). If the ASA ARP response is received before the actual host ARP response, then traffic will be mistakenly sent to the ASA.

Figure 26: Proxy ARP Problems with Identity NAT



In rare cases, you need proxy ARP for identity NAT; for example for virtual Telnet. When using AAA for network access, a host needs to authenticate with the ASA using a service like Telnet before any other traffic can pass. You can configure a virtual Telnet server on the ASA to provide the necessary login. When accessing the virtual Telnet address from the outside, you must configure an identity NAT rule for the address specifically for the proxy ARP functionality. Due to internal processes for virtual Telnet, proxy ARP lets the ASA keep traffic destined for the virtual Telnet address rather than send the traffic out the source interface according to the NAT rule. (See the following figure).

Figure 27: Proxy ARP and Virtual Telnet



Transparent Mode Routing Requirements for Remote Networks

When you use NAT in transparent mode, some types of traffic require static routes. See the general operations configuration guide for more information.

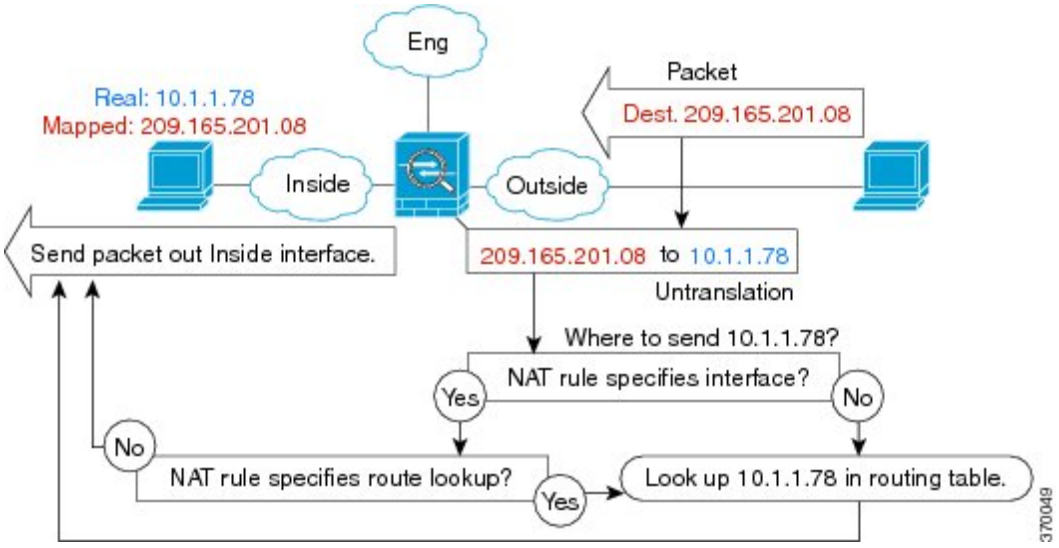
Determining the Egress Interface

When you use NAT and the ASA receives traffic for a mapped address, then the ASA untranslates the destination address according to the NAT rule, and then it sends the packet on to the real address. The ASA determines the egress interface for the packet in the following ways:

- Bridge group interfaces in Transparent mode or Routed Mode—The ASA determines the egress interface for the real address by using the NAT rule; you must specify the source and destination bridge group member interfaces as part of the NAT rule.
- Regular interfaces in Routed mode—The ASA determines the egress interface in one of the following ways:
  - You configure the interface in the NAT rule—The ASA uses the NAT rule to determine the egress interface. However, you have the option to always use a route lookup instead. In certain scenarios, a route lookup override is required.
  - You do not configure the interface in the NAT rule—The ASA uses a route lookup to determine the egress interface.

The following figure shows the egress interface selection method in routed mode. In almost all cases, a route lookup is equivalent to the NAT rule interface, but in some configurations, the two methods might differ.

Figure 28: Routed Mode Egress Interface Selection with NAT



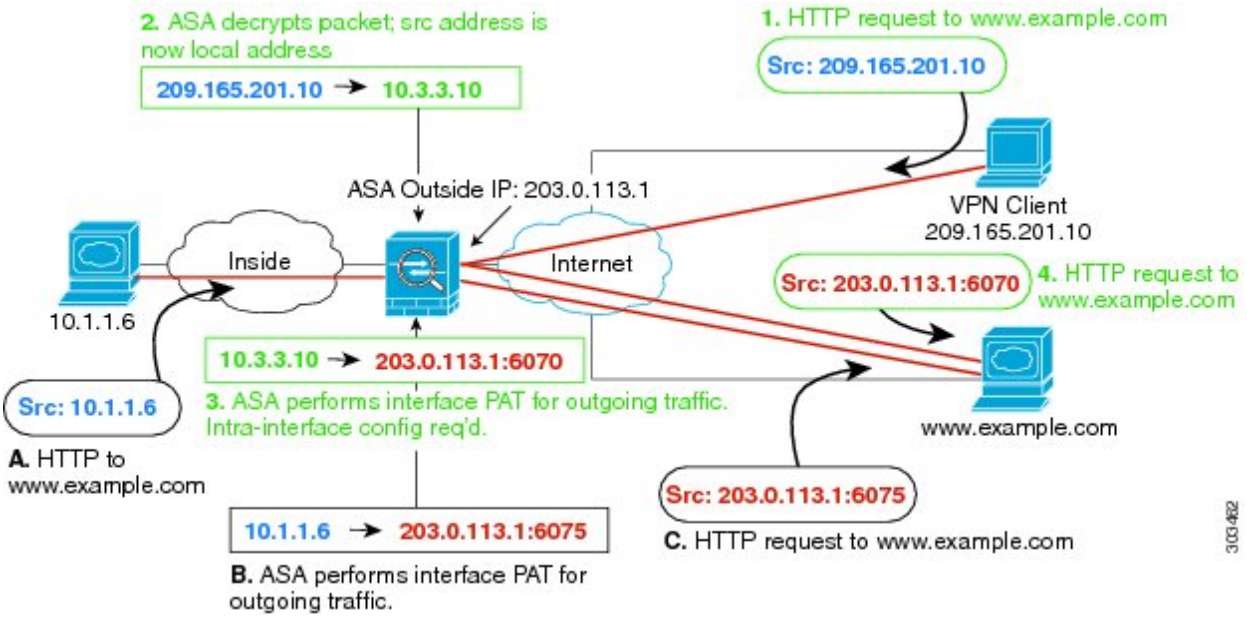
NAT for VPN

The following topics explain NAT usage with the various types of VPN.

NAT and Remote Access VPN

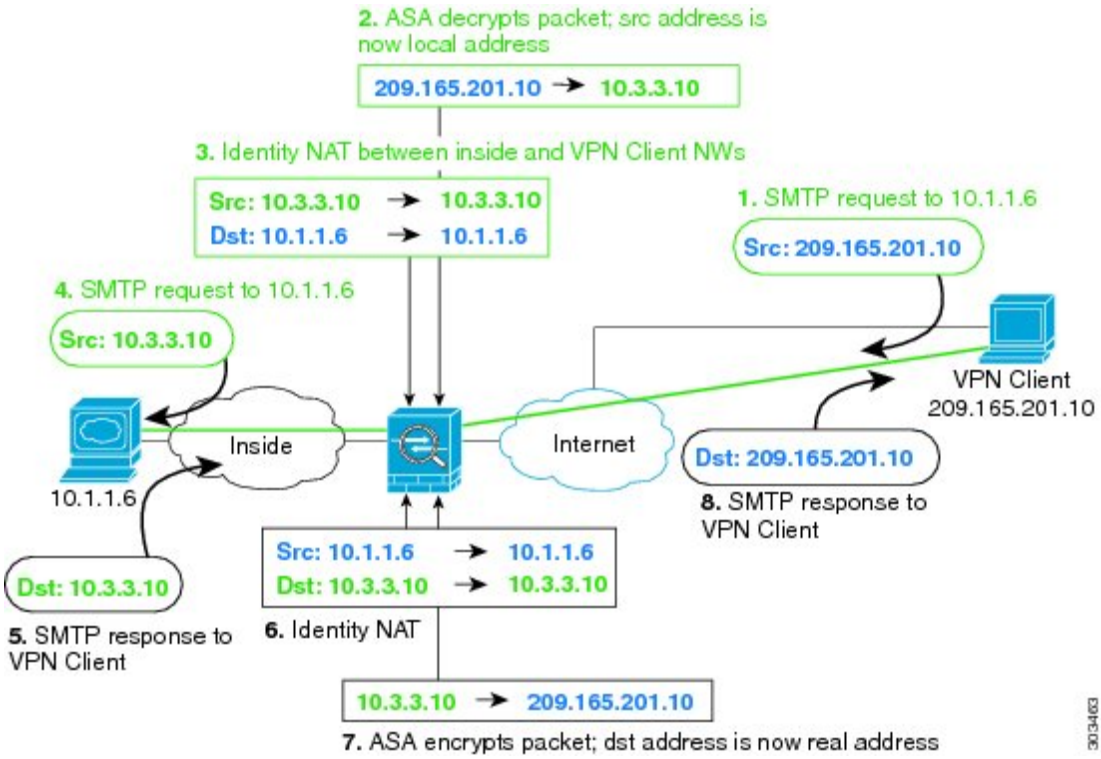
The following figure shows both an inside server (10.1.1.6) and a VPN client (209.165.201.10) accessing the Internet. Unless you configure split tunneling for the VPN client (where only specified traffic goes through the VPN tunnel), then Internet-bound VPN traffic must also go through the ASA. When the VPN traffic enters the ASA, the ASA decrypts the packet; the resulting packet includes the VPN client local address (10.3.3.10) as the source. For both inside and VPN client local networks, you need a public IP address provided by NAT to access the Internet. The below example uses interface PAT rules. To allow the VPN traffic to exit the same interface it entered, you also need to enable intra-interface communication (also known as “hairpin” networking).

Figure 29: Interface PAT for Internet-Bound VPN Traffic (Intra-Interface)



The following figure shows a VPN client that wants to access an inside mail server. Because the ASA expects traffic between the inside network and any outside network to match the interface PAT rule you set up for Internet access, traffic from the VPN client (10.3.3.10) to the SMTP server (10.1.1.6) will be dropped due to a reverse path failure: traffic from 10.3.3.10 to 10.1.1.6 does not match a NAT rule, but returning traffic from 10.1.1.6 to 10.3.3.10 *should* match the interface PAT rule for outgoing traffic. Because forward and reverse flows do not match, the ASA drops the packet when it is received. To avoid this failure, you need to exempt the inside-to-VPN client traffic from the interface PAT rule by using an identity NAT rule between those networks. Identity NAT simply translates an address to the same address.

Figure 30: Identity NAT for VPN Clients



See the following sample NAT configuration for the above network:

```
! Enable hairpin for non-split-tunneled VPN client traffic:
same-security-traffic permit intra-interface

! Identify local VPN network, & perform object interface PAT when going to Internet:
object network vpn_local
subnet 10.3.3.0 255.255.255.0
nat (outside,outside) dynamic interface

! Identify inside network, & perform object interface PAT when going to Internet:
object network inside_nw
subnet 10.1.1.0 255.255.255.0
nat (inside,outside) dynamic interface

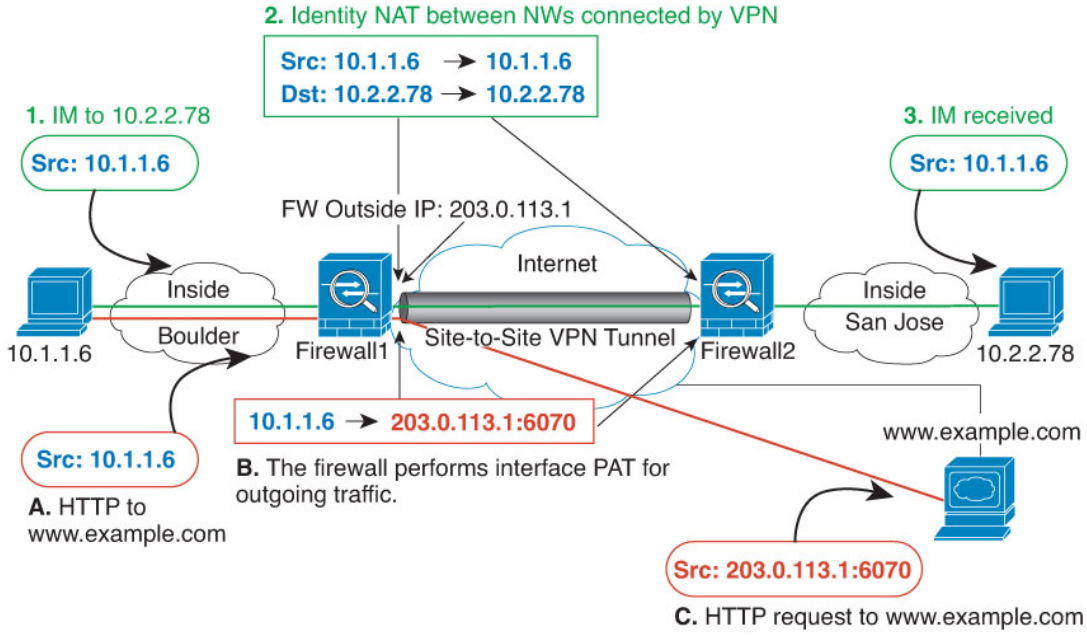
! Use twice NAT to pass traffic between the inside network and the VPN client without
! address translation (identity NAT):
nat (inside,outside) source static inside_nw inside_nw destination static vpn_local vpn_local
```

NAT and Site-to-Site VPN

The following figure shows a site-to-site tunnel connecting the Boulder and San Jose offices. For traffic that you want to go to the Internet (for example from 10.1.1.6 in Boulder to www.example.com), you need a public IP address provided by NAT to access the Internet. The below example uses interface PAT rules. However, for traffic that you want to go over the VPN tunnel (for example from 10.1.1.6 in Boulder to 10.2.2.78 in San

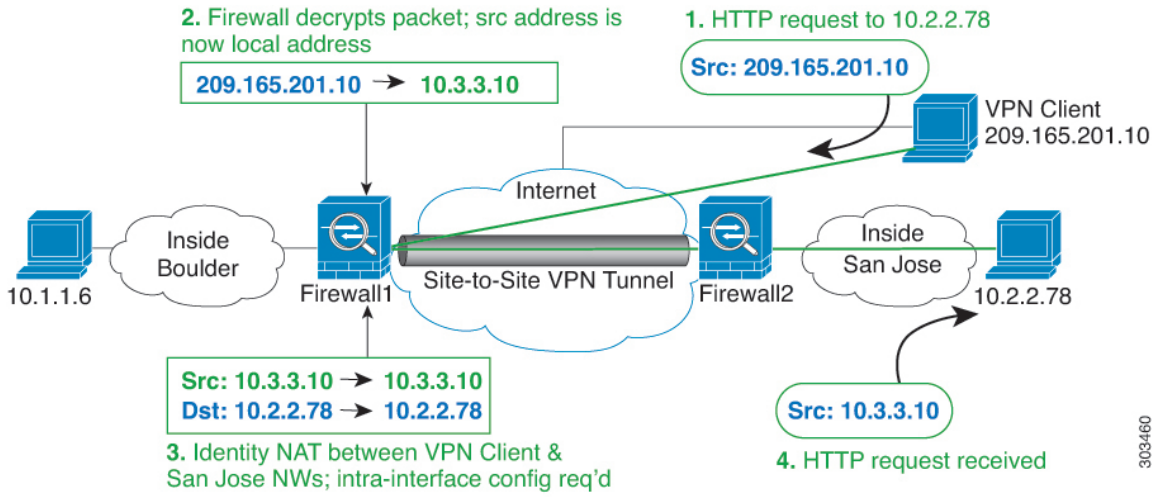
Jose), you do not want to perform NAT; you need to exempt that traffic by creating an identity NAT rule. Identity NAT simply translates an address to the same address.

Figure 31: Interface PAT and Identity NAT for Site-to-Site VPN



The following figure shows a VPN client connected to Firewall1 (Boulder), with a Telnet request for a server (10.2.2.78) accessible over a site-to-site tunnel between Firewall1 and Firewall2 (San Jose). Because this is a hairpin connection, you need to enable intra-interface communication, which is also required for non-split-tunneled Internet-bound traffic from the VPN client. You also need to configure identity NAT between the VPN client and the Boulder & San Jose networks, just as you would between any networks connected by VPN to exempt this traffic from outbound NAT rules.

Figure 32: VPN Client Access to Site-to-Site VPN



See the following sample NAT configuration for Firewall1 (Boulder) for the second example:

```
! Enable hairpin for VPN client traffic:
same-security-traffic permit intra-interface

! Identify local VPN network, & perform object interface PAT when going to Internet:
object network vpn_local
subnet 10.3.3.0 255.255.255.0
nat (outside,outside) dynamic interface

! Identify inside Boulder network, & perform object interface PAT when going to Internet:
object network boulder_inside
subnet 10.1.1.0 255.255.255.0
nat (inside,outside) dynamic interface

! Identify inside San Jose network for use in twice NAT rule:
object network sanjose_inside
subnet 10.2.2.0 255.255.255.0

! Use twice NAT to pass traffic between the Boulder network and the VPN client without
! address translation (identity NAT):
nat (inside,outside) source static boulder_inside boulder_inside
destination static vpn_local vpn_local

! Use twice NAT to pass traffic between the Boulder network and San Jose without
! address translation (identity NAT):
nat (inside,outside) source static boulder_inside boulder_inside
destination static sanjose_inside sanjose_inside

! Use twice NAT to pass traffic between the VPN client and San Jose without
! address translation (identity NAT):
nat (outside,outside) source static vpn_local vpn_local
destination static sanjose_inside sanjose_inside
```

See the following sample NAT configuration for Firewall2 (San Jose):

```
! Identify inside San Jose network, & perform object interface PAT when going to Internet:
object network sanjose_inside
subnet 10.2.2.0 255.255.255.0
nat (inside,outside) dynamic interface

! Identify inside Boulder network for use in twice NAT rule:
object network boulder_inside
subnet 10.1.1.0 255.255.255.0

! Identify local VPN network for use in twice NAT rule:
object network vpn_local
subnet 10.3.3.0 255.255.255.0

! Use twice NAT to pass traffic between the San Jose network and Boulder without
! address translation (identity NAT):
nat (inside,outside) source static sanjose_inside sanjose_inside
destination static boulder_inside boulder_inside

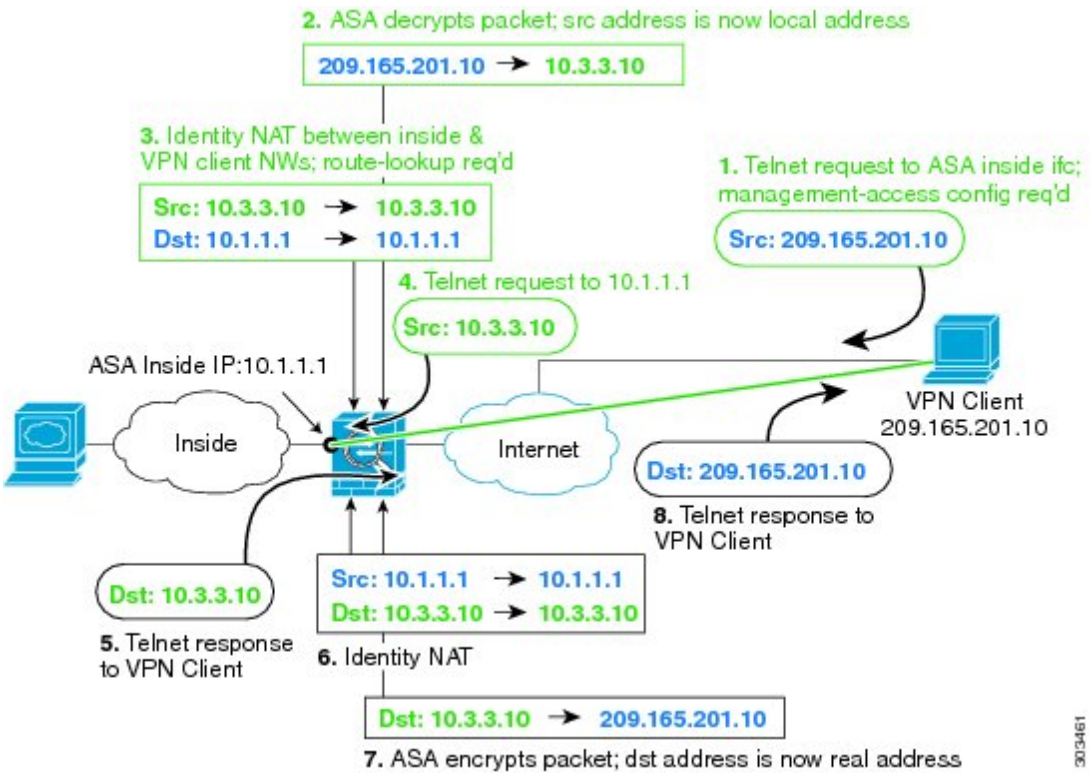
! Use twice NAT to pass traffic between the San Jose network and the VPN client without
! address translation (identity NAT):
nat (inside,outside) source static sanjose_inside sanjose_inside
destination static vpn_local vpn_local
```

NAT and VPN Management Access

When using VPN, you can allow management access to an interface other than the one from which you entered the ASA (see the **management-access** command). For example, if you enter the ASA from the outside interface, the management-access feature lets you connect to the inside interface using ASDM, SSH, Telnet, or SNMP; or you can ping the inside interface.

The following figure shows a VPN client Telnetting to the ASA inside interface. When you use a management-access interface, and you configure identity NAT according to NAT and Remote Access VPN, on page 195 or NAT and Site-to-Site VPN, on page 197, you must configure NAT with the route lookup option. Without route lookup, the ASA sends traffic out the interface specified in the NAT command, regardless of what the routing table says; in the below example, the egress interface is the inside interface. You do not want the ASA to send the management traffic out to the inside network; it will never return to the inside interface IP address. The route lookup option lets the ASA send the traffic directly to the inside interface IP address instead of to the inside network. For traffic from the VPN client to a host on the inside network, the route lookup option will still result in the correct egress interface (inside), so normal traffic flow is not affected. See the Determining the Egress Interface, on page 194 for more information about the route lookup option.

Figure 33: VPN Management Access



See the following sample NAT configuration for the above network:

```
! Enable hairpin for non-split-tunneled VPN client traffic:
same-security-traffic permit intra-interface

! Enable management access on inside ifc:
management-access inside

! Identify local VPN network, & perform object interface PAT when going to Internet:
```

```
object network vpn_local
subnet 10.3.3.0 255.255.255.0
nat (outside,outside) dynamic interface

! Identify inside network, & perform object interface PAT when going to Internet:
object network inside_nw
subnet 10.1.1.0 255.255.255.0
nat (inside,outside) dynamic interface

! Use twice NAT to pass traffic between the inside network and the VPN client without
! address translation (identity NAT), w/route-lookup:
nat (outside,inside) source static vpn_local vpn_local
destination static inside_nw inside_nw route-lookup
```

Troubleshooting NAT and VPN

See the following monitoring tools for troubleshooting NAT issues with VPN:

- Packet tracer—When used correctly, a packet tracer shows which NAT rules a packet is hitting.
- **show nat detail**—Shows hit counts and untranslated traffic for a given NAT rule.
- **show conn all**—Lets you see active connections including to and from the box traffic.

To familiarize yourself with a non-working configuration vs. a working configuration, you can perform the following steps:

1. Configure VPN without identity NAT.
2. Enter **show nat detail** and **show conn all**.
3. Add the identity NAT configuration.
4. Repeat **show nat detail** and **show conn all**.

Translating IPv6 networks

In cases where you need to pass traffic between IPv6-only and IPv4-only networks, you need to use NAT to convert between the address types. Even with two IPv6 networks, you might want to hide internal addresses from the outside network.

You can use the following translation types with IPv6 networks:

- NAT64, NAT46—Translates IPv6 packets into IPv4 and vice versa. You need to define two policies, one for the IPv6 to IPv4 translation, and one for the IPv4 to IPv6 translation. Although you can accomplish this with a single twice NAT rule, if the DNS server is on the external network, you probably need to rewrite the DNS response. Because you cannot enable DNS rewrite on a twice NAT rule when you specify a destination, creating two network object NAT rules is the better solution.



Note NAT46 supports static mappings only.

- NAT66—Translates IPv6 packets to a different IPv6 address. We recommend using static NAT. Although you can use dynamic NAT or PAT, IPv6 addresses are in such large supply, you do not have to use dynamic NAT.

Note NAT64 and NAT 46 are possible on standard routed interfaces only. NAT66 is possible on both routed and bridge group member interfaces.

NAT64/46: translating IPv6 addresses to IPv4

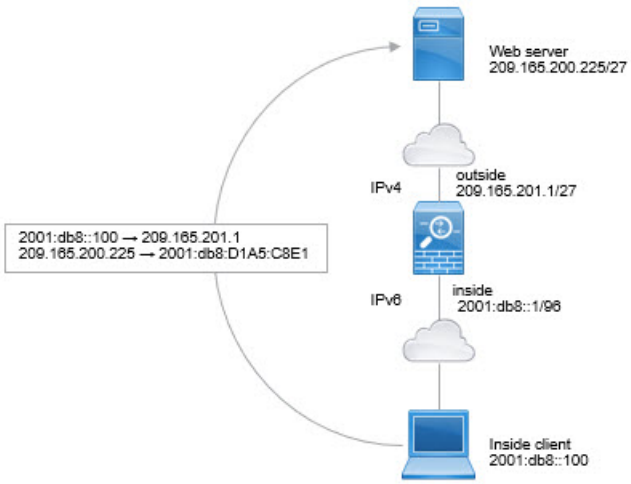
When traffic goes from an IPv6 network to an IPv4-only network, you need to convert the IPv6 address to IPv4, and return traffic from IPv4 to IPv6. You need to define two address pools, an IPv4 address pool to bind IPv6 addresses in the IPv4 network, and an IPv6 address pool to bind IPv4 addresses in the IPv6 network.

- The IPv4 address pool for the NAT64 rule is normally small and typically might not have enough addresses to map one-to-one with the IPv6 client addresses. Dynamic PAT might more easily meet the possible large number of IPv6 client addresses compared to dynamic or static NAT.
- The IPv6 address pool for the NAT46 rule can be equal to or larger than the number of IPv4 addresses to be mapped. This allows each IPv4 address to be mapped to a different IPv6 address. NAT46 supports static mappings only, so you cannot use dynamic PAT.

You need to define two policies, one for the source IPv6 network, and one for the destination IPv4 network. Although you can accomplish this with a single twice NAT rule, if the DNS server is on the external network, you probably need to rewrite the DNS response. Because you cannot enable DNS rewrite on a twice NAT rule when you specify a destination, creating two network object NAT rules is the better solution.

NAT64/46 Example: Inside IPv6 Network with Outside IPv4 Internet

Following is a straight-forward example where you have an inside IPv6-only network, and you want to convert to IPv4 for traffic sent to the Internet. This example assumes you do not need DNS translation, so you can perform both the NAT64 and NAT46 translations in a single twice NAT rule.



In this example, you translate the inside IPv6 network to IPv4 using dynamic interface PAT with the IP address of the outside interface. Outside IPv4 traffic is statically translated to addresses on the 2001:db8::/96 network, allowing transmission on the inside network.

Procedure

Step 1 Create a network object for the inside IPv6 network.

```
hostname(config)# object network inside_v6
hostname(config-network-object)# subnet 2001:db8::/96
```

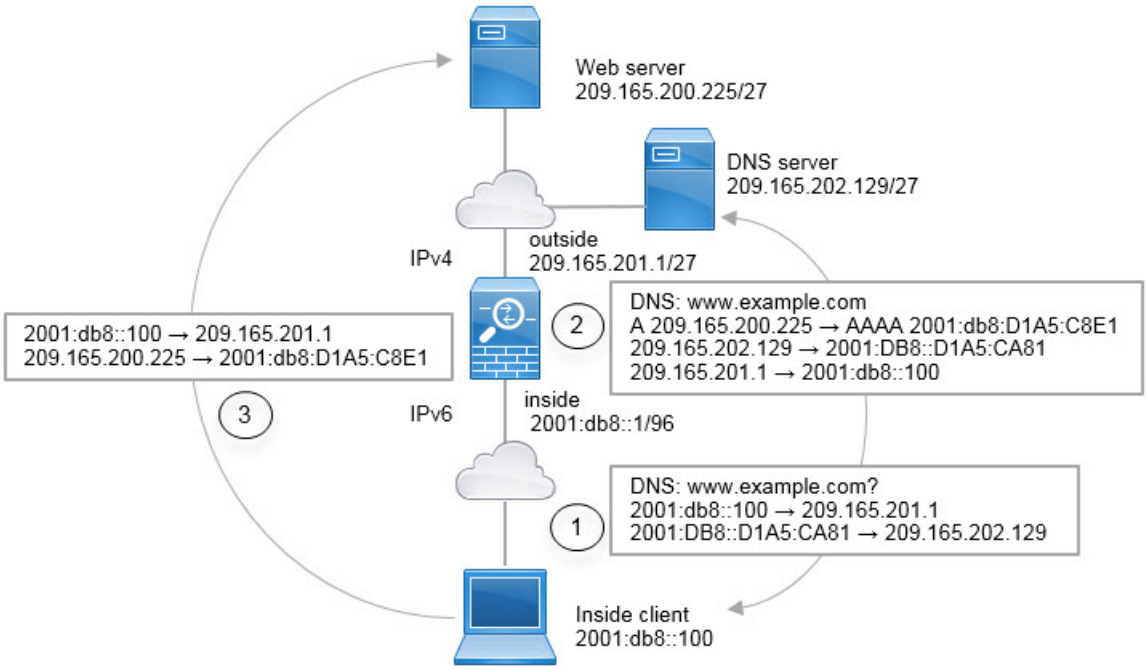
Step 2 Create the twice NAT rule to translate the IPv6 network to IPv4 and back again.

```
hostname(config)# nat (inside,outside) source dynamic inside_v6 interface
destination static inside_v6 any
```

With this rule, any traffic from the 2001:db8::/96 subnet on the inside interface going to the outside interface gets a NAT64 PAT translation using the IPv4 address of the outside interface. Conversely, any IPv4 address on the outside network coming to the inside interface is translated to an address on the 2001:db8::/96 network using the embedded IPv4 address method.

NAT64/46 Example: Inside IPv6 Network with Outside IPv4 Internet and DNS Translation

Following is a typical example where you have an inside IPv6-only network, but there are some IPv4-only services on the outside Internet that internal users need.



In this example, you translate the inside IPv6 network to IPv4 using dynamic interface PAT with the IP address of the outside interface. Outside IPv4 traffic is statically translated to addresses on the 2001:db8::/96 network, allowing transmission on the inside network. You enable DNS rewrite on the NAT46 rule, so that replies from the external DNS server can be converted from A (IPv4) to AAAA (IPv6) records, and the addresses converted from IPv4 to IPv6.

Following is a typical sequence for a web request where a client at 2001:DB8::100 on the internal IPv6 network tries to open www.example.com.

1. The client's computer sends a DNS request to the DNS server at 2001:DB8::D1A5:CA81. The NAT rules make the following translations to the source and destination in the DNS request:
   - 2001:DB8::100 to a unique port on 209.165.201.1 (The NAT64 interface PAT rule.)
   - 2001:DB8::D1A5:CA81 to 209.165.202.129 (The NAT46 rule. D1A5:CA81 is the IPv6 equivalent of 209.165.202.129.)
2. The DNS server responds with an A record indicating that www.example.com is at 209.165.200.225. The NAT46 rule, with DNS rewrite enabled, converts the A record to the IPv6-equivalent AAAA record, and translates 209.165.200.225 to 2001:db8:D1A5:C8E1in the AAAA record. In addition, the source and destination addresses in the DNS response are untranslated:
   - 209.165.202.129 to 2001:DB8::D1A5:CA81
   - 209.165.201.1 to 2001:db8::100
3. The IPv6 client now has the IP address of the web server, and makes an HTTP request to www.example.com at 2001:db8:D1A5:C8E1. (D1A5:C8E1 is the IPv6 equivalent of 209.165.200.225.) The source and destination of the HTTP request are translated:
   - 2001:DB8::100 to a unique port on 209.156.101.54 (The NAT64 interface PAT rule.)
   - 2001:db8:D1A5:C8E1 to 209.165.200.225 (The NAT46 rule.)

The following procedure explains how to configure this example.

Procedure

Step 1 Create a network object for the inside IPv6 network and add the NAT64 rule.

```
hostname(config)# object network inside_v6
hostname(config-network-object)# subnet 2001:db8::/96
hostname(config-network-object)# nat(inside,outside) dynamic interface
```

With this rule, any traffic from the 2001:db8::/96 subnet on the inside interface going to the outside interface gets a NAT64 PAT translation using the IPv4 address of the outside interface.

Step 2 Create a network object for the IPv6 translated network for the outside IPv4 network and add the NAT46 rule.

```
hostname(config)# object network outside_v4_any
hostname(config-network-object)# subnet 0.0.0.0 0.0.0.0
hostname(config-network-object)# nat(outside,inside) static 2001:db8::/96 dns
```

With this rule, any IPv4 address on the outside network coming to the inside interface is translated to an address on the 2001:db8::/96 network using the embedded IPv4 address method. In addition, DNS responses are converted from A (IPv4) to AAAA (IPv6) records, and the addresses converted from IPv4 to IPv6.

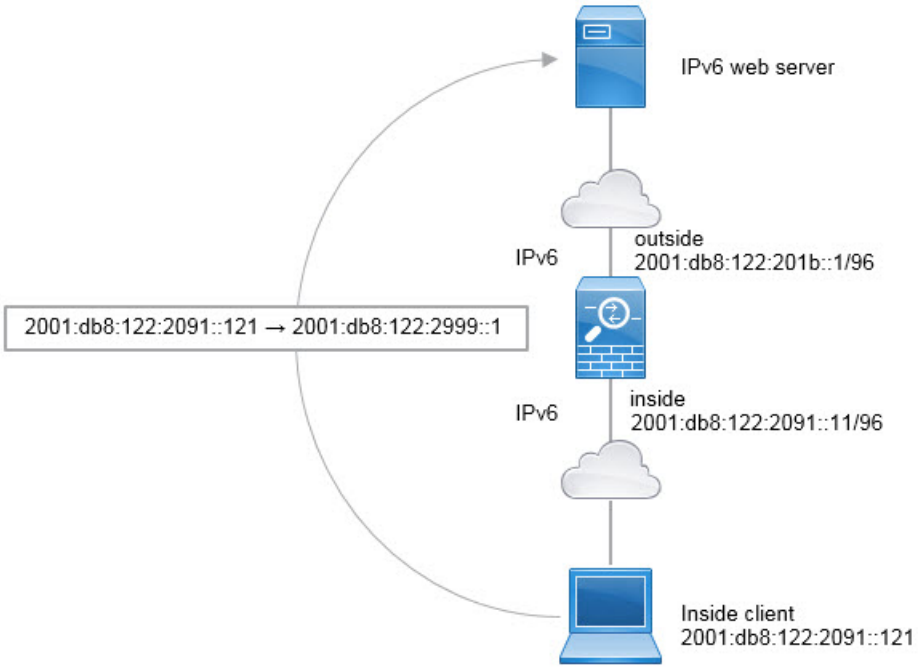
NAT66: translating IPv6 addresses to different IPv6 addresses

When going from an IPv6 network to another IPv6 network, you can translate the addresses to different IPv6 addresses on the outside network. We recommend using static NAT. Although you can use dynamic NAT or PAT, IPv6 addresses are in such large supply, you do not have to use dynamic NAT.

Because you are not translating between different address types, you need a single rule for NAT66 translations. You can easily model these rules using network object NAT. However, if you do not want to allow returning traffic, you can make the static NAT rule unidirectional using twice NAT only.

NAT66 Example, Static Translation between Networks

You can configure a static translation between IPv6 address pools using network object NAT. The following example explains how to convert inside addresses on the 2001:db8:122:2091::/96 network to outside addresses on the 2001:db8:122:2999::/96 network.



Procedure

Create the network object for the inside IPv6 network and add the static NAT rule.

```
hostname(config)# object network inside_v6
hostname(config-network-object)# subnet 2001:db8:122:2091::/96
```

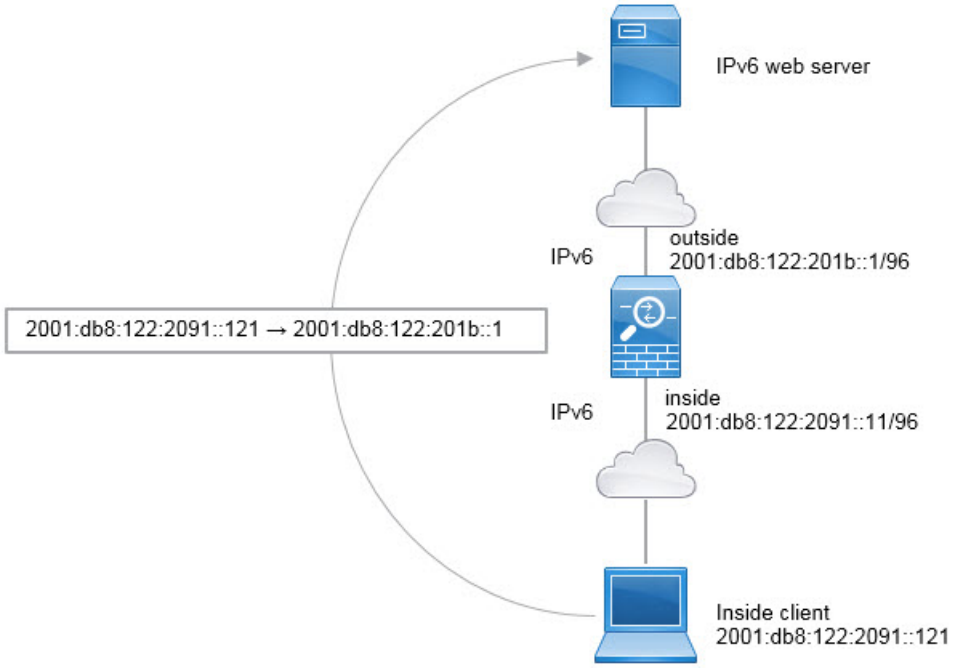
```
hostname(config-network-object)# nat(inside,outside) static 2001:db8:122:2999::/96
```

With this rule, any traffic from the 2001:db8:122:2091::/96 subnet on the inside interface going to the outside interface gets a static NAT66 translation to an address on the 2001:db8:122:2999::/96 network.

NAT66 Example, Simple IPv6 Interface PAT

A simple approach for implementing NAT66 is to dynamically assign internal addresses to different ports on the outside interface IPv6 address.

When you configure an interface PAT rule for NAT66, all the global addresses that are configured on that interface are used for PAT mapping. Link-local or site-local addresses for the interface are not used for PAT.



Procedure

Create the network object for the inside IPv6 network and add the dynamic PAT rule.

```
hostname(config)# object network inside_v6
hostname(config-network-object)# subnet 2001:db8:122:2091::/96
hostname(config-network-object)# nat(inside,outside) dynamic interface ipv6
```

With this rule, any traffic from the 2001:db8:122:2091::/96 subnet on the inside interface going to the outside interface gets a NAT66 PAT translation to one of the IPv6 global addresses configured for the outside interface.

Rewriting DNS queries and responses using NAT

You might need to configure the ASA to modify DNS replies by replacing the address in the reply with an address that matches the NAT configuration. You can configure DNS modification when you configure each translation rule. DNS modification is also known as DNS doctoring.

This feature rewrites the address in DNS queries and replies that match a NAT rule (for example, the A record for IPv4, the AAAA record for IPv6, or the PTR record for reverse DNS queries). For DNS replies traversing from a mapped interface to any other interface, the record is rewritten from the mapped value to the real value. Inversely, for DNS replies traversing from any interface to a mapped interface, the record is rewritten from the real value to the mapped value. This feature works with NAT44,NAT 66, NAT46, and NAT64.

Following are the main circumstances when you would need to configure DNS rewrite on a NAT rule.

- The rule is NAT64 or NAT46, and the DNS server is on the outside network. You need DNS rewrite to convert between DNS A records (for IPv4) and AAAA records (for IPv6).
- The DNS server is on the outside, clients are on the inside, and some of the fully-qualified domain names that the clients use resolve to other inside hosts.
- The DNS server is on the inside and responds with private IP addresses, clients are on the outside, and the clients access fully-qualified domain names that point to servers that are hosted on the inside.

DNS rewrite limitations

Following are some limitations with DNS rewrite:

- DNS rewrite is not applicable for PAT because multiple PAT rules are applicable for each A or AAAA record, and the PAT rule to use is ambiguous.
- If you configure a twice NAT rule, you cannot configure DNS modification if you specify the destination address as well as the source address. These kinds of rules can potentially have a different translation for a single address when going to A vs. B. Therefore, the can not accurately match the IP address inside the DNS reply to the correct twice NAT rule; the DNS reply does not contain information about which source/destination address combination was in the packet that prompted the DNS request.
- You must enable DNS application inspection with DNS NAT rewrite enabled for NAT rules to rewrite DNS queries and responses. By default, DNS inspection with DNS NAT rewrite enabled is globally applied, so you probably do not need to change the inspection configuration.
- DNS rewrite is actually done on the xlate entry, not the NAT rule. Thus, if there is no xlate for a dynamic rule, rewrite cannot be done correctly. The same problem does not occur for static NAT.
- DNS rewrite does not rewrite DNS Dynamic Update messages (opcode 5).

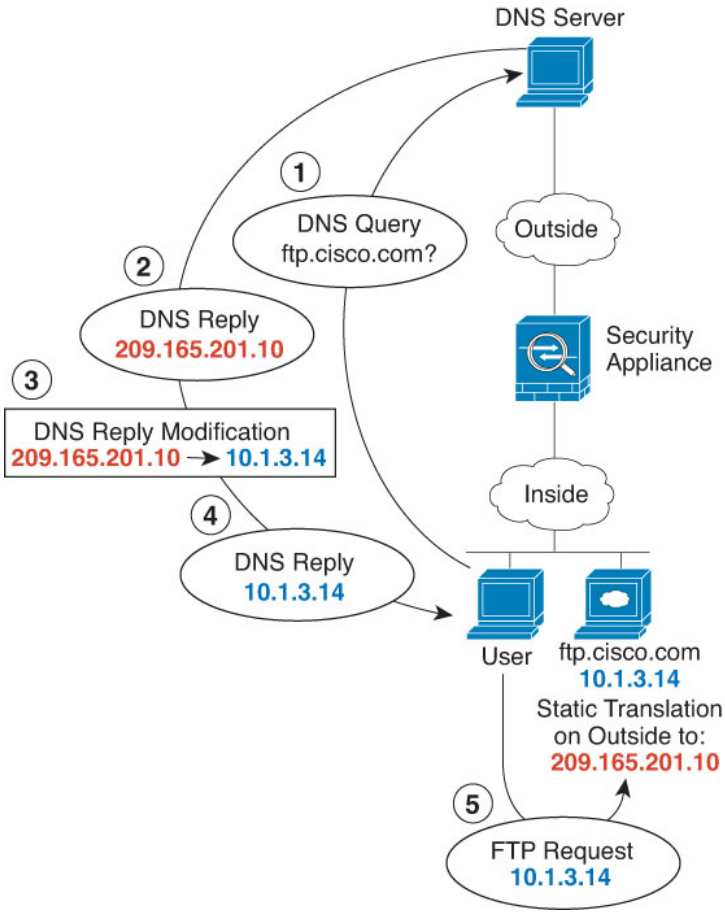
The following topics provide examples of DNS rewrite in NAT rules.

DNS Reply Modification, DNS Server on Outside

The following figure shows a DNS server that is accessible from the outside interface. A server, ftp.cisco.com, is on the inside interface. You configure NAT to statically translate the ftp.cisco.com real address (10.1.3.14) to a mapped address (209.165.201.10) that is visible on the outside network.

In this case, you want to enable DNS reply modification on this static rule so that inside users who have access to ftp.cisco.com using the real address receive the real address from the DNS server, and not the mapped address.

When an inside host sends a DNS request for the address of ftp.cisco.com, the DNS server replies with the mapped address (209.165.201.10). The system refers to the static rule for the inside server and translates the address inside the DNS reply to 10.1.3.14. If you do not enable DNS reply modification, then the inside host attempts to send traffic to 209.165.201.10 instead of accessing ftp.cisco.com directly.



Procedure

Step 1 Create a network object for the FTP server.

```
hostname(config)# object network FTP_SERVER
hostname(config-network-object)# host 10.1.3.14
```

Step 2 Configure static NAT with DNS modification.

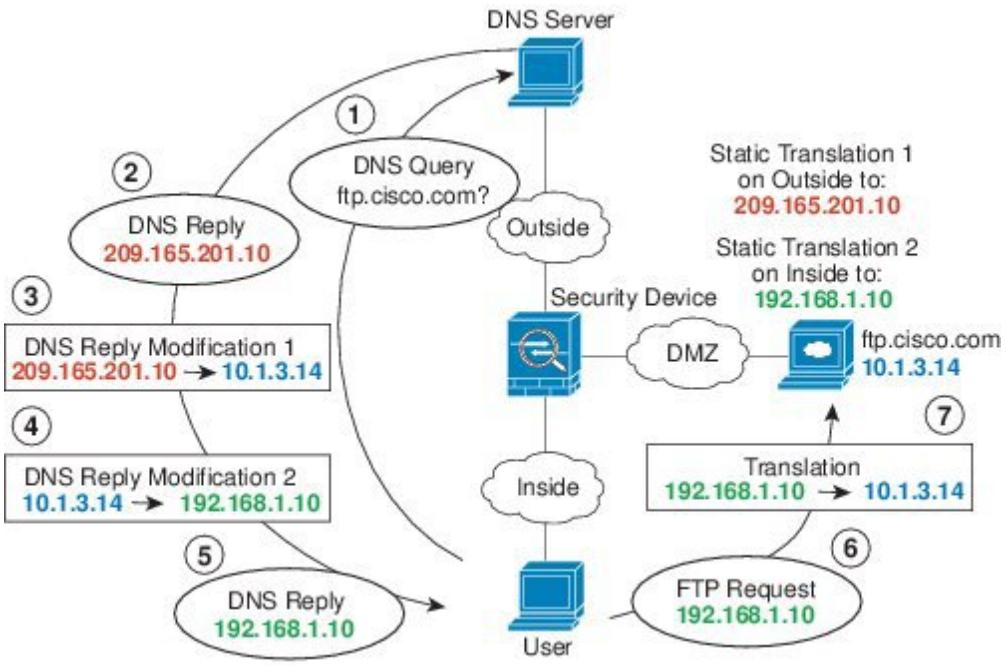
```
hostname(config-network-object)# nat (inside,outside) static 209.165.201.10 dns
```

DNS Reply Modification, DNS Server, Host, and Server on Separate Networks

The following figure shows a user on the inside network requesting the IP address for ftp.cisco.com, which is on the DMZ network, from an outside DNS server. The DNS server replies with the mapped address (209.165.201.10) according to the static rule between outside and DMZ even though the user is not on the DMZ network. The ASA translates the address inside the DNS reply to 10.1.3.14.

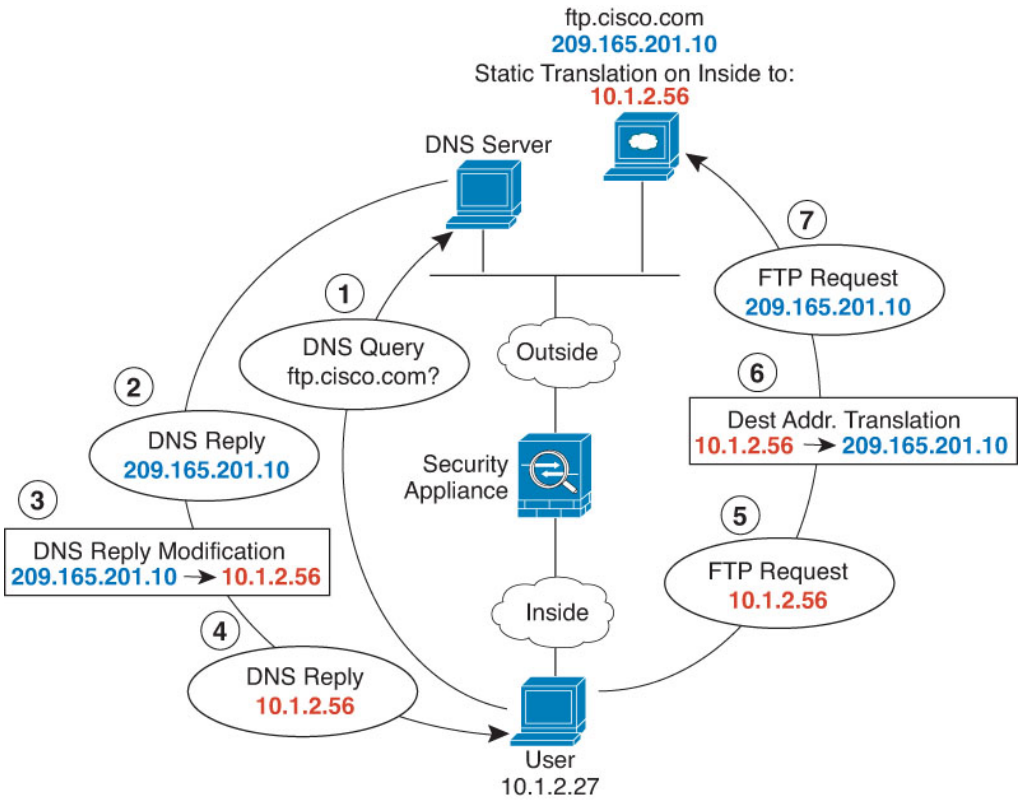
If the user needs to access ftp.cisco.com using the real address, then no further configuration is required. If there is also a static rule between the inside and DMZ, then you also need to enable DNS reply modification on this rule. The DNS reply will then be modified two times.In this case, the ASA again translates the address inside the DNS reply to 192.168.1.10 according to the static rule between inside and DMZ.

Figure 34: DNS Reply Modification, DNS Server, Host, and Server on Separate Networks



DNS Reply Modification, DNS Server on Host Network

The following figure shows an FTP server and DNS server on the outside. The system has a static translation for the outside server. In this case, when an inside user requests the address for ftp.cisco.com from the DNS server, the DNS server responds with the real address, 209.165.20.10. Because you want inside users to use the mapped address for ftp.cisco.com (10.1.2.56) you need to configure DNS reply modification for the static translation.



Procedure

Step 1 Create a network object for the FTP server.

```
hostname(config)# object network FTP_SERVER
hostname(config-network-object)# host 209.165.201.10
```

Step 2 Configure static NAT with DNS modification.

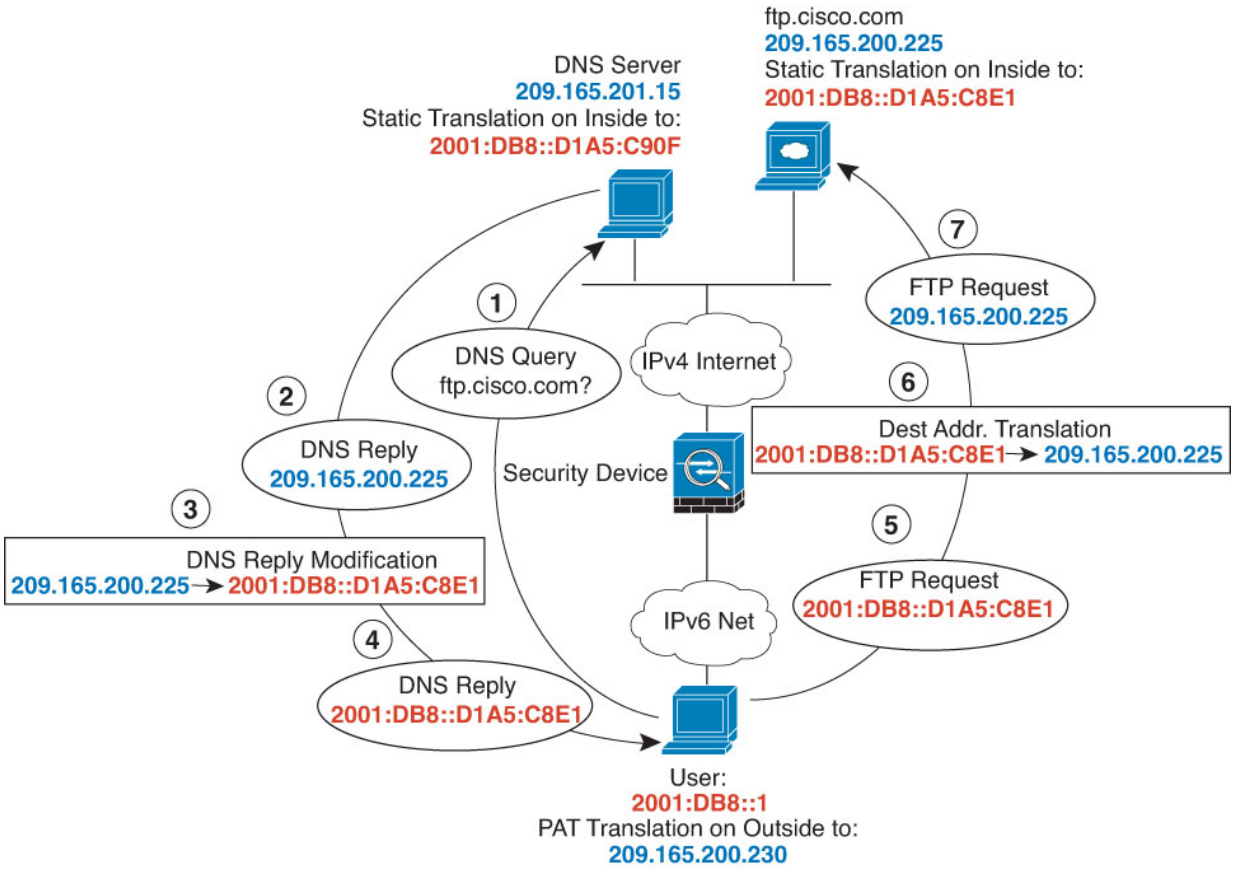
```
hostname(config-network-object)# nat (outside,inside) static 10.1.2.56 dns
```

DNS64 Reply Modification

The following figure shows an FTP server and DNS server on the outside IPv4 network. The system has a static translation for the outside server. In this case, when an inside IPv6 user requests the address for ftp.cisco.com from the DNS server, the DNS server responds with the real address, 209.165.200.225.

Because you want inside users to use the mapped address for ftp.cisco.com (2001:DB8::D1A5:C8E1, where D1A5:C8E1 is the IPv6 equivalent of 209.165.200.225) you need to configure DNS reply modification for

the static translation. This example also includes a static NAT translation for the DNS server, and a PAT rule for the inside IPv6 hosts.



Procedure

Step 1 Create a network object for the FTP server and configure static NAT with DNS modification. Because this is a one-to-one translation, include the **net-to-net** option for NAT46.

```
hostname(config)# object network FTP_SERVER
hostname(config-network-object)# host 209.165.200.225
hostname(config-network-object)# nat (outside,inside) static 2001:DB8::D1A5:C8E1/128
net-to-net dns
```

Step 2 Create a network object for the DNS server and configure static NAT. Include the **net-to-net** option for NAT46.

```
hostname(config)# object network DNS_SERVER
hostname(config-network-object)# host 209.165.201.15
hostname(config-network-object)# nat (outside,inside) static 2001:DB8::D1A5:C90F/128
net-to-net
```

Step 3 Configure an IPv4 PAT pool for translating the inside IPv6 network.

Example:

```
hostname(config)# object network IPv4_POOL
hostname(config-network-object)# range 209.165.200.230 209.165.200.235
```

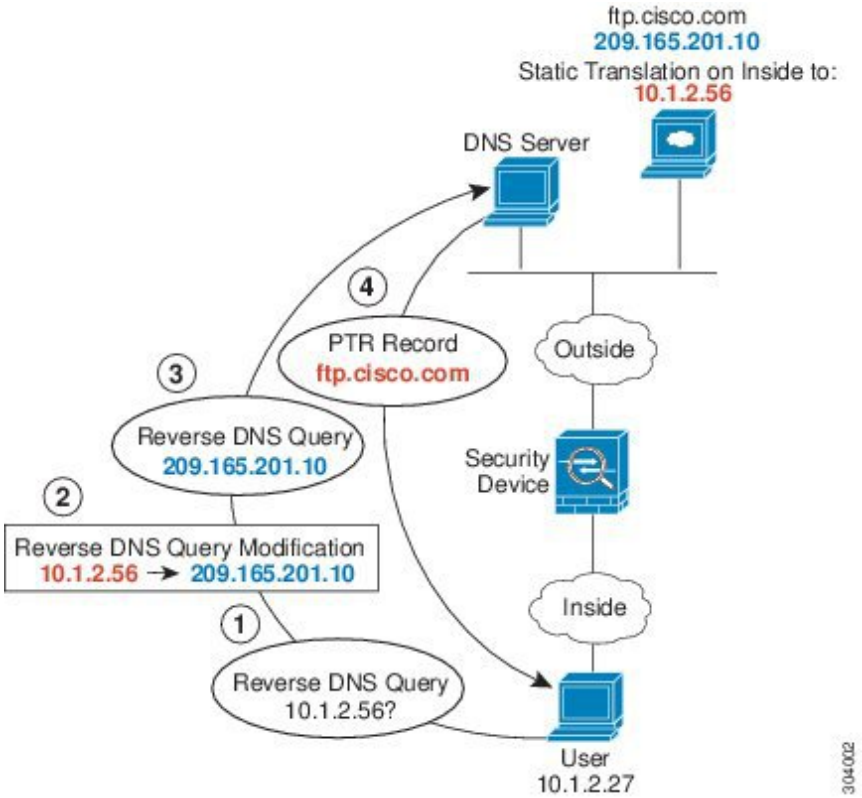
Step 4 Create a network object for the inside IPv6 network, and configure dynamic NAT with a PAT pool.

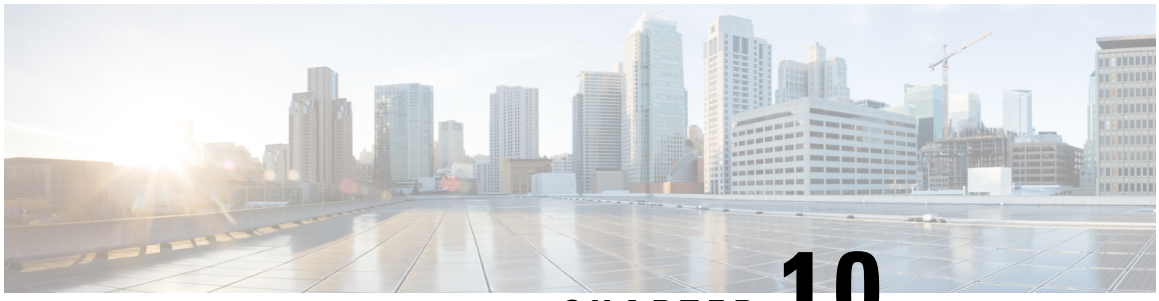
```
hostname(config)# object network IPv6_INSIDE
hostname(config-network-object)# subnet 2001:DB8::/96
hostname(config-network-object)# nat (inside,outside) dynamic pat-pool IPv4_POOL
```

PTR Modification, DNS Server on Host Network

The following figure shows an FTP server and DNS server on the outside. The ASA has a static translation for the outside server. In this case, when an inside user performs a reverse DNS lookup for 10.1.2.56, the ASA modifies the reverse DNS query with the real address, and the DNS server responds with the server name, ftp.cisco.com.

Figure 35: PTR Modification, DNS Server on Host Network





CHAPTER 10

Mapping Address and Port (MAP)

Mapping Address and Port (MAP) is a carrier-grade feature for translating IPv4 addresses to IPv6, so the traffic can be sent over the service provider's IPv6 network before being translated back to IPv4 at the service provider edge.

- About Mapping Address and Port (MAP), on page 213
- Guidelines for Mapping Address and Port (MAP), on page 214
- Configure MAP-T Domains, on page 216
- Monitoring MAP, on page 218
- History for MAP, on page 219

About Mapping Address and Port (MAP)

Mapping Address and Port (MAP) is primarily a feature for use in service provider (SP) networks. The service provider can operate an IPv6-only network, the MAP domain, while supporting IPv4-only subscribers and their need to communicate with IPv4-only sites on the public Internet. MAP is defined in RFC7597, RFC7598, and RFC7599.

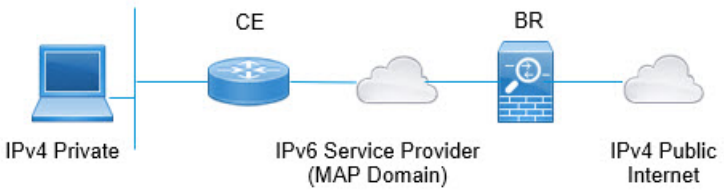
For the service provider, within the MAP domain, the benefit of MAP over NAT46 is that the substitution of an IPv6 address for the subscriber's IPv4 address (and back again to IPv4 at the SP network edge) is stateless. This provides greater efficiency within the SP network compared to NAT46.

There are two MAP techniques, MAP-Translation (MAP-T) and MAP-Encapsulation (MAP-E). The ASA supports MAP-T; MAP-E is not supported.

About Mapping Address and Port Translation (MAP-T)

With MAP-T, the subscriber's IPv4 address is first translated to the server provider's (SP) public IPv4 address, which could be either a one-to-one address mapping, or a mapping to a prefix or a shared address. Next, that IPv4 address is translated to an IPv6 address within the MAP domain, and the packet is transmitted over the SP IPv6 network. At the network edge, the SP's border relay is responsible for translating the IPv6 address back to the SP's IPv4 address before routing the packet to the public IPv4 network. The exact reverse is performed for traffic coming from the public IPv4 network to the subscriber.

Figure 36: MAP-T Network



By using MAP-T, you can transition the SP network to an IPv6-only architecture while allowed subscribers to continue using IPv4 and communicate with IPv4-only Internet or other sites outside the SP network.

MAP-T behaves like a NAT64 translation but instead of using an IPv6 address with an embedded IPv4 address, it uses an encoding scheme that also embeds the port number. Thus, MAP-T provides a way to restrict the port range used by devices.

A MAP-T system includes the following:

- **Customer Edge (CE) device**—The CE is a home gateway (wireless router, cable modem with router, and so forth). The CE provides IPv4/IPv6 translation as well as native IPv6 forwarding. It has one WAN-side provider-facing IPv6-addressed interface and one or more LAN-side interfaces addressed using private IPv4 addressing. You would configure one or more MAP domains for the CE to use to translate IPv4 packets to IPv6 and vice-verse.
- **Border Relay (BR) device**—You would install the ASA as a border relay. The BR is a provider-side component at the edge of the MAP domain that supports the IPv4/IPv6 translation. The BR has at least one IPv6-enabled interface and one IPv4 interface connected to the IPv4 network. You would configure one or more MAP domains for the BR to use to translate IPv4 packets to IPv6 and vice-verse. You must configure the CEs and BR with the same MAP domain rules.
- **MAP Domain**—A MAP domain is a mechanism to group a set of MAP-T CE devices with a set of MAP-T BR devices. A domain is a set of parameters that are shared between the BR and CE devices assigned to the domain. You configure the same domain with the same parameters on each of the BR and CE devices.

Guidelines for Mapping Address and Port (MAP)

Firewall Mode Guidelines

You can configure MAP in routed mode only. Transparent mode is not supported.

Additional Guidelines

- The ASA does not get involved in packet forwarding in mesh mode. Thus, you cannot configure forward mapping rules (FMR) in a MAP domain.
- MAP does not support tunneled VPN, multicast, or anycast traffic.
- You cannot use both NAT and MAP on a given connection. Ensure that your NAT and MAP rules do not overlap. You will get unexpected results if you have overlapping rules.
- The follow inspections do not support MAP translation. Packets subject to these inspections are not translated.

- CTIQBE
- DCERPC
- Diameter
- Name resolution over WINS
- GTP
- H.323 and H.225 and H.245 and RAS
- ILS (LDAP)
- Instant Messaging
- IP Options (RFC 791, 2113)
- IPSec Pass Through
- LISP
- M3UA
- MGCP
- MMP
- NetBIOS
- PPTP
- RADIUS accounting
- RSH
- RTSP
- SIP
- SKINNY
- SMTP and ESMTP
- SNMP
- SQL*Net
- STUN
- Sun RPC
- TFTP
- WAAS
- XDMCP
- Active FTP

Configure MAP-T Domains

To configure MAP-T, you create one or more domains. When you configure MAP-T on customer edge (CE) and border relay (BR) devices, ensure that you use the same parameters for each device that will participate in each domain.

You can configure up to 25 MAP-T domains. In multiple-context mode, you can configure up to 25 domains per context.

Procedure

Step 1 Create (or edit) the MAP domain.

map-domain *name*

Where the name is an alphanumeric string up to 48 characters. The name can also include the following special characters: period (.), slash (/), and colon (:).

Example:

```
ciscoasa(config)# map-domain 1
ciscoasa(config-map-domain)#
```

Step 2 Configure the default mapping rule.

default-mapping-rule *ipv6_prefix/prefix_length*

Specify an IPv6 prefix to be used to embed IPv4 destination addresses per RFC 6052. The prefix length should normally be 64, but allowed values are 32, 40, 48, 56, 64 or 96. Any trailing bits after the embedded IPv4 address are set to 0.

The border relay (BR) device uses this rule to translate all IPv4 addresses outside the MAP domain to an IPv6 address that works within the MAP domain.

Example:

```
ciscoasa(config-map-domain)# default-mapping-rule 2001:DB8:CAFE:CAFE::/64
```

Step 3 Configure the basic mapping rule.

The customer edge (CE) device uses the basic mapping rule to determine its dedicated IPv4 addressing or shared address and port set assignment. The CE device first translates the system's IPv4 address to an IPv4 address and port within the pool's prefix and port range (using NAT44), then MAP translates the new IPv4 address to an IPv6 address within the pool defined by the rule's IPv6 prefix. The packet is then ready to be transmitted over the service provider's IPv6-only network to a border relay (BR) device.

a) Enter basic mapping rule configuration mode.

basic-mapping-rule

b) Configure the IPv4 prefix.

ipv4-prefix *ipv4_network_address netmask*

The IPv4 prefix defines the IPv4 address pool for the customer edge (CE) device. The CE device first translates its IPv4 address to an address (and port number) in the pool defined by the IPv4 prefix. MAP then translates this new address to an IPv6 address using the prefix in the default mapping rule.

Specify a network address and subnet mask, for example, 192.168.3.0 255.255.255.0. You cannot use the same IPv4 prefix in different MAP domains.

c) Configure the IPv6 prefix.

ipv6-prefix *ipv6_prefix/prefix_length*

The IPv6 prefix defines the address pool for the CE device's IPv6 address. MAP translates IPv6 packets back to IPv4 only if the packets have a destination address with this prefix and a source address with the IPv6 prefix defined in the default mapping rule, and is within the right port range. Any IPv6 packets sent to the CE device from other addresses are simply processed as IPv6 traffic without MAP translation. Packets from the MAP source/destination pools, but with out-of-range ports, are simply dropped.

Specify an IPv6 prefix and prefix length, which is normally 64, but cannot be less than 8. You cannot use the same IPv6 prefix in different MAP domains.

d) Configure the starting port.

start-port *number*

The first port in the port pool for the translated address. The port you specify must be a power of 2, from 1-32768 such as 1, 2, 4, 8, and so forth. If you want to exclude the well-known ports, start at 1024 or higher.

e) Configure the port ratio, which determines how many ports are in the port pool.

share-ratio *number*

Specify the number of ports that should be in the pool. The number must be a power of 2, from 1-65536, such as 1, 2, 4, 8, and so forth.

Example:

```
ciscoasa(config-map-domain)# basic-mapping-rule
ciscoasa(config-map-domain-bmr)# ipv4-prefix 192.168.3.0 255.255.255.0
ciscoasa(config-map-domain-bmr)# ipv6-prefix 2001:cafe:cafe:1::/64
ciscoasa(config-map-domain-bmr)# start-port 1024
ciscoasa(config-map-domain-bmr)# share-ratio 16
```

Example

```
ciscoasa(config)# map-domain 1
ciscoasa(config-map-domain)# default-mapping-rule 2001:DB8:CAFE:CAFE::/64
ciscoasa(config-map-domain)# basic-mapping-rule
ciscoasa(config-map-domain-bmr)# ipv4-prefix 192.168.3.0 255.255.255.0
ciscoasa(config-map-domain-bmr)# ipv6-prefix 2001:cafe:cafe:1::/64
ciscoasa(config-map-domain-bmr)# start-port 1024
ciscoasa(config-map-domain-bmr)# share-ratio 16
```

Monitoring MAP

The following topics explain how you can monitor the MAP configuration and activity.

Verifying the MAP Domain Configuration

You can view the map domains and their status to verify the configuration is correct.

The **show map-domain** command displays the MAP configuration (similar to the **show running-config map-domain** command), but also indicates whether a domain configuration is valid. In the following example, there are two domains, 1 and 2. The output explains that MAP domain 2 is incomplete, and thus it is not active.

```
MAP Domain 1
  Default Mapping Rule
    IPv6 prefix 2001:db8:cafe:cafe::/64
  Basic Mapping Rule
    IPv6 prefix 2001:cafe:cafe:1::/64
    IPv4 prefix 192.168.3.0 255.255.255.0
    share ratio 16
    start port 1024
    PSID length 4
    PSID offset 6
    Rule EA-bit length 12

MAP Domain 2
  Default Mapping Rule
    IPv6 prefix 2001:db8:1234:1234::/64

Warning: map-domain 2 configuration is incomplete and not in effect.
```

Monitoring MAP Syslog Messages

If you enable syslog, you can monitor MAP behavior with the following syslog messages:

- 305018: MAP translation from *interface name:source IP address/source port-destination IP address/destination port* to *interface name:translated source IP address/translated source port-translated destination IP address/translated destination port*

  A new MAP translation was made. The message shows the original and translated source and destination.

- 305019: MAP node address *IP address/port* has inconsistent Port Set ID encoding

  A packet has an address that matches MAP basic mapping rules, which means it is meant to be translated, but the Port Set ID encoded within the address is inconsistent per RFC7599. This likely means there is a software fault on the MAP node where this packet originates.

- 305020: MAP node with address *IP address* is not allowed to use port *port*

  A packet has an address that matches MAP basic mapping rules, which means it is meant to be translated, but the associated port does not fall within the range allocated to that address. This likely means there is misconfiguration on the MAP node where this packet originates.

History for MAP

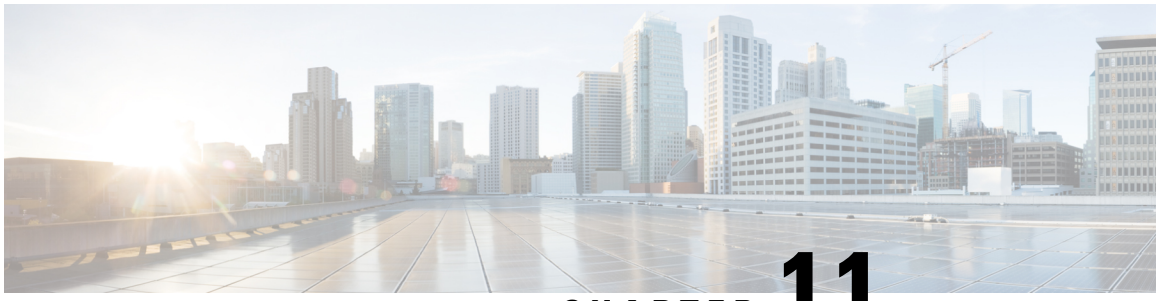
Feature Name	Platform Releases	Description
Mapping Address and Port-Translation (MAP-T)	9.13(1)	Mapping Address and Port (MAP) is primarily a feature for use in service provider (SP) networks. The service provider can operate an IPv6-only network, the MAP domain, while supporting IPv4-only subscribers and their need to communicate with IPv4-only sites on the public Internet. MAP is defined in RFC7597, RFC7598, and RFC7599. We introduced or modified the following commands: **basic-mapping-rule**, **default-mapping-rule**, **ipv4-prefix**, **ipv6-prefix**, **map-domain**, **share-ratio**, **show map-domain**, **start-port**.



PART IV

Service Policies and Application Inspection

- Service Policy, on page 223
- Getting Started with Application Layer Protocol Inspection, on page 245
- Inspection of Basic Internet Protocols, on page 265
- Inspection for Voice and Video Protocols, on page 305
- Inspection for Mobile Networks, on page 329



CHAPTER 11

Service Policy

Service policies using Modular Policy Framework provide a consistent and flexible way to configure ASA features. For example, you can use a service policy to create a timeout configuration that is specific to a particular TCP application, as opposed to one that applies to all TCP applications. A service policy consists of multiple actions or rules applied to an interface or applied globally.

- About Service Policies, on page 223
- Guidelines for Service Policies, on page 229
- Defaults for Service Policies, on page 231
- Configure Service Policies, on page 232
- Monitoring Service Policies, on page 240
- Examples for Service Policies (Modular Policy Framework), on page 240
- History for Service Policies, on page 243

About Service Policies

The following topics describe how service policies work.

The Components of a Service Policy

The point of service policies is to apply advanced services to the traffic you are allowing. Any traffic permitted by access rules can have service policies applied, and thus receive special processing, such as being redirected to a service module or having application inspection applied.

You can have these types of service policy:

- One global policy that gets applied to all interfaces.
- One service policy applied per interface. The policy can be a mix of classes for traffic going through the device and management traffic directed at the ASA interface rather than going through it,

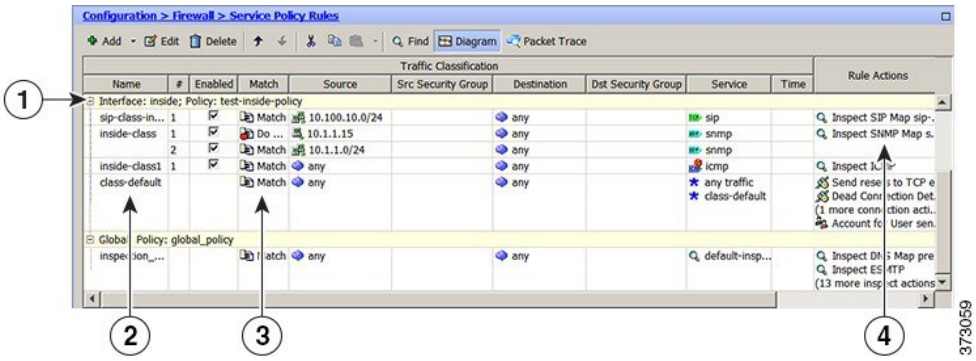
Each service policy is composed of the following elements:

1. Service policy map, which is the ordered set of rules, and is named on the **service-policy** command. In ASDM, the policy map is represented as a folder on the Service Policy Rules page.
2. Rules, each rule being a **class** command within the service policy map and the commands associated with the **class** command. In ASDM, each rule is shown on a separate row, and the name of the rule is the class name.

The **class** command defines the traffic matching criteria for the rule.

The commands associated with class, such as **inspect**, **set connection timeout**, and so forth, define the services and constraints to apply to matching traffic. Note that inspect commands can point to inspection policy maps, which define actions to apply to inspected traffic. Keep in mind that inspection policy maps are not the same as service policy maps.

The following example compares how service policies appear in the CLI with how they appear in ASDM. Note that there is not a one-to-one mapping between the figure call-outs and lines in the CLI.



The following CLI is generated by the rules shown in the figure above.

```
: Access lists used in class maps.
: In ASDM, these map to call-out 3, from the Match to the Time fields.
access-list inside_mpc line 1 extended permit tcp 10.100.10.0 255.255.255.0 any eq sip
access-list inside_mpc_1 line 1 extended deny udp host 10.1.1.15 any eq snmp
access-list inside_mpc_1 line 2 extended permit udp 10.1.1.0 255.255.255.0 any eq snmp
access-list inside_mpc_2 line 1 extended permit icmp any any
: SNMP map for SNMP inspection. Denies all but v3.
: In ASDM, this maps to call-out 4, rule actions, for the class-inside policy.
snmp-map snmp-v3only
  deny version 1
  deny version 2
  deny version 2c
: Inspection policy map to define SIP behavior.
: The sip-high inspection policy map must be referred to by an inspect sip command
: in the service policy map.
: In ASDM, this maps to call-out 4, rule actions, for the sip-class-inside policy.
policy-map type inspect sip sip-high
  parameters
    rtp-conformance enforce-payloadtype
    no traffic-non-sip
    software-version action mask log
    uri-non-sip action mask log
    state-checking action drop-connection log
    max-forwards-validation action drop log
    strict-header-validation action drop log
: Class map to define traffic matching for the inside-class rule.
: In ASDM, this maps to call-out 3, from the Match to the Time fields.
class-map inside-class
  match access-list inside_mpc_1
: Class map to define traffic matching for the sip-class-inside rule.
: In ASDM, this maps to call-out 3, from the Match to the Time fields.
class-map sip-class-inside
  match access-list inside_mpc
: Class map to define traffic matching for the inside-class1 rule.
: In ASDM, this maps to call-out 3, from the Match to the Time fields.
```

```
class-map inside-class1
  match access-list inside_mpc_2
: Policy map that actually defines the service policy rule set named test-inside-policy.
: In ASDM, this corresponds to the folder at call-out 1.
policy-map test-inside-policy
: First rule in test-inside-policy, named sip-class-inside. Inspects SIP traffic.
: The sip-class-inside rule applies the sip-high inspection policy map to SIP inspection.
: In ASDM, each rule corresponds to call-out 2.
  class sip-class-inside
    inspect sip sip-high
: Second rule, inside-class. Applies SNMP inspection using an SNMP map.
  class inside-class
    inspect snmp snmp-v3only
: Third rule, inside-class1. Applies ICMP inspection.
  class inside-class1
    inspect icmp
: Fourth rule, class-default. Applies connection settings and enables user statistics.
  class class-default
    set connection timeout embryonic 0:00:30 half-closed 0:10:00 idle 1:00:00
reset dcd 0:15:00 5
    user-statistics accounting
: The service-policy command applies the policy map rule set to the inside interface.
: This command activates the policies.
service-policy test-inside-policy interface inside
```

Features Configured with Service Policies

The following table lists the features you configure using service policies.

Table 8: Features Configured with Service Policies

Feature	For Through Traffic?	For Management Traffic?	See:
Application inspection (multiple types)	**All** except RADIUS accounting	**RADIUS accounting** only	• Getting Started with Application Layer Protocol Inspection, on page 245. • Inspection of Basic Internet Protocols, on page 265. • Inspection for Voice and Video Protocols, on page 305. • Inspection for Mobile Networks, on page 329.
NetFlow Secure Event Logging filtering	**Yes**	**Yes**	See the NetFlow implementation guide.
QoS input and output policing	**Yes**	No	Quality of Service, on page 411.
QoS standard priority queue	**Yes**	No	Quality of Service, on page 411.
TCP and UDP connection limits and timeouts, and TCP sequence number randomization	**Yes**	**Yes**	Connection Settings, on page 377.
TCP normalization	**Yes**	No	Connection Settings, on page 377.
TCP state bypass	**Yes**	No	Connection Settings, on page 377.

Feature	For Through Traffic?	For Management Traffic?	See:
User statistics for Identity Firewall	**Yes**	**Yes**	See the **user-statistics** command in the command reference.

Feature Directionality

Actions are applied to traffic bidirectionally or unidirectionally depending on the feature. For features that are applied bidirectionally, all traffic that enters or exits the interface to which you apply the policy map is affected if the traffic matches the class map for both directions.



Note When you use a global policy, all features are unidirectional; features that are normally bidirectional when applied to a single interface only apply to the ingress of each interface when applied globally. Because the policy is applied to all interfaces, the policy will be applied in both directions so bidirectionality in this case is redundant.

For features that are applied unidirectionally, for example QoS priority queue, only traffic that enters (or exits, depending on the feature) the interface to which you apply the policy map is affected. See the following table for the directionality of each feature.

Table 9: Feature Directionality

Feature	Single Interface Direction	Global Direction
Application inspection (multiple types)	Bidirectional	Ingress
NetFlow Secure Event Logging filtering	N/A	Ingress
QoS input policing	Ingress	Ingress
QoS output policing	Egress	Egress
QoS standard priority queue	Egress	Egress
TCP and UDP connection limits and timeouts, and TCP sequence number randomization	Bidirectional	Ingress
TCP normalization	Bidirectional	Ingress
TCP state bypass	Bidirectional	Ingress
User statistics for Identity Firewall	Bidirectional	Ingress

Feature Matching Within a Service Policy

A packet matches class maps in a policy map for a given interface according to the following rules:

1. A packet can match only one class map in the policy map for each feature type.

2. When the packet matches a class map for a feature type, the ASA does not attempt to match it to any subsequent class maps for that feature type.
3. If the packet matches a subsequent class map for a different feature type, however, then the ASA also applies the actions for the subsequent class map, if supported. See Incompatibility of Certain Feature Actions, on page 228 for more information about unsupported combinations.

Note Application inspection includes multiple inspection types, and most are mutually exclusive. For inspections that can be combined, each inspection is considered to be a separate feature.

Examples of Packet Matching

For example:

- If a packet matches a class map for connection limits, and also matches a class map for an application inspection, then both actions are applied.
- If a packet matches a class map for HTTP inspection, but also matches another class map that includes HTTP inspection, then the second class map actions are not applied.
- If a packet matches a class map for HTTP inspection, but also matches another class map that includes FTP inspection, then the second class map actions are not applied because HTTP and FTP inspections cannot be combined.
- If a packet matches a class map for HTTP inspection, but also matches another class map that includes IPv6 inspection, then both actions are applied because the IPv6 inspection can be combined with any other type of inspection.

Order in Which Multiple Feature Actions are Applied

The order in which different types of actions in a policy map are performed is independent of the order in which the actions appear in the policy map.

Actions are performed in the following order:

1. QoS input policing
2. TCP normalization, TCP and UDP connection limits and timeouts, TCP sequence number randomization, and TCP state bypass.

Note When a the ASA performs a proxy service (such as AAA) or it modifies the TCP payload (such as FTP inspection), the TCP normalizer acts in dual mode, where it is applied before and after the proxy or payload modifying service.

3. Application inspections that can be combined with other inspections:
   a. IPv6
   b. IP options

c. WAAS

4. Application inspections that cannot be combined with other inspections. See Incompatibility of Certain Feature Actions, on page 228 for more information.
5. QoS output policing
6. QoS standard priority queue

Note NetFlow Secure Event Logging filtering and User statistics for Identity Firewall are order-independent.

Incompatibility of Certain Feature Actions

Some features are not compatible with each other for the same traffic. The following list might not include all incompatibilities; for information about compatibility of each feature, see the chapter or section for the feature:

- You cannot configure QoS priority queuing and QoS policing for the same set of traffic.
- Most inspections should not be combined with another inspection, so the ASA only applies one inspection if you configure multiple inspections for the same traffic. Exceptions are listed in Order in Which Multiple Feature Actions are Applied, on page 227.

Note The **match default-inspection-traffic** command, which is used in the default global policy, is a special CLI shortcut to match the default ports for all inspections. When used in a policy map, this class map ensures that the correct inspection is applied to each packet, based on the destination port of the traffic. For example, when UDP traffic for port 69 reaches the ASA, then the ASA applies the TFTP inspection; when TCP traffic for port 21 arrives, then the ASA applies the FTP inspection. So in this case only, you can configure multiple inspections for the same class map. Normally, the ASA does not use the port number to determine which inspection to apply, thus giving you the flexibility to apply inspections to non-standard ports, for example.

An example of a misconfiguration is if you configure multiple inspections in the same policy map and do not use the default-inspection-traffic shortcut. In the first example, traffic destined to port 21 is mistakenly configured for both FTP and HTTP inspection. In the second example, traffic destined to port 80 is mistakenly configured for both FTP and HTTP inspection. In both cases of misconfiguration examples, only the FTP inspection is applied, because FTP comes before HTTP in the order of inspections applied.

Example 1: Misconfiguration for FTP packets: HTTP Inspection Also Configured

```
class-map ftp
   match port tcp eq 21
class-map http
   match port tcp eq 21 [it should be 80]
policy-map test
   class ftp
     inspect ftp
   class http
```

```
     inspect http
```

Example 2: Misconfiguration for HTTP packets: FTP Inspection Also Configured

```
class-map ftp
   match port tcp eq 80 [it should be 21]
class-map http
   match port tcp eq 80
policy-map test
   class ftp
     inspect ftp
   class http
     inspect http
```

Feature Matching for Multiple Service Policies

For TCP and UDP traffic (and ICMP when you enable stateful ICMP inspection), service policies operate on traffic flows, and not just individual packets. If traffic is part of an existing connection that matches a feature in a policy on one interface, that traffic flow cannot also match the same feature in a policy on another interface; only the first policy is used.

For example, if HTTP traffic matches a policy on the inside interface to inspect HTTP traffic, and you have a separate policy on the outside interface for HTTP inspection, then that traffic is not also inspected on the egress of the outside interface. Similarly, the return traffic for that connection will not be inspected by the ingress policy of the outside interface, nor by the egress policy of the inside interface.

For traffic that is not treated as a flow, for example ICMP when you do not enable stateful ICMP inspection, returning traffic can match a different policy map on the returning interface.

Guidelines for Service Policies

Inspection Guidelines

There is a separate topic that provides detailed guidelines for application inspection service policies. See Guidelines for Application Inspection, on page 247.

IPv6 Guidelines

Supports IPv6 for the following features:

- Application inspection for several, but not all, protocols. For details, see Guidelines for Application Inspection, on page 247.
- NetFlow Secure Event Logging filtering
- SCTP state bypass
- TCP and UDP connection limits and timeouts, TCP sequence number randomization
- TCP normalization
- TCP state bypass

- User statistics for Identity Firewall

Class Map (Traffic Class) Guidelines

The maximum number of class maps (traffic classes) of all types is 255 in single mode or per context in multiple mode. Class maps include the following types:

- Layer 3/4 class maps (for through traffic and management traffic).
- Inspection class maps
- Regular expression class maps
- **match** commands used directly underneath an inspection policy map

This limit also includes default class maps of all types, limiting user-configured class maps to approximately 235.

Policy Map Guidelines

See the following guidelines for using policy maps:

- You can only assign one policy map per interface. For low-end firewalls, you can create up to 64 policy maps in the configuration; for more powerful firewalls, you can create up to 128.

  You can only assign one policy map per interface. For lower-end firewalls, such as the Secure Firewall 3100, you can create up to 64 policy maps in the configuration. For more powerful firewalls, such as the Firepower 4100, you can create up to 128.

- You can apply the same policy map to multiple interfaces.
- You can identify up to 63 Layer 3/4 class maps in a Layer 3/4 policy map.
- For each class map, you can assign multiple actions from one or more feature types, if supported. See Incompatibility of Certain Feature Actions, on page 228.

Service Policy Guidelines

- Interface service policies on ingress interfaces take precedence over the global service policy for a given feature. For example, if you have a global policy with FTP inspection, and an interface policy with TCP normalization, then both FTP inspection and TCP normalization are applied to the interface. However, if you have a global policy with FTP inspection, and an ingress interface policy with FTP inspection, then only the ingress interface policy FTP inspection is applied to that interface. If no ingress or global policy implements a feature, then an interface service policy on the egress interface that specifies the feature is applied.
- You can only apply one global policy. For example, you cannot create a global policy that includes feature set 1, and a separate global policy that includes feature set 2. All features must be included in a single policy.
- When you make service policy changes to the configuration, all *new* connections use the new service policy. Existing connections continue to use the policy that was configured at the time of the connection establishment. Output for the **show** command will not include data about the old connections.

  For example, if you remove a QoS service policy from an interface, then add a modified version, then the **show service-policy** command only displays QoS counters associated with new connections that

match the new service policy; existing connections on the old policy no longer show in the command output.

To ensure that all connections use the new policy, you need to disconnect the current connections so they can reconnect using the new policy. Use the **clear conn** or **clear local-host** commands.

Defaults for Service Policies

The following topics describe the default settings for service policies and the Modular Policy Framework.

Default Service Policy Configuration

By default, the configuration includes a policy that matches all default application inspection traffic and applies certain inspections to the traffic on all interfaces (a global policy). Not all inspections are enabled by default. You can only apply one global policy, so if you want to alter the global policy, you need to either edit the default policy or disable it and apply a new one. (An interface policy overrides the global policy for a particular feature.)

The default policy includes the following application inspections:

- DNS
- FTP
- H323 (H225)
- H323 (RAS)
- RSH
- RTSP
- ESMTP
- SQLnet
- Skinny (SCCP)
- SunRPC
- SIP
- NetBios
- TFTP
- IP Options

Default Class Maps (Traffic Classes)

The configuration includes a default Layer 3/4 class map (traffic class) that the ASA uses in the default global policy called default-inspection-traffic; it matches the default inspection traffic. This class, which is used in the default global policy, is a special shortcut to match the default ports for all inspections.

When used in a policy, this class ensures that the correct inspection is applied to each packet, based on the destination port of the traffic. For example, when UDP traffic for port 69 reaches the ASA, then the ASA applies the TFTP inspection; when TCP traffic for port 21 arrives, then the ASA applies the FTP inspection. So in this case only, you can configure multiple inspections for the same class map. Normally, the ASA does not use the port number to determine which inspection to apply, thus giving you the flexibility to apply inspections to non-standard ports, for example.

```
class-map inspection_default
 match default-inspection-traffic
```

Another class map that exists in the default configuration is called class-default, and it matches all traffic. This class map appears at the end of all Layer 3/4 policy maps and essentially tells the ASA to not perform any actions on all other traffic. If you use the class-default class (the Any traffic class in ASDM), use it for broader purposes like enabling TCP sequence randomization, TTL decrement, or any other optional attribute changes. Do not use it for inspection, or it can cause traffic disruptions. Some features are only available for class-default.

```
class-map class-default
 match any
```

Configure Service Policies

To configure service policies using the Modular Policy Framework, perform the following steps:

Procedure

Step 1 Identify the traffic on which you want to act by creating Layer 3/4 class maps, as described in Identify Traffic (Layer 3/4 Class Maps), on page 234.

For example, you might want to perform actions on all traffic that passes through the ASA; or you might only want to perform certain actions on traffic from 10.1.1.0/24 to any destination address.



Step 2 Optionally, perform additional actions on some inspection traffic.

If one of the actions you want to perform is application inspection, and you want to perform additional actions on some inspection traffic, then create an inspection policy map. The inspection policy map identifies the traffic and specifies what to do with it.

For example, you might want to drop all HTTP requests with a body length greater than 1000 bytes.

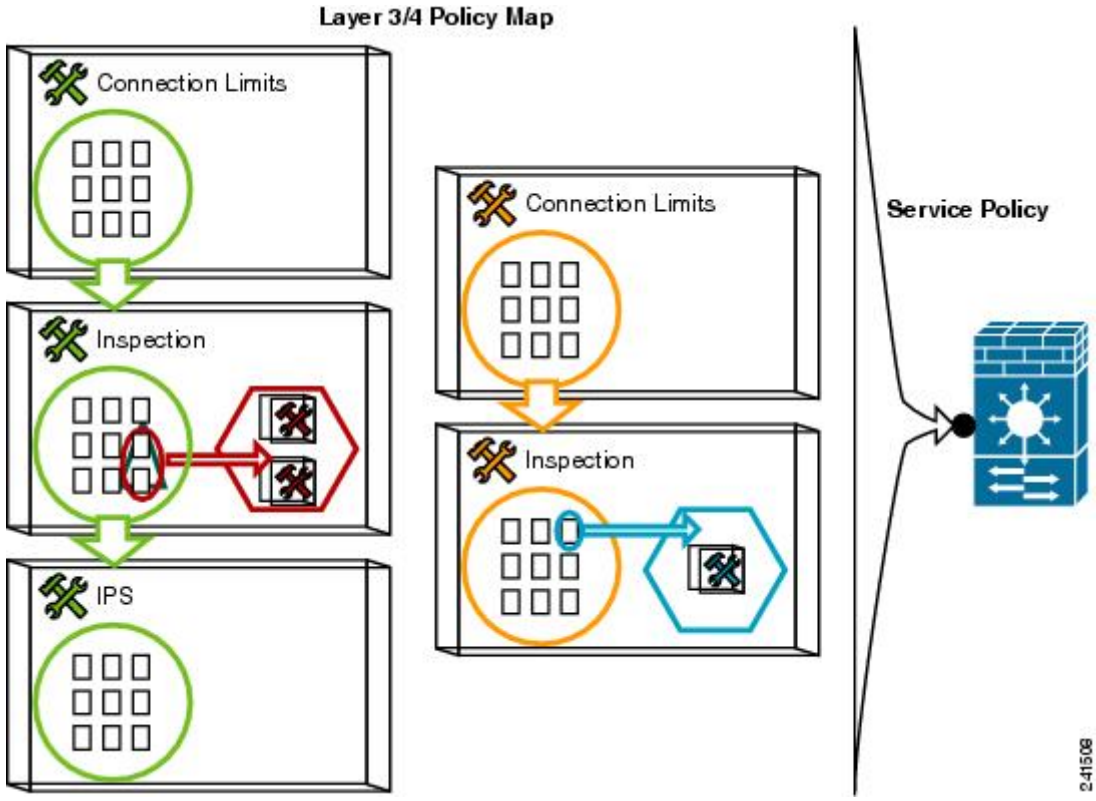


You can create a self-contained inspection policy map that identifies the traffic directly with **match** commands, or you can create an inspection class map for reuse or for more complicated matching. For example, you could match text within a inspected packets using a regular expression or a group of regular expressions (a regular expression class map), and target actions based on narrower criteria. For example, you might want to drop all HTTP requests with a URL including the text "example.com."



See Configure Application Layer Protocol Inspection, on page 253.

Step 3 Define the actions you want to perform on each Layer 3/4 class map by creating a Layer 3/4 policy map, as described in Define Actions (Layer 3/4 Policy Map), on page 238.



Step 4 Determine on which interfaces you want to apply the policy map, or apply it globally, as described in Apply Actions to an Interface (Service Policy), on page 239.

Identify Traffic (Layer 3/4 Class Maps)

A Layer 3/4 class map identifies Layer 3 and 4 traffic to which you want to apply actions. You can create multiple Layer 3/4 class maps for each Layer 3/4 policy map.

Create a Layer 3/4 Class Map for Through Traffic

A Layer 3/4 class map matches traffic based on protocols, ports, IP addresses and other Layer 3 or 4 attributes.

Tip We suggest that you only inspect traffic on ports on which you expect application traffic; if you inspect all traffic, for example using **match any**, the ASA performance can be impacted.

Procedure

Step 1 Create a Layer 3/4 class map: **class-map** *class_map_name*

Where *class_map_name* is a string up to 40 characters in length.

The name “class-default” is reserved. All types of class maps use the same name space, so you cannot reuse a name already used by another type of class map. The CLI enters class-map configuration mode.

Example:

```
hostname(config)# class-map all_udp
```

Step 2 (Optional) Add a description to the class map.

description *string*

Example:

```
hostname(config-cmap)# description All UDP traffic
```

Step 3 Match traffic using one of the following commands. Unless otherwise specified, you can include only one **match** command in the class map.

- **match any**—Matches all traffic.

  ```
  hostname(config-cmap)# match any
  ```

- **match access-list** *access_list_name*—Matches traffic specified by an extended ACL.

  ```
  hostname(config-cmap)# match access-list udp
  ```

- **match port** {**tcp** | **udp** | **sctp**} {**eq** *port_num* | **range** *port_num port_num*}—Matches destination ports, either a single port or a contiguous range of ports, for the indicated protocol. For applications that use multiple, non-contiguous ports, use the **match access-list** command and define an ACE to match each port.

  ```
  hostname(config-cmap)# match tcp eq 80
  ```

- **match default-inspection-traffic**—Matches default traffic for inspection: the default TCP and UDP ports used by all applications that the ASA can inspect.

  ```
  hostname(config-cmap)# match default-inspection-traffic
  ```

  This command, which is used in the default global policy, is a special CLI shortcut that when used in a policy map, ensures that the correct inspection is applied to each packet, based on the destination port of the traffic. For example, when UDP traffic for port 69 reaches the ASA, then the ASA applies the TFTP inspection; when TCP traffic for port 21 arrives, then the ASA applies the FTP inspection. So in this case only, you can configure multiple inspections for the same class map (with the exception of WAAS inspection, which can be configured with other inspections. See Incompatibility of Certain Feature Actions, on page 228 for more information about combining actions). Normally, the ASA does not use the port number to determine the inspection applied, thus giving you the flexibility to apply inspections to non-standard ports, for example.

  See Default Inspections and NAT Limitations, on page 249 for a list of default ports. Not all applications whose ports are included in the **match default-inspection-traffic** command are enabled by default in the policy map.

You can specify a **match access-list** command along with the **match default-inspection-traffic** command to narrow the matched traffic. Because the **match default-inspection-traffic** command specifies the ports and protocols to match, any ports and protocols in the ACL are ignored.

- **match dscp** *value1* [*value2*] [...] [*value8*]—Matches the DSCP value in an IP header, up to eight DSCP values.

```
hostname(config-cmap)# match dscp af43 cs1 ef
```

- **match precedence** *value1* [*value2*] [*value3*] [*value4*]—Matches up to four precedence values, represented by the TOS byte in the IP header, where the precedence values can be 0 to 7.

```
hostname(config-cmap)# match precedence 1 4
```

- **match rtp** *starting_port range*—Matches RTP traffic, where the *starting_port* specifies an even-numbered UDP destination port between 2000 and 65534. The *range* specifies the number of additional UDP ports to match above the *starting_port*, between 0 and 16383.

```
hostname(config-cmap)# match rtp 4004 100
```

- **match tunnel-group** *name*—Matches VPN tunnel group traffic to which you want to apply QoS.

  You can also specify one other **match** command to refine the traffic match. You can specify any of the preceding commands, except for the **match any**, **match access-list**, or **match default-inspection-traffic** commands. Or you can also enter the **match flow ip destination-address** command to match flows in the tunnel group going to each IP address.

```
hostname(config-cmap)# match tunnel-group group1
hostname(config-cmap)# match flow ip destination-address
```

Examples

The following is an example for the **class-map** command:

```
hostname(config)# access-list udp permit udp any any
hostname(config)# access-list tcp permit tcp any any
hostname(config)# access-list host_foo permit ip any 10.1.1.1 255.255.255.255

hostname(config)# class-map all_udp
hostname(config-cmap)# description "This class-map matches all UDP traffic"
hostname(config-cmap)# match access-list udp

hostname(config-cmap)# class-map all_tcp
hostname(config-cmap)# description "This class-map matches all TCP traffic"
hostname(config-cmap)# match access-list tcp

hostname(config-cmap)# class-map all_http
hostname(config-cmap)# description "This class-map matches all HTTP traffic"
hostname(config-cmap)# match port tcp eq http
```

```
hostname(config-cmap)# class-map to_server
hostname(config-cmap)# description "This class-map matches all traffic to server 10.1.1.1"
hostname(config-cmap)# match access-list host_foo
```

Create a Layer 3/4 Class Map for Management Traffic

For management traffic to the ASA, you might want to perform actions specific to this kind of traffic. You can specify a management class map that can match an ACL or TCP or UDP ports. The types of actions available for a management class map in the policy map are specialized for management traffic.

Procedure

Step 1 Create a management class map: **class-map type management** *class_map_name*

Where *class_map_name* is a string up to 40 characters in length.

The name "class-default" is reserved. All types of class maps use the same name space, so you cannot reuse a name already used by another type of class map. The CLI enters class-map configuration mode.

Example:

```
hostname(config)# class-map management all_udp
```

Step 2 (Optional) Add a description to the class map.

description *string*

Example:

```
hostname(config-cmap)# description All UDP traffic
```

Step 3 Match traffic using one of the following commands.

- **match access-list** *access_list_name*—Matches traffic specified by an extended ACL.

  ```
  hostname(config-cmap)# match access-list udp
  ```

- **match port** {**tcp** | **udp** | **sctp**} {**eq** *port_num* | **range** *port_num port_num*}—Matches destination ports, either a single port or a contiguous range of ports, for the indicated protocol. For applications that use multiple, non-contiguous ports, use the **match access-list** command and define an ACE to match each port.

  ```
  hostname(config-cmap)# match tcp eq 80
  ```

Define Actions (Layer 3/4 Policy Map)

After you configure Layer 3/4 class maps to identify traffic, use a Layer 3/4 policy map to associate actions to those classes.

Tip The maximum number of policy maps is 64, but you can only apply one policy map per interface.

Procedure

Step 1 Add the policy map: **policy-map** *policy_map_name*

Where *policy_map_name* is the name of the policy map, up to 40 characters in length. All types of policy maps use the same name space, so you cannot reuse a name already used by another type of policy map. The CLI enters policy-map configuration mode.

Example:

```
hostname(config)# policy-map global_policy
```

Step 2 Specify a previously configured Layer 3/4 class map: **class** *class_map_name*

Where the *class_map_name* is the name of the class map.

See Identify Traffic (Layer 3/4 Class Maps), on page 234 to add a class map.

Example:

```
hostname(config-pmap)# class all_http
```

Step 3 Specify one or more actions for this class map.

See Features Configured with Service Policies, on page 225.

Note
If there is no **match default-inspection-traffic** command in a class map, then at most one **inspect** command is allowed to be configured under the class.

Step 4 Repeat the process for each class map you want to include in this policy map.

Examples

The following is an example of a **policy-map** command for a connection policy. It limits the number of connections allowed to the web server 10.1.1.1:

```
hostname(config)# access-list http-server permit tcp any host 10.1.1.1
hostname(config)# class-map http-server
hostname(config-cmap)# match access-list http-server
```

```
hostname(config)# policy-map global-policy
hostname(config-pmap)# description This policy map defines a policy concerning
connection to http server.
hostname(config-pmap)# class http-server
hostname(config-pmap-c)# set connection conn-max 256
```

The following example shows how multi-match works in a policy map:

```
hostname(config)# class-map inspection_default
hostname(config-cmap)# match default-inspection-traffic
hostname(config)# class-map http_traffic
hostname(config-cmap)# match port tcp eq 80

hostname(config)# policy-map outside_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect http http_map
hostname(config-pmap-c)# inspect sip
hostname(config-pmap)# class http_traffic
hostname(config-pmap-c)# set connection timeout idle 0:10:0
```

The following example shows how traffic matches the first available class map, and will not match any subsequent class maps that specify actions in the same feature domain:

```
hostname(config)# class-map telnet_traffic
hostname(config-cmap)# match port tcp eq 23
hostname(config)# class-map ftp_traffic
hostname(config-cmap)# match port tcp eq 21
hostname(config)# class-map tcp_traffic
hostname(config-cmap)# match port tcp range 1 65535
hostname(config)# class-map udp_traffic
hostname(config-cmap)# match port udp range 0 65535
hostname(config)# policy-map global_policy
hostname(config-pmap)# class telnet_traffic
hostname(config-pmap-c)# set connection timeout idle 0:0:0
hostname(config-pmap-c)# set connection conn-max 100
hostname(config-pmap)# class ftp_traffic
hostname(config-pmap-c)# set connection timeout idle 0:5:0
hostname(config-pmap-c)# set connection conn-max 50
hostname(config-pmap)# class tcp_traffic
hostname(config-pmap-c)# set connection timeout idle 2:0:0
hostname(config-pmap-c)# set connection conn-max 2000
```

When a Telnet connection is initiated, it matches **class telnet_traffic**. Similarly, if an FTP connection is initiated, it matches **class ftp_traffic**. For any TCP connection other than Telnet and FTP, it will match **class tcp_traffic**. Even though a Telnet or FTP connection can match **class tcp_traffic**, the ASA does not make this match because they previously matched other classes.

Apply Actions to an Interface (Service Policy)

To activate the Layer 3/4 policy map, create a service policy that applies it to one or more interfaces or that applies it globally to all interfaces. Use the following command:

service-policy *policy_map_name* {**global** | **interface** *interface_name*} [**fail-close**]

Where:

- *policy_map_name* is the name of the policy map.

- **global** creates a service policy that applies to all interfaces that do not have a specific policy.

  You can only apply one global policy, so if you want to alter the global policy, you need to either edit the default policy or disable it and apply a new one. By default, the configuration includes a global policy that matches all default application inspection traffic and applies inspection to the traffic globally. The default service policy includes the following command: **service-policy global_policy global**.

- **interface** *interface_name* creates a service policy by associating a policy map with an interface.

- **fail-close** generates a syslog (767001) for IPv6 traffic that is dropped by application inspections that do not support IPv6 traffic. By default, syslogs are not generated.

Examples

For example, the following command enables the inbound_policy policy map on the outside interface:

```
hostname(config)# service-policy inbound_policy interface outside
```

The following commands disable the default global policy, and enables a new one called new_global_policy.

```
hostname(config)# no service-policy global_policy global
hostname(config)# service-policy new_global_policy global
```

Monitoring Service Policies

To monitor service policies, enter the following command:

- **show service-policy**

  Displays the service policy statistics.

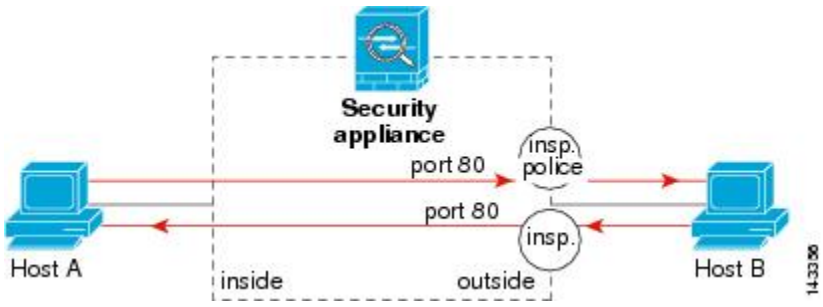
Examples for Service Policies (Modular Policy Framework)

This section includes several Modular Policy Framework examples.

Applying Inspection and QoS Policing to HTTP Traffic

In this example, any HTTP connection (TCP traffic on port 80) that enters or exits the ASA through the outside interface is classified for HTTP inspection. Any HTTP traffic that exits the outside interface is classified for policing.

Figure 37: HTTP Inspection and QoS Policing



See the following commands for this example:

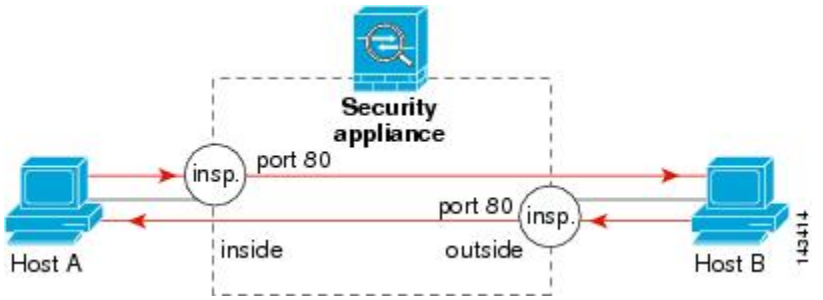
```
hostname(config)# class-map http_traffic
hostname(config-cmap)# match port tcp eq 80

hostname(config)# policy-map http_traffic_policy
hostname(config-pmap)# class http_traffic
hostname(config-pmap-c)# inspect http
hostname(config-pmap-c)# police output 250000
hostname(config)# service-policy http_traffic_policy interface outside
```

Applying Inspection to HTTP Traffic Globally

In this example, any HTTP connection (TCP traffic on port 80) that enters the ASA through any interface is classified for HTTP inspection. Because the policy is a global policy, inspection occurs only as the traffic enters each interface.

Figure 38: Global HTTP Inspection



See the following commands for this example:

```
hostname(config)# class-map http_traffic
hostname(config-cmap)# match port tcp eq 80

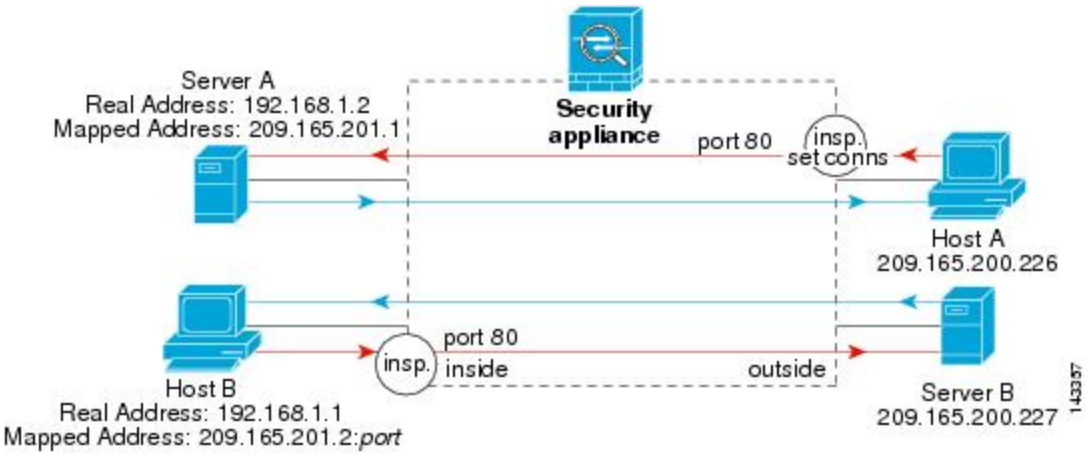
hostname(config)# policy-map http_traffic_policy
hostname(config-pmap)# class http_traffic
hostname(config-pmap-c)# inspect http
hostname(config)# service-policy http_traffic_policy global
```

Applying Inspection and Connection Limits to HTTP Traffic to Specific Servers

In this example, any HTTP connection destined for Server A (TCP traffic on port 80) that enters the ASA through the outside interface is classified for HTTP inspection and maximum connection limits. Connections initiated from Server A to Host A do not match the ACL in the class map, so they are not affected.

Any HTTP connection destined for Server B that enters the ASA through the inside interface is classified for HTTP inspection. Connections initiated from Server B to Host B do not match the ACL in the class map, so they are not affected.

Figure 39: HTTP Inspection and Connection Limits to Specific Servers



See the following commands for this example:

```
hostname(config)# object network obj-192.168.1.2
hostname(config-network-object)# host 192.168.1.2
hostname(config-network-object)# nat (inside,outside) static 209.165.201.1
hostname(config)# object network obj-192.168.1.0
hostname(config-network-object)# subnet 192.168.1.0 255.255.255.0
hostname(config-network-object)# nat (inside,outside) dynamic 209.165.201.2
hostname(config)# access-list serverA extended permit tcp any host 209.165.201.1 eq 80
hostname(config)# access-list ServerB extended permit tcp any host 209.165.200.227 eq 80

hostname(config)# class-map http_serverA
hostname(config-cmap)# match access-list serverA
hostname(config)# class-map http_serverB
hostname(config-cmap)# match access-list serverB

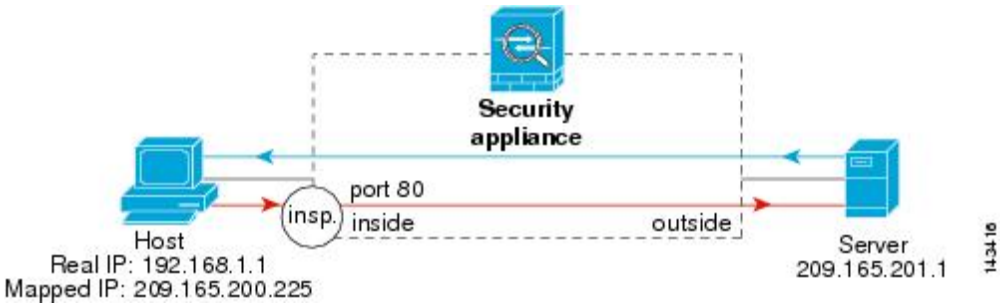
hostname(config)# policy-map policy_serverA
hostname(config-pmap)# class http_serverA
hostname(config-pmap-c)# inspect http
hostname(config-pmap-c)# set connection conn-max 100
hostname(config)# policy-map policy_serverB
hostname(config-pmap)# class http_serverB
hostname(config-pmap-c)# inspect http

hostname(config)# service-policy policy_serverB interface inside
hostname(config)# service-policy policy_serverA interface outside
```

Applying Inspection to HTTP Traffic with NAT

In this example, the Host on the inside network has two addresses: one is the real IP address 192.168.1.1, and the other is a mapped IP address used on the outside network, 209.165.200.225. You must use the real IP address in the ACL in the class map. If you applied it to the outside interface, you would also use the real address.

Figure 40: HTTP Inspection with NAT



See the following commands for this example:

```
hostname(config)# object network obj-192.168.1.1
hostname(config-network-object)# host 192.168.1.1
hostname(config-network-object)# nat (VM1,outside) static 209.165.200.225

hostname(config)# access-list http_client extended permit tcp host 192.168.1.1 any eq 80

hostname(config)# class-map http_client
hostname(config-cmap)# match access-list http_client

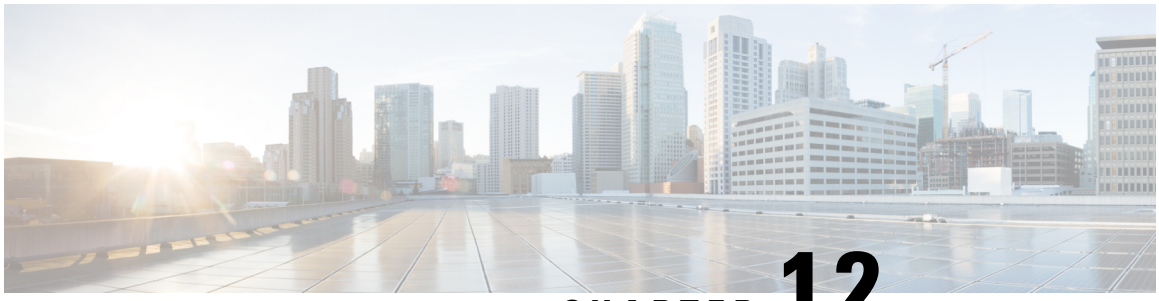
hostname(config)# policy-map http_client
hostname(config-pmap)# class http_client
hostname(config-pmap-c)# inspect http

hostname(config)# service-policy http_client interface inside
```

History for Service Policies

Feature Name	Releases	Description
Modular Policy Framework	7.0(1)	Modular Policy Framework was introduced.
Management class map for use with RADIUS accounting traffic	7.2(1)	The management class map was introduced for use with RADIUS accounting traffic. The following commands were introduced: **class-map type management**, and **inspect radius-accounting**.
Inspection policy maps	7.2(1)	The inspection policy map was introduced. The following command was introduced: **class-map type inspect**.

Feature Name	Releases	Description
Regular expressions and policy maps	7.2(1)	Regular expressions and policy maps were introduced to be used under inspection policy maps. The following commands were introduced: **class-map type regex**, **regex**, **match regex**.
Match any for inspection policy maps	8.0(2)	The **match any** keyword was introduced for use with inspection policy maps: traffic can match one or more criteria to match the class map. Formerly, only **match all** was available.



CHAPTER 12

Getting Started with Application Layer Protocol Inspection

The following topics describe how to configure application layer protocol inspection.

- Application Layer Protocol Inspection, on page 245
- Configure Application Layer Protocol Inspection, on page 253
- Configure Regular Expressions, on page 259
- Monitoring Inspection Policies, on page 263
- History for Application Inspection, on page 264

Application Layer Protocol Inspection

Inspection engines are required for services that embed IP addressing information in the user data packet or that open secondary channels on dynamically assigned ports. These protocols require the ASA to do a deep packet inspection instead of passing the packet through the fast path. As a result, inspection engines can affect overall throughput. Several common inspection engines are enabled on the ASA by default, but you might need to enable others depending on your network.

The following topics explain application inspection in more detail.

When to Use Application Protocol Inspection

When a user establishes a connection, the ASA checks the packet against ACLs, creates an address translation, and creates an entry for the session in the fast path, so that further packets can bypass time-consuming checks. However, the fast path relies on predictable port numbers and does not perform address translations inside a packet.

Many protocols open secondary TCP or UDP ports. The initial session on a well-known port is used to negotiate dynamically assigned port numbers.

Other applications embed an IP address in the packet that needs to match the source address that is normally translated when it goes through the ASA.

If you use applications like these, then you need to enable application inspection.

When you enable application inspection for a service that embeds IP addresses, the ASA translates embedded addresses and updates any checksum or other fields that are affected by the translation.

When you enable application inspection for a service that uses dynamically assigned ports, the ASA monitors sessions to identify the dynamic port assignments, and permits data exchange on these ports for the duration of the specific session.

Inspection Policy Maps

You can configure special actions for many application inspections using an *inspection policy map*. These maps are optional: you can enable inspection for a protocol that supports inspection policy maps without configuring a map. These maps are needed only if you want something other than the default inspection actions.

An inspection policy map consists of one or more of the following elements. The exact options available for an inspection policy map depends on the application.

- Traffic matching criteria—You match application traffic to criteria specific to the application, such as a URL string, for which you then enable actions.

  For some traffic matching criteria, you use regular expressions to match text inside a packet. Be sure to create and test the regular expressions before you configure the policy map, either singly or grouped together in a regular expression class map.

- Inspection class map—Some inspection policy maps let you use an inspection class map to include multiple traffic matching criteria. You then identify the inspection class map in the inspection policy map and enable actions for the class as a whole. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that you can create more complex match criteria and you can reuse class maps. However, you cannot set different actions for different matches.

- Parameters—Parameters affect the behavior of the inspection engine.

The following topics provide more details.

Replacing an In-Use Inspection Policy Map

If you have an inspection enabled with a policy map in a service policy, replacing the policy map is a two-step process. First, you must remove the inspection. Then, you add it back with the new policy map name.

For example, to replace sip-map1 with sip-map2 in SIP inspection, use the following command sequence:

```
hostname(config)# policy-map test
hostname(config-pmap)# class sip
hostname(config-pmap-c)# no inspect sip sip-map1
hostname(config-pmap-c)# inspect sip sip-map2
```

How Multiple Traffic Classes are Handled

You can specify multiple inspection class maps or direct matches in the inspection policy map.

If a packet matches multiple different classes or direct matches, then the order in which the ASA applies the actions is determined by internal ASA rules, and not by the order they are added to the inspection policy map. The internal rules are determined by the application type and the logical progression of parsing a packet, and are not user-configurable. For example for HTTP traffic, parsing a Request Method field precedes parsing the Header Host Length field; an action for the Request Method field occurs before the action for the Header Host Length field. For example, the following match commands can be entered in any order, but the **match request method get** command is matched first.

```
match request header host length gt 100
  reset
match request method get
  log
```

If an action drops a packet, then no further actions are performed in the inspection policy map. For example, if the first action is to reset the connection, then it will never match any further match criteria. If the first action is to log the packet, then a second action, such as resetting the connection, can occur.

If a packet matches multiple match criteria that are the same, then they are matched in the order they appear in the policy map. For example, for a packet with the header length of 1001, it will match the first command below, and be logged, and then will match the second command and be reset. If you reverse the order of the two **match** commands, then the packet will be dropped and the connection reset before it can match the second **match** command; it will never be logged.

```
match request header length gt 100
  log
match request header length gt 1000
  reset
```

A class map is determined to be the same type as another class map or direct match based on the lowest priority match option in the class map (the priority is based on the internal rules). If a class map has the same type of lowest priority match option as another class map, then the class maps are matched according to the order they are added to the policy map. If the lowest priority match for each class map is different, then the class map with the higher priority match option is matched first. For example, the following three class maps contain two types of **match** commands: **match request-cmd** (higher priority) and **match filename** (lower priority). The ftp3 class map includes both commands, but it is ranked according to the lowest priority command, **match filename**. The ftp1 class map includes the highest priority command, so it is matched first, regardless of the order in the policy map. The ftp3 class map is ranked as being of the same priority as the ftp2 class map, which also contains the **match filename** command. They are matched according to the order in the policy map: ftp3 and then ftp2.

```
class-map type inspect ftp match-all ftp1
  match request-cmd get
class-map type inspect ftp match-all ftp2
  match filename regex abc
class-map type inspect ftp match-all ftp3
  match request-cmd get
  match filename regex abc

policy-map type inspect ftp ftp
  class ftp3
    log
  class ftp2
    log
  class ftp1
    log
```

Guidelines for Application Inspection

Failover

State information for multimedia sessions that require inspection are not passed over the state link for stateful failover. The exceptions are GTP, M3UA, and SIP, which are replicated over the state link. You must configure strict application server process (ASP) state checking in M3UA inspection to get stateful failover.

Clustering

The following inspections are not supported in clustering:

- CTIQBE
- H323, H225, and RAS
- IPsec passthrough
- MGCP
- MMP
- RTSP
- SCCP (Skinny)
- WAAS

IPv6

Supports IPv6 for the following inspections:

- Diameter
- DNS over UDP
- FTP
- GTP
- HTTP
- ICMP
- IPsec pass-through
- IPv6
- M3UA
- SCCP (Skinny)
- SCTP
- SIP
- SMTP
- VXLAN

Supports NAT64 for the following inspections:

- DNS over UDP
- FTP
- HTTP
- ICMP

- SCTP

Additional Guidelines

- Some inspection engines do not support PAT, NAT, outside NAT, or NAT between same security interfaces. For more information about NAT support, see Default Inspections and NAT Limitations, on page 249.
- For all the application inspections, the ASA limits the number of simultaneous, active data connections to 200 connections. For example, if an FTP client opens multiple secondary connections, the FTP inspection engine allows only 200 active connections and the 201 connection is dropped and the adaptive security appliance generates a system error message.
- Inspected protocols are subject to advanced TCP-state tracking, and the TCP state of these connections is not automatically replicated. While these connections are replicated to the standby unit, there is a best-effort attempt to re-establish a TCP state.
- If the system determines that a TCP connection requires inspection, the system clears all TCP options except for the MSS and selective-acknowledgment (SACK) options on the packets before inspecting them. Other options are cleared even if you allow them in a TCP map applied to the connections.
- TCP/UDP Traffic directed to the ASA (to an interface) is inspected by default. However, ICMP traffic directed to an interface is never inspected, even if you enable ICMP inspection. Thus, a ping (echo request) to an interface can fail under specific circumstances, such as when the echo request comes from a source that the ASA can reach through a backup default route.

Defaults for Application Inspection

The following topics explain the default operations for application inspection.

Default Inspections and NAT Limitations

By default, the configuration includes a policy that matches all default application inspection traffic and applies inspection to the traffic on all interfaces (a global policy). Default application inspection traffic includes traffic to the default ports for each protocol. You can only apply one global policy, so if you want to alter the global policy, for example, to apply inspection to non-standard ports, or to add inspections that are not enabled by default, you need to either edit the default policy or disable it and apply a new one.

The following table lists all inspections supported, the default ports used in the default class map, and the inspection engines that are on by default, shown in bold. This table also notes any NAT limitations. In this table:

- Inspection engines that are enabled by default for the default port are in bold.
- The ASA is in compliance with the indicated standards, but it does not enforce compliance on packets being inspected. For example, FTP commands are supposed to be in a particular order, but the ASA does not enforce the order.

Table 10: Supported Application Inspection Engines

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
CTIQBE	TCP/2748	No extended PAT. No NAT64. (Clustering) No static PAT.	—	—
DCERPC	TCP/135	No NAT64.	—	—
Diameter	TCP/3868 TCP/5868 (for TCP/TLS) SCTP/3868	No NAT/PAT.	RFC 6733	Requires the Carrier license.
DNS over UDP DNS over TCP	UDP/53 UDP/443 TCP/53	No NAT support is available for name resolution through WINS.	RFC 1123	You must enable DNS/TCP inspection in the DNS inspection policy map to inspect DNS over TCP. UDP/443 is used for Cisco Umbrella DNScrypt sessions only.
FTP	TCP/21	(Clustering) No static PAT.	RFC 959	—
GTP	UDP/3386 (GTPv0) UDP/2123 (GTPv1+)	No extended PAT. No NAT.	—	Requires the Carrier license.
H.323 H.225 and **RAS**	TCP/1720 UDP/1718 UDP (RAS) 1718-1719	(Clustering) No static PAT. No extended PAT. No NAT on same security interfaces. No NAT64.	ITU-T H.323, H.245, H225.0, Q.931, Q.932	—
HTTP	TCP/80	—	RFC 2616	Beware of MTU limitations stripping ActiveX and Java. If the MTU is too small to allow the Java or ActiveX tag to be included in one packet, stripping may not occur.
ICMP	ICMP	—	—	ICMP traffic directed to an ASA interface is never inspected.
ICMP ERROR	ICMP	—	—	—
ILS (LDAP)	TCP/389	No extended PAT. No NAT64.	—	—

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
Instant Messaging (IM)	Varies by client	No extended PAT. No NAT64.	RFC 3860	—
IP Options	RSVP	No NAT64.	RFC 791, RFC 2113	—
IPsec Pass Through	UDP/500	No PAT. No NAT64.	—	—
IPv6	—	No NAT64.	RFC 2460	—
LISP	—	No NAT or PAT.	—	—
M3UA	SCTP/2905	No NAT or PAT for embedded addresses.	RFC 4666	Requires the Carrier license.
MGCP	UDP/2427, 2727	No extended PAT. No NAT64. (Clustering) No static PAT.	RFC 2705bis-05	—
MMP	TCP/5443	No extended PAT. No NAT64.	—	—
NetBIOS Name Server over IP	UDP/137, 138 (Source ports)	No extended PAT. No NAT64.	—	NetBIOS is supported by performing NAT of the packets for NBNS UDP port 137 and NBDS UDP port 138.
PPTP	TCP/1723	No NAT64. (Clustering) No static PAT.	RFC 2637	—
RADIUS Accounting	UDP/1646	No NAT64.	RFC 2865	—
RSH	TCP/514	No PAT. No NAT64. (Clustering) No static PAT.	Berkeley UNIX	—
RTSP	TCP/554	No extended PAT. No NAT64. (Clustering) No static PAT.	RFC 2326, 2327, 1889	No handling for HTTP cloaking.

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
SCTP	SCTP	—	RFC 4960	Requires the Carrier license. Although you can do static network object NAT on SCTP traffic (no dyamic NAT/PAT), the inspection engine is not used for NAT.
SIP	TCP/5060 UDP/5060	No NAT/PAT on interfaces with the same, or lower to higher, security levels. No extended PAT. No NAT64 or NAT46. (Clustering) No static PAT.	RFC 2543	Does not handle TFTP uploaded Cisco IP Phone configurations under certain circumstances.
SKINNY (SCCP)	TCP/2000	No NAT on same security interfaces. No extended PAT. No NAT64, NAT46, or NAT66. (Clustering) No static PAT.	—	Does not handle TFTP uploaded Cisco IP Phone configurations under certain circumstances.
SMTP and **ESMTP**	TCP/25	No NAT64.	RFC 821, 1123	—
SNMP	UDP/161, 162 UDP/4161 on platforms that also run Secure Firewall eXtensible Operating System (FXOS).	No NAT or PAT.	RFC 1155, 1157, 1212, 1213, 1215	v.2 RFC 1902-1908; v.3 RFC 2570-2580. On FXOS platforms, if you configure SNMP, this inspection is enabled automatically, and you cannot disable it.
SQL*Net	TCP/1521	No extended PAT. No NAT64. (Clustering) No static PAT.	—	v.1 and v.2.
STUN	TCP/3478 UDP/3478	(WebRTC) Static NAT/PAT44 only. (Cisco Spark) Static NAT/PAT44 and 64; and dynamic NAT/PAT.	RFC 5245, 5389	—
Sun RPC	TCP/111 UDP/111	No extended PAT. No NAT64.	—	—
TFTP	UDP/69	No NAT64. (Clustering) No static PAT.	RFC 1350	Payload IP addresses are not translated.

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
WAAS	TCP/1- 65535	No extended PAT. No NAT64.	—	—
XDMCP	UDP/177	No extended PAT. No NAT64. (Clustering) No static PAT.	—	—
VXLAN	UDP/4789	Not applicable	RFC 7348	Virtual Extensible Local Area Network.

Default Inspection Policy Maps

Some inspection types use hidden default policy maps. For example, if you enable ESMTP inspection without specifying a map, _default_esmtp_map is used.

The default inspection is described in the sections that explain each inspection type. You can view these default maps using the **show running-config all policy-map** command.

DNS inspection is the only one that uses an explicitly-configured default map, preset_dns_map.

Configure Application Layer Protocol Inspection

You configure application inspection in service policies.

Inspection is enabled by default globally on all interfaces for some applications on their standard ports and protocols. See Default Inspections and NAT Limitations, on page 249 for more information on default inspections. A common method for customizing the inspection configuration is to customize the default global policy. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Before you begin

For some applications, you can perform special actions when you enable inspection by configuring inspection policy maps. The table later in this procedure shows which protocols allow inspection policy maps, with pointers to the instructions on configuring them. If you want to configure these advanced features, create the map before configuring inspection.

Procedure

Step 1 Unless you are adding inspection to an existing class map, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map dns_class_map
hostname(config-cmap)# match access-list dns
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (**match default-inspection-traffic**). You can have more than one inspection on the inspection_default class only, and you might want to simply edit the existing global policy that applies the inspection defaults. If you are using this class map in either the default policy or for a new service policy, you can skip this step. For detailed information on which class map to choose, see Choosing the Right Traffic Class for Inspection, on page 258.

For information on matching statements, see Create a Layer 3/4 Class Map for Through Traffic, on page 234. For RADIUS accounting inspection, which uses a management layer 3/4 class, see Configure RADIUS Accounting Inspection, on page 364.

Step 2 Add or edit a Layer 3/4 policy map that sets the actions to take with the class map traffic: **policy-map** *name*

Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

Step 3 Identify the L3/L4 class map you are using for inspection: **class** *name*

Example:

```
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify **inspection_default** for the *name*. Otherwise, you are specifying the class you created earlier in this procedure.

You can combine multiple class maps in the same policy if desired, so you can create one class map to match certain traffic, and another to match different traffic. However, if traffic matches a class map that contains an inspection command, and then matches another class map that also has an inspection command, only the first matching class is used. For example, SNMP matches the inspection_default class map. To enable SNMP inspection, enable SNMP inspection for the default class. Do not add another class that matches SNMP.

Step 4 Enable application inspection: **inspect** *protocol*

The *protocol* is one of the following values:

Table 11: Inspection Protocol Keywords

Keywords	Notes
ctiqbe	See CTIQBE Inspection, on page 305.
dcerpc [*map_name*]	See DCERPC Inspection, on page 265. If you added a DCERPC inspection policy map according to Configure a DCERPC Inspection Policy Map, on page 266, identify the map name in this command.

Keywords	Notes
diameter [*map_name*] [**tls-proxy** *proxy_name*]	See Diameter Inspection, on page 333. If you added a Diameter inspection policy map according to Configure a Diameter Inspection Policy Map, on page 343, identify the map name in this command. **tls-proxy** *proxy_name* identifies the TLS proxy to use for this inspection. You need a TLS proxy only if you want to enable inspection of encrypted traffic.
dns [*map_name*] [**dynamic-filter-snoop**]	See DNS Inspection, on page 268. If you added a DNS inspection policy map according to Configure DNS Inspection Policy Map, on page 269, identify the map name in this command. The default DNS inspection policy map name is "preset_dns_map." **dynamic-filter-snoop** enables dynamic filter snooping, which is used exclusively by the Botnet Traffic Filter. Include this keyword only if you use Botnet Traffic Filtering. We suggest that you enable DNS snooping only on interfaces where external DNS requests are going. Enabling DNS snooping on all UDP DNS traffic, including that going to an internal DNS server, creates unnecessary load on the ASA.
esmtp [*map_name*]	See SMTP and Extended SMTP Inspection, on page 294. If you added an ESMTP inspection policy map according to Configure an ESMTP Inspection Policy Map, on page 296, identify the map name in this command.
ftp [**strict** [*map_name*]]	See FTP Inspection, on page 273. Use the **strict** keyword to increase the security of protected networks by preventing web browsers from sending embedded commands in FTP requests. See Strict FTP, on page 274 for more information. If you added an FTP inspection policy map according to Configure an FTP Inspection Policy Map, on page 275, identify the map name in this command.
gtp [*map_name*]	See GTP Inspection Overview, on page 329. If you added a GTP inspection policy map according to Configure a GTP Inspection Policy Map, on page 337, identify the map name in this command.
h323 h225 [*map_name*]	See H.323 Inspection, on page 306. If you added an H323 inspection policy map according to Configure H.323 Inspection Policy Map, on page 308, identify the map name in this command.

Keywords	Notes
h323 ras [*map_name*]	See H.323 Inspection, on page 306. If you added an H323 inspection policy map according to Configure H.323 Inspection Policy Map, on page 308, identify the map name in this command.
http [*map_name*]	See HTTP Inspection, on page 278. If you added an HTTP inspection policy map according to Configure an HTTP Inspection Policy Map, on page 278, identify the map name in this command.
icmp	See ICMP Inspection, on page 282.
icmp error	See ICMP Error Inspection, on page 282.
ils	See ILS Inspection, on page 283.
im [*map_name*]	See Instant Messaging Inspection, on page 284. If you added an Instant Messaging inspection policy map, identify the map name in this command.
ip-options [*map_name*]	See IP Options Inspection, on page 286. If you added an IP Options inspection policy map according to Configure an IP Options Inspection Policy Map, on page 287, identify the map name in this command.
ipsec-pass-thru [*map_name*]	See IPsec Pass Through Inspection, on page 289. If you added an IPsec Pass Through inspection policy map according to Configure an IPsec Pass Through Inspection Policy Map, on page 289, identify the map name in this command.
ipv6 [*map_name*]	See IPv6 Inspection, on page 290. If you added an IPv6 inspection policy map according to Configure an IPv6 Inspection Policy Map, on page 291, identify the map name in this command.
lisp [*map_name*]	For detailed information on configuring LISP, including inspection, see the clustering chapter in the general configuration guide. If you added a LISP inspection policy map, identify the map name in this command.
m3ua [*map_name*]	See M3UA Inspection, on page 334. If you added an M3UA inspection policy map according to Configure an M3UA Inspection Policy Map, on page 359, identify the map name in this command.

Keywords	Notes
mgcp [*map_name*]	See MGCP Inspection, on page 311. If you added an MGCP inspection policy map according to Configure an MGCP Inspection Policy Map, on page 312, identify the map name in this command.
netbios [*map_name*]	See NetBIOS Inspection, on page 292. If you added a NetBIOS inspection policy map, identify the map name in this command.
pptp	See PPTP Inspection, on page 293.
radius-accounting *map_name*	See RADIUS Accounting Inspection Overview, on page 335. The **radius-accounting** keyword is only available for a management class map. You must specify a RADIUS accounting inspection policy map; see Configure a RADIUS Accounting Inspection Policy Map, on page 365.
rsh	See RSH Inspection, on page 294.
rtsp [*map_name*]	See RTSP Inspection, on page 314. If you added a RTSP inspection policy map according to Configure RTSP Inspection Policy Map, on page 315, identify the map name in this command.
sctp [*map_name*]	See SCTP Application Layer Inspection, on page 332. If you added an SCTP inspection policy map according to Configure an SCTP Inspection Policy Map, on page 342, identify the map name in this command.
sip [*map_name*] [**tls-proxy** *proxy_name*]	See SIP Inspection, on page 317. If you added a SIP inspection policy map according to Configure SIP Inspection Policy Map, on page 319, identify the map name in this command. **tls-proxy** *proxy_name* identifies the TLS proxy to use for this inspection. You need a TLS proxy only if you want to enable inspection of encrypted traffic.
skinny [*map_name*]	See Skinny (SCCP) Inspection, on page 323. If you added a Skinny inspection policy map according to Configure a Skinny (SCCP) Inspection Policy Map, on page 324, identify the map name in this command.
snmp [*map_name*]	See SNMP Inspection, on page 298. If you added an SNMP inspection policy map, identify the map name in this command.
sqlnet	See SQL*Net Inspection, on page 299.

Keywords	Notes
stun	See STUN Inspection, on page 326.
sunrpc	See Sun RPC Inspection, on page 300. The default class map includes UDP port 111; if you want to enable Sun RPC inspection for TCP port 111, you need to create a new class map that matches TCP port 111, add the class to the policy, and then apply the **inspect sunrpc** command to that class.
tftp	See TFTP Inspection, on page 301.
waas	Enables TCP option 33 parsing. Use when deploying Cisco Wide Area Application Services products.
xdmcp	See XDMCP Inspection, on page 302.
vxlan	See VXLAN Inspection, on page 302.

Note
If you are editing the default global policy (or any in-use policy) to use a different inspection policy map, you must remove the old inspection with the **no inspect** *protocol* command, and then re-add it with the new inspection policy map name.

Example:

```
hostname(config-class)# no inspect sip
hostname(config-class)# inspect sip sip-map
```

Step 5 If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy global_policy global
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Choosing the Right Traffic Class for Inspection

The default Layer 3/4 class map for through traffic is called “inspection_default.” It matches traffic using a special **match** command, **match default-inspection-traffic**, to match the default protocols and ports for each application protocol. This traffic class (along with **match any**, which is not typically used for inspection) matches both IPv4 and IPv6 traffic for inspections that support IPv6. See Guidelines for Application Inspection, on page 247 for a list of IPv6-enabled inspections.

You can specify a **match access-list** command along with the **match default-inspection-traffic** command to narrow the matched traffic to specific IP addresses. Because the **match default-inspection-traffic** command specifies the ports to match, any ports in the ACL are ignored.

Tip We suggest that you only inspect traffic on ports on which you expect application traffic; if you inspect all traffic, for example using **match any**, the ASA performance can be impacted.

If you want to match non-standard ports, then create a new class map for the non-standard ports. See Default Inspections and NAT Limitations, on page 249 for the standard ports for each inspection engine. You can combine multiple class maps in the same policy if desired, so you can create one class map to match certain traffic, and another to match different traffic. However, if traffic matches a class map that contains an inspection command, and then matches another class map that also has an inspection command, only the first matching class is used. For example, SNMP matches the inspection_default class. To enable SNMP inspection, enable SNMP inspection for the default class. Do not add another class that matches SNMP.

For example, to limit inspection to traffic from 10.1.1.0 to 192.168.1.0 using the default class map, enter the following commands:

```
hostname(config)# access-list inspect extended permit ip 10.1.1.0 255.255.255.0
192.168.1.0 255.255.255.0
hostname(config)# class-map inspection_default
hostname(config-cmap)# match access-list inspect
```

View the entire class map using the following command:

```
hostname(config-cmap)# show running-config class-map inspection_default
!
class-map inspection_default
 match default-inspection-traffic
 match access-list inspect
!
```

To inspect FTP traffic on port 21 as well as 1056 (a non-standard port), create an ACL that specifies the ports, and assign it to a new class map:

```
hostname(config)# access-list ftp_inspect extended permit tcp any any eq 21
hostname(config)# access-list ftp_inspect extended permit tcp any any eq 1056
hostname(config)# class-map new_inspection
hostname(config-cmap)# match access-list ftp_inspect
```

Configure Regular Expressions

Regular expressions define pattern matching for text strings. You can use these expressions in some protocol inspection maps to match packets based on strings such as URLs or the contents of particular header fields.

Create a Regular Expression

A regular expression matches text strings either literally as an exact string, or by using *metacharacters* so that you can match multiple variants of a text string. You can use a regular expression to match the content of certain application traffic; for example, you can match a URL string inside an HTTP packet.

Before you begin

Use **Ctrl+V** to escape all of the special characters in the CLI, such as question mark (?) or a tab. For example, type **d[Ctrl+V]?g** to enter **d?g** in the configuration.

See the **regex** command in the command reference for performance impact information when matching a regular expression to packets. In general, matching against long input strings, or trying to match a large number of regular expressions, will reduce system performance.



Note As an optimization, the ASA searches on the deobfuscated URL. Deobfuscation compresses multiple forward slashes (/) into a single slash. For strings that commonly use double slashes, like "http://", be sure to search for "http:/" instead.

The following table lists the metacharacters that have special meanings.

Table 12: Regular Expression Metacharacters

Character	Description	Notes
.	Dot	Matches any single character. For example, **d.g** matches dog, dag, dtg, and any word that contains those characters, such as doggonnit.
(*exp*)	Subexpression	A subexpression segregates characters from surrounding characters, so that you can use other metacharacters on the subexpression. For example, **d(o|a)g** matches dog and dag, but **do|ag** matches do and ag. A subexpression can also be used with repeat quantifiers to differentiate the characters meant for repetition. For example, **ab(xy){3}z** matches abxyxyxyz.
|	Alternation	Matches either expression it separates. For example, **dog|cat** matches dog or cat.
?	Question mark	A quantifier that indicates that there are 0 or 1 of the previous expression. For example, **lo?se** matches lse or lose.
*	Asterisk	A quantifier that indicates that there are 0, 1 or any number of the previous expression. For example, **lo*se** matches lse, lose, loose, and so on.
+	Plus	A quantifier that indicates that there is at least 1 of the previous expression. For example, **lo+se** matches lose and loose, but not lse.
{*x*} or {*x*,}	Minimum repeat quantifier	Repeat at least *x* times. For example, **ab(xy){2,}z** matches abxyxyz, abxyxyxyz, and so on.
[*abc*]	Character class	Matches any character in the brackets. For example, **[abc]** matches a, b, or c.

Character	Description	Notes
[^*abc*]	Negated character class	Matches a single character that is not contained within the brackets. For example, **[^abc]** matches any character other than a, b, or c. **[^A-Z]** matches any single character that is not an uppercase letter.
[*a-c*]	Character range class	Matches any character in the range. **[a-z]** matches any lowercase letter. You can mix characters and ranges: **[abcq-z]** matches a, b, c, q, r, s, t, u, v, w, x, y, z, and so does [**a-cq-z]**. The dash (-) character is literal only if it is the last or the first character within the brackets: **[abc-]** or **[-abc]**.
“”	Quotation marks	Preserves trailing or leading spaces in the string. For example, **“ test”** preserves the leading space when it looks for a match.
^	Caret	Specifies the beginning of a line.
\	Escape character	When used with a metacharacter, matches a literal character. For example, **\[** matches the left square bracket.
char	Character	When character is not a metacharacter, matches the literal character.
\r	Carriage return	Matches a carriage return 0x0d.
\n	Newline	Matches a new line 0x0a.
\t	Tab	Matches a tab 0x09.
\f	Formfeed	Matches a form feed 0x0c.
\x*NN*	Escaped hexadecimal number	Matches an ASCII character using hexadecimal (exactly two digits).
NNN	Escaped octal number	Matches an ASCII character as octal (exactly three digits). For example, the character 040 represents a space.

Procedure

Step 1 Test a regular expression to make sure it matches what you think it will match: **test regex** *input_text regular_expression*

Where the *input_text* argument is a string you want to match using the regular expression, up to 201 characters in length. The *regular_expression* argument can be up to 100 characters in length.

Use **Ctrl+V** to escape all of the special characters in the CLI. For example, to enter a tab in the input text in the **test regex** command, you must enter **test regex “test[Ctrl+V Tab]” “test\t”**.

If the regular expression matches the input text, you see the following message:

```
INFO: Regular expression match succeeded.
```

If the regular expression does not match the input text, you see the following message:

```
INFO: Regular expression match failed.
```

Step 2 To add a regular expression after you tested it, enter the following command: **regex** *name regular_expression*

Where the *name* argument can be up to 40 characters in length. The *regular_expression* argument can be up to 100 characters in length.

Examples

The following example creates two regular expressions for use in an inspection policy map:

```
hostname(config)# regex url_example example\.com
hostname(config)# regex url_example2 example2\.com
```

Create a Regular Expression Class Map

A regular expression class map identifies one or more regular expression. It is simply a collection of regular expression objects. You can use a regular expression class map in many cases in replace of a regular expression object.

Procedure

Step 1 Create the regular expression class map: **class-map type regex match-any** *class_map_name*

Where *class_map_name* is a string up to 40 characters in length. The name "class-default" is reserved. All types of class maps use the same name space, so you cannot reuse a name already used by another type of class map.

The **match-any** keyword specifies that the traffic matches the class map if it matches at least one of the regular expressions.

Step 2 (Optional) Add a description to the class map: **description** *string*

Step 3 Identify the regular expressions you want to include by entering the following command for each regular expression: **match regex** *regex_name*

Examples

The following example creates two regular expressions, and adds them to a regular expression class map. Traffic matches the class map if it includes the string "example.com" or "example2.com."

```
hostname(config)# regex url_example example\.com
hostname(config)# regex url_example2 example2\.com
hostname(config)# class-map type regex match-any URLs
hostname(config-cmap)# match regex url_example
hostname(config-cmap)# match regex url_example2
```

Monitoring Inspection Policies

To monitor inspection service policies, enter the following commands. See the command reference on Cisco.com for detailed syntax and examples.

- **show service-policy inspect** *protocol*

  Displays statistics for inspection service policies. The *protocol* is the protocol from the inspect command, for example **dns**. However, not all inspection protocols show statistics with this command. For example:

```
asa# show service-policy inspect dns

Global policy:
  Service-policy: global_policy
    Class-map: inspection_default
      Inspect: dns preset_dns_map, packet 0, lock fail 0, drop 0, reset-drop 0,
5-min-pkt-rate 0 pkts/sec, v6-fail-close 0
        message-length maximum client auto, drop 0
        message-length maximum 512, drop 0
        dns-guard, count 0
        protocol-enforcement, drop 0
        nat-rewrite, count 0
asa#
```

- **show conn**

  Shows current connections for traffic passing through the device. This command has a wide range of keywords so that you can get information about various protocols.

- Additional commands for specific inspected protocols:
  - **show ctiqbe**

    Displays information about the media connections allocated by the CTIQBE inspection engine

  - **show h225**

    Displays information for H.225 sessions.

  - **show h245**

    Displays information for H.245 sessions established by endpoints using slow start.

  - **show h323 ras**

    Displays connection information for H.323 RAS sessions established between a gatekeeper and its H.323 endpoint.

  - **show mgcp** {**commands** | **sessions** }

Displays the number of MGCP commands in the command queue or the number of existing MGCP sessions.

- **show sip**

  Displays information for SIP sessions.

- **show skinny**

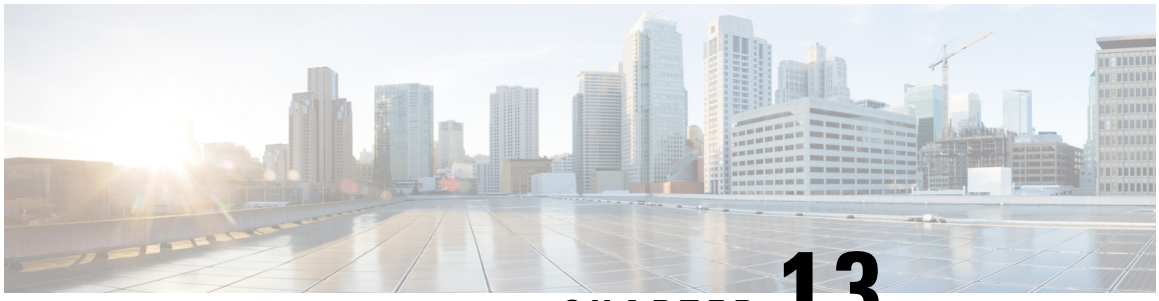
  Displays information for Skinny (SCCP) sessions.

- **show sunrpc-server active**

  Displays the pinholes opened for Sun RPC services.

History for Application Inspection

Feature Name	Releases	Description
Inspection policy maps	7.2(1)	The inspection policy map was introduced. The following command was introduced: **class-map type inspect**.
Regular expressions and policy maps	7.2(1)	Regular expressions and policy maps were introduced to be used under inspection policy maps. The following commands were introduced: **class-map type regex**, **regex**, **match regex**.
Match any for inspection policy maps	8.0(2)	The **match any** keyword was introduced for use with inspection policy maps: traffic can match one or more criteria to match the class map. Formerly, only **match all** was available.



CHAPTER 13

Inspection of Basic Internet Protocols

The following topics explain application inspection for basic Internet protocols. For information on why you need to use inspection for certain protocols, and the overall methods for applying inspection, see Getting Started with Application Layer Protocol Inspection, on page 245.

- DCERPC Inspection, on page 265
- DNS Inspection, on page 268
- FTP Inspection, on page 273
- HTTP Inspection, on page 278
- ICMP Inspection, on page 282
- ICMP Error Inspection, on page 282
- ILS Inspection, on page 283
- Instant Messaging Inspection, on page 284
- IP Options Inspection, on page 286
- IPsec Pass Through Inspection, on page 289
- IPv6 Inspection, on page 290
- NetBIOS Inspection, on page 292
- PPTP Inspection, on page 293
- RSH Inspection, on page 294
- SMTP and Extended SMTP Inspection, on page 294
- SNMP Inspection, on page 298
- SQL*Net Inspection, on page 299
- Sun RPC Inspection, on page 300
- TFTP Inspection, on page 301
- XDMCP Inspection, on page 302
- VXLAN Inspection, on page 302
- History for Basic Internet Protocol Inspection, on page 302

DCERPC Inspection

DCERPC inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. You can simply edit the default global inspection policy to add DCERPC inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

The following sections describe the DCERPC inspection engine.

DCERPC Overview

Microsoft Remote Procedure Call (MSRPC), based on DCERPC, is a protocol widely used by Microsoft distributed client and server applications that allows software clients to execute programs on a server remotely.

This typically involves a client querying a server called the Endpoint Mapper listening on a well known port number for the dynamically allocated network information of a required service. The client then sets up a secondary connection to the server instance providing the service. The security appliance allows the appropriate port number and network address and also applies NAT, if needed, for the secondary connection.

The DCERPC inspection engine inspects for native TCP communication between the EPM and client on well known TCP port 135. Map and lookup operations of the EPM are supported for clients. Client and server can be located in any security zone. The embedded server IP address and Port number are received from the applicable EPM response messages. Since a client may attempt multiple connections to the server port returned by EPM, multiple use of pinholes are allowed, which have configurable timeouts.

DCE inspection supports the following universally unique identifiers (UUIDs) and messages:

- End point mapper (EPM) UUID. All EPM messages are supported.
- ISystemMapper UUID (non-EPM). Supported messages are:
  - RemoteCreateInstance opnum4
  - RemoteGetClassObject opnum3
- OxidResolver UUID (non-EPM). Supported message is:
  - ServerAlive2 opnum5
- Any message that does not contain an IP address or port information because these messages do not require inspection.

Configure a DCERPC Inspection Policy Map

To specify additional DCERPC inspection parameters, create a DCERPC inspection policy map. You can then apply the inspection policy map when you enable DCERPC inspection.

When defining traffic matching criteria, you can either create a class map or include the match statements directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that you can reuse class maps.

Procedure

Step 1 (Optional) Create a DCERPC inspection class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect dcerpc** [**match-all** | **match-any**] *class_map_name*

Where *class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one **match** statement. The CLI enters class-map configuration mode.

b) Specify the traffic on which you want to perform actions using the following **match** command. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **uuid** *type*—Matches the universally unique identifier (UUID) of the DCERPC message. The *type* can be one of the following:
  - **ms-rpc-epm**—Matches Microsoft RPC EPM messages.
  - **ms-rpc-isystemactivator**—Matches ISystemMapper messages.
  - **ms-rpc-oxidresolver**—Matches OxidResolver messages.

c) Enter **exit** to leave class map configuration mode.

Step 2 Create a DCERPC inspection policy map: **policy-map type inspect dcerpc** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created a DCERPC class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for DCERPC class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **reset** [**log**]—Drop the packet, close the connection, and send a TCP reset to the server or client.
- **log**—Send a system log message. You can use this option alone or with one of the other actions.

You can specify multiple **class** or **match** commands in the policy map.

Example:

```
hostname(config)# policy-map type inspect dcerpc dcerpc-map
hostname(config-pmap)# match uuid ms-rpc-epm
hostname(config-pmap-c)# log
```

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **timeout pinhole** *hh:mm:ss*—Configures the timeout for DCERPC pinholes and override the global system pinhole timeout of two minutes. The timeout can be from 00:00:01 to 119:00:00.
- **endpoint-mapper** [**epm-service-only**] [**lookup-operation** [**timeout** *hh:mm:ss*]]—Configures options for the endpoint mapper traffic. The **epm-service-only** keyword enforces endpoint mapper service during binding so that only its service traffic is processed. The **lookup-operation** keyword enables the lookup operation of the endpoint mapper service. You can configure the timeout for pinholes generated from the lookup operation. If no timeout is configured for the lookup operation, the timeout pinhole command or the default is used.

Examples

The following example shows how to define a DCERPC inspection policy map with the timeout configured for DCERPC pinholes.

```
hostname(config)# policy-map type inspect dcerpc dcerpc_map
hostname(config-pmap)# timeout pinhole 0:10:00

hostname(config)# class-map dcerpc
hostname(config-cmap)# match port tcp eq 135

hostname(config)# policy-map global-policy
hostname(config-pmap)# class dcerpc
hostname(config-pmap-c)# inspect dcerpc dcerpc-map

hostname(config)# service-policy global-policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

DNS Inspection

DNS inspection is enabled by default. You need to configure it only if you want non-default processing. The following sections describe DNS application inspection.

Defaults for DNS Inspection

DNS inspection is enabled by default, using the preset_dns_map inspection class map:

- The maximum DNS message length is 512 bytes.
- DNS over TCP inspection is disabled.
- The maximum client DNS message length is automatically set to match the Resource Record.

- DNS Guard is enabled, so the ASA tears down the DNS session associated with a DNS query as soon as the DNS reply is forwarded by the ASA. The ASA also monitors the message exchange to ensure that the ID of the DNS reply matches the ID of the DNS query.
- Translation of the DNS record based on the NAT configuration is enabled.
- Protocol enforcement is enabled, which enables DNS message format check, including domain name length of no more than 255 characters, label length of 63 characters, compression, and looped pointer check.

See the following default DNS inspection commands:

```
class-map inspection_default
 match default-inspection-traffic
policy-map type inspect dns preset_dns_map
 parameters
    message-length maximum client auto
    message-length maximum 512
    dns-guard
    protocol-enforcement
    nat-rewrite
policy-map global_policy
 class inspection_default
  inspect dns preset_dns_map
! ...
service-policy global_policy global
```



Note Elevated CPU utilization is likely to occur in Cisco ASA when a large number of packets is inspected per second. As the number of packets being processed increases—especially when the device nears its maximum throughput or connection capacity—the CPU usage too rises. This occurs because ASA must perform more operations per second, which increases CPU resource consumption.

Configure DNS Inspection Policy Map

You can create a DNS inspection policy map to customize DNS inspection actions if the default inspection behavior is not sufficient for your network.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create a DNS inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string "example.com," then any traffic that includes "example.com" does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect dns** [**match-all** | **match-any**] *class_map_name*

Where *class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one **match** statement. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **header-flag** [**eq**] {*f_name* [*f_name...*] | *f_value*}—Matches the DNS flag. The *f_name* argument is the DNS flag name, one of the following: **AA** (Authoritative Answer), **QR** (Query), **RA** (Recursion Available), **RD** (Recursion Desired), **TC** (Truncation). The *f_value* argument is the 16-bit value in hex starting with 0x, from 0x0 to 0xffff. The eq keyword specifies an exact match (match all); without the **eq** keyword, the packet only needs to match one of the specified headers (match any). For example, **match header-flag AA QR**.
- **match** [**not**] **dns-type {eq** {*t_name* | *t_value*} | **range** *t_value1 t_value2*}—Matches the DNS type. The *t_name* argument is the DNS type name, one of the following: **A** (IPv4 address), **AXFR** (full zone transfer), **CNAME** (canonical name), **IXFR** (incremental zone transfer), **NS** (authoritative name server), **SOA** (start of a zone of authority) or **TSIG** (transaction signature). The t_value arguments are arbitrary values in the DNS type field (0-65535). The range keyword specifies a range, and the eq keyword specifies an exact match. For example: **match dns-type eq A**.
- **match** [**not**] **dns-class {eq** {**in** | *c_value*} | **range** *c_value1 c_value2*}—Matches the DNS class. The class is either **in** (for Internet) or c_value, an arbitrary value from 0 to 65535 in the DNS class field. The range keyword specifies a range, and the eq keyword specifies an exact match. For example: **match dns-class eq in**.
- **match** [**not**] {**question** | **resource-record** {**answer** | **authority** | **additional**}}—Matches a DNS question or resource record. The **question** keyword specifies the question portion of a DNS message. The resource-record keyword specifies one of these sections of the resource record: **answer**, authority, or **additional**. For example: **match resource-record answer**.
- **match** [**not**] **domain-name regex** {*regex_name* | **class** *class_name*}—Matches the DNS message domain name list against the specified regular expression or regular expression class.

d) Enter **exit** to leave class map configuration mode.

Step 2 Create a DNS inspection policy map: **policy-map type inspect dns** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created a DNS class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for DNS class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop** [**log**]—Drop all packets that match.
- **drop-connection** [**log**]—Drop the packet and close the connection.
- **mask** [**log**]—Mask out the matching portion of the packet. This action is available for header flag matches only.
- **log**—Send a system log message. You can use this option alone or with one of the other actions.
- **enforce-tsig** [**drop**] [**log**]—Enforce the presence of the TSIG resource record in a message. You can drop a packet without the TSIG resource record, log it, or drop and log it. You can use this option in conjunction with the mask action for header flag matches; otherwise, this action is exclusive with the other actions.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

Example:

```
hostname(config)# policy-map type inspect dns dns-map
hostname(config-pmap)# class dns-class-map
hostname(config-pmap-c)# drop
hostname(config-pmap-c)# match header-flag eq aa
hostname(config-pmap-c)# drop log
```

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option.

- **dnscrypt**—Enables DNScrypt to encrypt connections between the device and Cisco Umbrella. Enabling DNScrypt starts the key-exchange thread with the Umbrella resolver. The key-exchange thread performs the handshake with the resolver every hour and updates the device with a new secret key. Because DNScrypt uses UDP/443, you must ensure that the class map used for DNS inspection includes that port. Note that the default inspection class already includes UDP/443 for DNS inspection.

- **dns-guard**—Enables DNS Guard. The ASA tears down the DNS session associated with a DNS query as soon as the DNS reply is forwarded by the ASA. The ASA also monitors the message exchange to ensure that the ID of the DNS reply matches the ID of the DNS query.
- **id-mismatch count** *number* **duration** *seconds* **action log**—Enables logging for excessive DNS ID mismatches, where the **count** *number* **duration** *seconds* arguments specify the maximum number of mismatch instances per second before a system message log is sent.
- **id-randomization**—Randomizes the DNS identifier for a DNS query.
- **message-length maximum** {*length* | **client** {*length* | **auto**} | **server** {*length* | **auto**}}—Sets the maximum DNS message length, from 512 to 65535 bytes. You can also set the maximum length for client or server messages. The **auto** keyword sets the maximum length to the value in the Resource Record.
- **nat-rewrite**—Translates the DNS record based on the NAT configuration.
- **protocol-enforcement**—Enables DNS message format check, including domain name length of no more than 255 characters, label length of 63 characters, compression, and looped pointer check.
- **tcp-inspection**—Enables inspection of DNS over TCP traffic. Ensure that DNS/TCP port 53 traffic is part of the class to which you apply DNS inspection. The inspection default class includes TCP/53.
- **tsig enforced action** {[**drop**] [**log**]}—Requires a TSIG resource record to be present. You can **drop** a non-conforming packet, **log** the packet, or both.
- **umbrella** [**tag** *umbrella_policy*] [**fail-open**]—Enables Cisco Umbrella and optionally specifies the name of the Cisco Umbrella policy (**tag**) to apply to the device. If you do not specify a policy, the default policy is applied. For more information, see Cisco Umbrella, on page 97.

  Include the **fail-open** keyword if you want DNS resolution to work if the Umbrella DNS server is unavailable. When failing open, if the Cisco Umbrella DNS server is unavailable, Umbrella disables itself on this policy map and allows DNS requests to go to the other DNS servers configured on the system, if any. When the Umbrella DNS servers are available again, the policy map resumes using them. If you do not include this option, DNS requests continue to go to the unreachable Umbrella resolver, so they will not get a response.

Example:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)# dns-guard
hostname(config-pmap-p)# message-length maximum 1024
hostname(config-pmap-p)# nat-rewrite
hostname(config-pmap-p)# protocol-enforcement
```

Example

The following example shows a how to use a new inspection policy map in the global default configuration:

```
regex domain_example "example\.com"
regex domain_foo "foo\.com"
```

```
! define the domain names that the server serves
class-map type regex match-any my_domains
   match regex domain_example
   match regex domain_foo

! Define a DNS map for query only
class-map type inspect dns match-all pub_server_map
   match not header-flag QR
   match question
   match not domain-name regex class my_domains

policy-map type inspect dns new_dns_map
    class pub_server_map
        drop log
    match header-flag RD
        mask log
    parameters
        message-length maximum client auto
        message-length maximum 512
        dns-guard
        protocol-enforcement
        nat-rewrite

policy-map global_policy
 class inspection_default
  no inspect dns preset_dns_map
  inspect dns new_dns_map
service-policy global_policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

FTP Inspection

FTP inspection is enabled by default. You need to configure it only if you want non-default processing. The following sections describe the FTP inspection engine.

FTP Inspection Overview

The FTP application inspection inspects the FTP sessions and performs four tasks:

- Prepares dynamic secondary data connection channels for FTP data transfer. Ports for these channels are negotiated through PORT or PASV commands. The channels are allocated in response to a file upload, a file download, or a directory listing event.
- Tracks the FTP command-response sequence.
- Generates an audit trail.
  - Audit record 303002 is generated for each file that is retrieved or uploaded.
  - Audit record 201005 is generated if the secondary dynamic channel preparation failed due to memory shortage.

- Translates the embedded IP address.

Note If you disable FTP inspection, outbound users can start connections only in passive mode, and all inbound FTP is disabled.

Strict FTP

Strict FTP increases the security of protected networks by preventing web browsers from sending embedded commands in FTP requests. To enable strict FTP, include the strict option with the **inspect ftp** command.

When you use strict FTP, you can optionally specify an FTP inspection policy map to specify FTP commands that are not permitted to pass through the ASA.

Strict FTP inspection enforces the following behavior:

- An FTP command must be acknowledged before the ASA allows a new command.
- The ASA drops connections that send embedded commands.
- The 227 and PORT commands are checked to ensure they do not appear in an error string.

Caution Using strict FTP may cause the failure of FTP clients that are not strictly compliant with FTP RFCs. Additionally, you must ensure you apply the inspection to your FTP ports only (TCP/21 is the normal FTP port). Strict FTP inspection applied to non-FTP traffic can result in unexpected traffic loss, especially HTTP traffic.

With strict FTP inspection, each FTP command and response sequence is tracked for the following anomalous activity:

- Truncated command—Number of commas in the PORT and PASV reply command is checked to see if it is five. If it is not five, then the PORT command is assumed to be truncated and the TCP connection is closed.
- Incorrect command—Checks the FTP command to see if it ends with <CR><LF> characters, as required by the RFC. If it does not, the connection is closed.
- Size of RETR and STOR commands—These are checked against a fixed constant. If the size is greater, then an error message is logged and the connection is closed.
- Command spoofing—The PORT command should always be sent from the client. The TCP connection is denied if a PORT command is sent from the server.
- Reply spoofing—PASV reply command (227) should always be sent from the server. The TCP connection is denied if a PASV reply command is sent from the client. This prevents the security hole when the user executes “227 xxxxx a1, a2, a3, a4, p1, p2.”
- TCP stream editing—The ASA closes the connection if it detects TCP stream editing.

- Invalid port negotiation—The negotiated dynamic port value is checked to see if it is less than 1024. As port numbers in the range from 1 to 1024 are reserved for well-known connections, if the negotiated port falls in this range, then the TCP connection is freed.
- Command pipelining—The number of characters present after the port numbers in the PORT and PASV reply command is cross checked with a constant value of 8. If it is more than 8, then the TCP connection is closed.
- The ASA replaces the FTP server response to the SYST command with a series of Xs to prevent the server from revealing its system type to FTP clients. To override this default behavior, use the **no mask-syst-reply** command in the FTP map.

Configure an FTP Inspection Policy Map

FTP command filtering and security checks are provided using strict FTP inspection for improved security and control. Protocol conformance includes packet length checks, delimiters and packet format checks, command terminator checks, and command validation.

Blocking FTP based on user values is also supported so that it is possible for FTP sites to post files for download, but restrict access to certain users. You can block FTP connections based on file type, server name, and other attributes. System message logs are generated if an FTP connection is denied after inspection.

If you want FTP inspection to allow FTP servers to reveal their system type to FTP clients, and limit the allowed FTP commands, then create and configure an FTP inspection policy map. You can then apply the map when you enable FTP inspection.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create an FTP inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect ftp** [**match-all** | **match-any**] *class_map_name*

Where *class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic

matches the class map if it matches at least one **match** statement. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **filename regex** {*regex_name* | **class** *class_name*}—Matches the filename in the FTP transfer against the specified regular expression or regular expression class.
- **match** [**not**] **filetype regex** {*regex_name* | **class** *class_name*}—Matches the file type in the FTP transfer against the specified regular expression or regular expression class.
- **match** [**not**] **request-command** *ftp_command* [*ftp_command*...]—Matches the FTP command, one or more of the following:
  - **APPE**—Append to a file.
  - **CDUP**—Changes to the parent directory of the current working directory.
  - **DELE**—Delete a file on the server.
  - **GET**—Gets a file from the server.
  - **HELP**—Provides help information.
  - **MKD**—Makes a directory on the server.
  - **PUT**—Sends a file to the server.
  - **RMD**—Deletes a directory on the server.
  - **RNFR**—Specifies the “rename-from” filename.
  - **RNTO**—Specifies the “rename-to” filename.
  - **SITE**—Used to specify a server-specific command. This is usually used for remote administration.
  - **STOU**—Stores a file using a unique file name.
- **match** [**not**] **server regex** {*regex_name* | **class** *class_name*}—Matches the FTP server name against the specified regular expression or regular expression class.
- **match** [**not**] **username regex** {*regex_name* | **class** *class_name*}—Matches the FTP username against the specified regular expression or regular expression class.

d) Enter **exit** to leave class map configuration mode.

Step 2 Create an FTP inspection policy map: **policy-map type inspect ftp** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created an FTP class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for FTP class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering the following command:

- **reset** [**log**]—Drop the packet, close the connection, and send a TCP reset to the server or client. Add the **log** keyword to send a system log message.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **mask-banner**—Masks the greeting banner from the FTP server.
- **mask-syst-reply**—Masks the reply to **syst** command.

Example

Before submitting a username and password, all FTP users are presented with a greeting banner. By default, this banner includes version information useful to hackers trying to identify weaknesses in a system. The following example shows how to mask this banner:

```
hostname(config)# policy-map type inspect ftp mymap
hostname(config-pmap)# parameters
hostname(config-pmap-p)# mask-banner

hostname(config)# class-map match-all ftp-traffic
hostname(config-cmap)# match port tcp eq ftp

hostname(config)# policy-map ftp-policy
hostname(config-pmap)# class ftp-traffic
hostname(config-pmap-c)# inspect ftp strict mymap

hostname(config)# service-policy ftp-policy interface inside
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

HTTP Inspection

HTTP inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. However, the default inspect class does include the default HTTP ports, so you can simply edit the default global inspection policy to add HTTP inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

The following sections describe the HTTP inspection engine.

HTTP Inspection Overview

Use the HTTP inspection engine to protect against specific attacks and other threats that are associated with HTTP traffic.

HTTP application inspection scans HTTP headers and body, and performs various checks on the data. These checks prevent various HTTP constructs, content types, and tunneling and messaging protocols from traversing the security appliance.

The enhanced HTTP inspection feature, which is also known as an application firewall and is available when you configure an HTTP inspection policy map, can help prevent attackers from using HTTP messages for circumventing network security policy.

HTTP application inspection can block tunneled applications and non-ASCII characters in HTTP requests and responses, preventing malicious content from reaching the web server. Size limiting of various elements in HTTP request and response headers, URL blocking, and HTTP server header type spoofing are also supported.

Enhanced HTTP inspection verifies the following for all HTTP messages:

- Conformance to RFC 2616
- Use of RFC-defined methods only.
- Compliance with the additional criteria.

Configure an HTTP Inspection Policy Map

To specify actions when a message violates a parameter, create an HTTP inspection policy map. You can then apply the inspection policy map when you enable HTTP inspection.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create an HTTP inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string "example.com," then any traffic that includes "example.com" does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect http** [**match-all** | **match-any**] *class_map_name*

Where *class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one **match** statement. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **req-resp content-type mismatch**—Matches traffic with a content-type field in the HTTP response that does not match the accept field in the corresponding HTTP request message.
- **match** [**not**] **request args regex** {*regex_name* | **class** *class_name*}—Matches text found in the HTTP request message arguments against the specified regular expression or regular expression class.
- **match** [**not**] **request body** {**regex** {*regex_name* | **class** *class_name*} | **length gt** *bytes*}—Matches text found in the HTTP request message body against the specified regular expression or regular expression class, or messages where the request body is greater than the specified length.
- **match** [**not**] **request header** {*field* | **regex** *regex_name*} **regex** {*regex_name* | **class** *class_name*}—Matches the content of a field in the HTTP request message header against the specified regular expression or regular expression class. You can specify the field name explicitly or match the field name to a regular expression. Field names are: accept, accept-charset, accept-encoding, accept-language, allow, authorization, cache-control, connection, content-encoding, content-language, content-length, content-location, content-md5, content-range, content-type, cookie, date, expect, expires, from, host, if-match, if-modified-since, if-none-match, if-range, if-unmodified-since, last-modified, max-forwards, pragma, proxy-authorization, range, referer, te, trailer, transfer-encoding, upgrade, user-agent, via, warning.
- **match** [**not**] **request header** {*field* | **regex** {*regex_name* | **class** *class_name*}} {**length gt** *bytes* | **count gt** *number*}—Matches the length of the specified fields in the HTTP request message header,

or the overall number of fields (count) in the header. You can specify the field name explicitly or match the field name to a regular expression or regular expression class. Field names are listed in the previous bullet.

- **match** [**not**] **request header** {**length gt** *bytes* | **count gt** *number* | **non-ascii**}—Matches the overall length of the HTTP request message header, or the overall number of fields (count) in the header, or headers that have non-ASCII characters.
- **match** [**not**] **request method** {*method* | **regex** {*regex_name* | **class** *class_name*}}—Matches the HTTP request method. You can specify the method explicitly or match the method to a regular expression or regular expression class. Methods are: bcopy, bdelete, bmove, bpropfind, bproppatch, connect, copy, delete, edit, get, getattribute, getattributenames, getproperties, head, index, lock, mkcol, mkdir, move, notify, options, poll, post, propfind, proppatch, put, revadd, revlabel, revlog, revnum, save, search, setattribute, startrev, stoprev, subscribe, trace, unedit, unlock, unsubscribe.
- **match** [**not**] **request uri** {**regex** {*regex_name* | **class** *class_name*} | **length gt** *bytes*}—Matches text found in the HTTP request message URI against the specified regular expression or regular expression class, or messages where the request URI is greater than the specified length.
- **match** [**not**] **response body** {**active-x** | **java-applet** | **regex** {*regex_name* | **class** *class_name*}}—Matches text found in the HTTP response message body against the specified regular expression or regular expression class, or comments out Java applet and Active X object tags in order to filter them.
- **match** [**not**] **response body length gt** *bytes*—Matches HTTP response messages where the body is greater than the specified length.
- **match** [**not**] **response header** {*field* | **regex** *regex_name*} **regex** {*regex_name* | **class** *class_name*}—Matches the content of a field in the HTTP response message header against the specified regular expression or regular expression class. You can specify the field name explicitly or match the field name to a regular expression. Field names are: accept-ranges, age, allow, cache-control, connection, content-encoding, content-language, content-length, content-location, content-md5, content-range, content-type, date, etag, expires, last-modified, location, pragma, proxy-authenticate, retry-after, server, set-cookie, trailer, transfer-encoding, upgrade, vary, via, warning, www-authenticate.
- **match** [**not**] **response header** {*field* | **regex** {*regex_name* | **class** *class_name*}} {**length gt** *bytes* | **count gt** *number*}—Matches the length of the specified fields in the HTTP response message header, or the overall number of fields (count) in the header. You can specify the field name explicitly or match the field name to a regular expression or regular expression class. Field names are listed in the previous bullet.
- **match** [**not**] **response header** {**length gt** *bytes* | **count gt** *number* | **non-ascii**}—Matches the overall length of the HTTP response message header, or the overall number of fields (count) in the header, or headers that have non-ASCII characters.
- **match** [**not**] **response status-line regex** {*regex_name* | **class** *class_name*}—Matches text found in the HTTP response message status line against the specified regular expression or regular expression class.

d) Enter **exit** to leave class map configuration mode.

Step 2 Create an HTTP inspection policy map: **policy-map type inspect http** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created an HTTP class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for HTTP class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop-connection** [**log**]—Drop the packet and close the connection.
- **reset** [**log**]—Drop the packet, close the connection, and send a TCP reset to the server or client.
- **log**—Send a system log message. You can use this option alone or with one of the other actions.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **body-match-maximum** *number*—Sets the maximum number of characters in the body of an HTTP message that should be searched in a body match. The default is 200 bytes. A large number will have a significant impact on performance.
- **protocol-violation action** {**drop-connection** [**log**] | **reset** [**log**] | **log**}—Checks for HTTP protocol violations. You must also choose the action to take for violations (drop connection, reset, or log) and whether to enable or disable logging.
- **spoof-server** *string*—Substitutes a string for the server header field. WebVPN streams are not subject to the spoof-server command.

Example

The following example shows how to define an HTTP inspection policy map that will allow and log any HTTP connection that attempts to access “www\.xyz.com/.*\.asp" or "www\.xyz[0-9][0-9]\.com" with methods "GET" or "PUT." All other URL/Method combinations will be silently allowed.

```
hostname(config)# regex url1 “www\.xyz.com/.*\.asp”
hostname(config)# regex url2 “www\.xyz[0-9][0-9]\.com”
hostname(config)# regex get “GET”
```

```
hostname(config)# regex put "PUT"

hostname(config)# class-map type regex match-any url_to_log
hostname(config-cmap)# match regex url1
hostname(config-cmap)# match regex url2
hostname(config-cmap)# exit

hostname(config)# class-map type regex match-any methods_to_log
hostname(config-cmap)# match regex get
hostname(config-cmap)# match regex put
hostname(config-cmap)# exit

hostname(config)# class-map type inspect http http_url_policy
hostname(config-cmap)# match request uri regex class url_to_log
hostname(config-cmap)# match request method regex class methods_to_log
hostname(config-cmap)# exit

hostname(config)# policy-map type inspect http http_policy
hostname(config-pmap)# class http_url_policy
hostname(config-pmap-c)# log
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

ICMP Inspection

The ICMP inspection engine allows ICMP traffic to have a "session" so it can be inspected like TCP and UDP traffic. Without the ICMP inspection engine, we recommend that you do not allow ICMP through the ASA in an ACL. Without stateful inspection, ICMP can be used to attack your network. The ICMP inspection engine ensures that there is only one response for each request, and that the sequence number is correct.

However, ICMP traffic directed to an ASA interface is never inspected, even if you enable ICMP inspection. Thus, a ping (echo request) to an interface can fail under specific circumstances, such as when the echo request comes from a source that the ASA can reach through a backup default route.



Note NAT uses ICMP inspection when translating packets even if you disable ICMP inspection.

For information on enabling ICMP inspection, see Configure Application Layer Protocol Inspection, on page 253.

ICMP Error Inspection

When ICMP Error inspection is enabled, the ASA creates translation sessions for intermediate hops that send ICMP error messages, based on the NAT configuration. The ASA overwrites the packet with the translated IP addresses.

When disabled, the ASA does not create translation sessions for intermediate nodes that generate ICMP error messages. ICMP error messages generated by the intermediate nodes between the inside host and the ASA

reach the outside host without consuming any additional NAT resource. This is undesirable when an outside host uses the traceroute command to trace the hops to the destination on the inside of the ASA. When the ASA does not translate the intermediate hops, all the intermediate hops appear with the mapped destination IP address.

Note You should always enable ICMP Error inspection if there is a possibility that NAT will be used on ICMP packets. Because NAT automatically uses ICMP inspection for ICMP packets, even if you have ICMP inspection disabled, the use of the mapped destination address as the source address can make it look like a scanner is examining your network. For example, without ICMP Error inspection also enabled, if the echo request packet has its destination translated, when it is embedded in a ICMP time exceeded response, the outer header of the time exceeded request uses the translated destination as the source address. If you enable ICMP Error inspection, the time exceeded source address will be set to the correct value.

For information on enabling ICMP Error inspection, see Configure Application Layer Protocol Inspection, on page 253.

ILS Inspection

The Internet Locator Service (ILS) inspection engine provides NAT support for Microsoft NetMeeting, SiteServer, and Active Directory products that use LDAP to exchange directory information with an ILS server. You cannot use PAT with ILS inspection because only IP addresses are stored by an LDAP database.

For search responses, when the LDAP server is located outside, consider using NAT to allow internal peers to communicate locally while registered to external LDAP servers. If you do not need to use NAT, we recommend that you turn off the inspection engine to provide better performance.

Additional configuration may be necessary when the ILS server is located inside the ASA border. This would require a hole for outside clients to access the LDAP server on the specified port, typically TCP 389.

Note Because ILS traffic (H225 call signaling) only occurs on the secondary UDP channel, the TCP connection is disconnected after the TCP inactivity interval. By default, this interval is 60 minutes and can be adjusted using the TCP **timeout** command. In ASDM, this is on the **Configuration > Firewall > Advanced > Global Timeouts** pane.

ILS inspection has the following limitations:

- Referral requests and responses are not supported.
- Users in multiple directories are not unified.
- Single users having multiple identities in multiple directories cannot be recognized by NAT.

For information on enabling ILS inspection, see Configure Application Layer Protocol Inspection, on page 253.

Instant Messaging Inspection

The Instant Messaging (IM) inspect engine lets you control the network usage of IM and stop leakage of confidential data, propagation of worms, and other threats to the corporate network.

IM inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. However, the default inspect class does include the default IM ports, so you can simply edit the default global inspection policy to add IM inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

If you decide to implement IM inspection, you can also configure an IM inspection policy map to specify actions when a message violates a parameter. The following procedure explains IM inspection policy maps.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create an IM inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect im** [**match-all** | **match-any**] *class_map_name*

Where *the class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one **match** statement. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **protocol {im-yahoo | im-msn}**—Matches a specific IM protocol, either Yahoo or MSN.

- **match** [**not**] **service {chat | file-transfer | webcam | voice-chat | conference | games}**—Matches the specific IM service.
- **match** [**not**] **login-name regex** {*regex_name* | **class** *class_name*}—Matches the source client login name of the IM message against the specified regular expression or regular expression class.
- **match** [**not**] **peer-login-name regex** {*regex_name* | **class** *class_name*}—Matches the destination peer login name of the IM message against the specified regular expression or regular expression class.
- **match** [**not**] **ip-address** *ip_address mask*}—Matches the source IP address and mask of the IM message.
- **match** [**not**] **peer-ip-address** *ip_address mask*}—Matches the destination IP address and mask of the IM message.
- **match** [**not**] **version regex** {*regex_name* | **class** *class_name*}—Matches the version of the IM message against the specified regular expression or regular expression class.
- **match** [**not**] **filename regex** {*regex_name* | **class** *class_name*}—Matches the filename of the IM message against the specified regular expression or regular expression class. This match is not supported for the MSN IM protocol.

d) Enter **exit** to leave class map configuration mode.

Step 2 Create an IM inspection policy map: **policy-map type inspect im** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created an IM class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for IM class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering the following command:

- **drop-connection** [**log**]—Drop the packet and close the connection.
- **reset** [**log**]—Drop the packet, close the connection, and send a TCP reset to the server or client.
- **log**—Send a system log message. You can use this option alone or with one of the other actions.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

Example

The following example shows how to define an IM inspection policy map.

```
hostname(config)# regex loginname1 "ying\@yahoo.com"
hostname(config)# regex loginname2 "Kevin\@yahoo.com"
hostname(config)# regex loginname3 "rahul\@yahoo.com"
hostname(config)# regex loginname4 "darshant\@yahoo.com"
hostname(config)# regex yahoo_version_regex "1\.0"
hostname(config)# regex gif_files ".*\.gif"
hostname(config)# regex exe_files ".*\.exe"

hostname(config)# class-map type regex match-any yahoo_src_login_name_regex
hostname(config-cmap)# match regex loginname1
hostname(config-cmap)# match regex loginname2

hostname(config)# class-map type regex match-any yahoo_dst_login_name_regex
hostname(config-cmap)# match regex loginname3
hostname(config-cmap)# match regex loginname4

hostname(config)# class-map type inspect im match-any yahoo_file_block_list
hostname(config-cmap)# match filename regex gif_files
hostname(config-cmap)# match filename regex exe_files

hostname(config)# class-map type inspect im match-all yahoo_im_policy
hostname(config-cmap)# match login-name regex class yahoo_src_login_name_regex
hostname(config-cmap)# match peer-login-name regex class yahoo_dst_login_name_regex

hostname(config)# class-map type inspect im match-all yahoo_im_policy2
hostname(config-cmap)# match version regex yahoo_version_regex

hostname(config)# class-map im_inspect_class_map
hostname(config-cmap)# match default-inspection-traffic

hostname(config)# policy-map type inspect im im_policy_all
hostname(config-pmap)# class yahoo_file_block_list
hostname(config-pmap-c)# match service file-transfer
hostname(config-pmap)# class yahoo_im_policy
hostname(config-pmap-c)# drop-connection
hostname(config-pmap)# class yahoo_im_policy2
hostname(config-pmap-c)# reset
hostname(config)# policy-map global_policy_name
hostname(config-pmap)# class im_inspect_class_map
hostname(config-pmap-c)# inspect im im_policy_all
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

IP Options Inspection

You can configure IP Options inspection to control which IP packets are allowed based on the contents of the IP Options field in the packet header. You can drop packets that have unwanted options, clear the options (and allow the packet), or allow the packet without change.

IP options provide control functions that are required in some situations but unnecessary for most common communications. In particular, IP options include provisions for time stamps, security, and special routing. Use of IP Options is optional, and the field can contain zero, one, or more options.

For a list of IP options, with references to the relevant RFCs, see the IANA page, http://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml.

IP options inspection is enabled by default, but for RSVP traffic only. You need to configure it only if you want to allow additional options than the default map allows, or if you want to apply it to other types of traffic by using a non-default inspection traffic class map.

Note IP options inspection does not work on fragmented packets. For example, options are not cleared from fragments.

The following sections describe IP Options inspection.

Defaults for IP Options Inspection

IP Options inspection is enabled by default for RSVP traffic only, using the _default_ip_options_map inspection policy map.

- The Router Alert option is allowed.

  This option notifies transit routers to inspect the contents of the packet even when the packet is not destined for that router. This inspection is valuable when implementing RSVP and similar protocols that require relatively complex processing from the routers along the packet's delivery path. Dropping RSVP packets containing the Router Alert option can cause problems in VoIP implementations.

- Packets that contain any other options are dropped.

  Each time a packet is dropped due to inspection, syslog 106012 is issued. The message shows which option caused the drop. Use the **show service-policy inspect ip-options** command to view statistics for each option.

Following is the policy map configuration:

```
policy-map type inspect ip-options _default_ip_options_map
 description Default IP-OPTIONS policy-map
 parameters
  router-alert action allow
```

Configure an IP Options Inspection Policy Map

If you want to perform non-default IP options inspection, create an IP options inspection policy map to specify how you want to handle each option type.

Procedure

Step 1 Create an IP options inspection policy map: **policy-map type inspect ip-options** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

Step 4 Identify the options you want to allow.

You can inspect the following options. In all cases, the **allow** action allows packets that contain the option without modification; the **clear** action allows the packets but removes the option from the header.

Use the **no** form of the command to remove the option from the map. Any packet that contains an option that you do not include in the map is dropped, even if the packet contains otherwise allowed or cleared options.

For a list of IP options, with references to the relevant RFCs, see the IANA page, http://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml.

- **default action** {**allow** | **clear**}—Sets the default action for any option not explicitly included in the map. If you do not set a default action of allow or clear, packets that contain non-allowed options are dropped
- **basic-security action** {**allow** | **clear**}—Allows or clears the Security (SEC) option.
- **commercial-security action** {**allow** | **clear**}—Allows or clears the Commercial Security (CIPSO) option.
- **eool action** {**allow** | **clear**}—Allows or clears the End of Options List option.
- **exp-flow-control action** {**allow** | **clear**}—Allows or clears the Experimental Flow Control (FINN) option.
- **exp-measurement action** {**allow** | **clear**}—Allows or clears the Experimental Measurement (ZSU) option.
- **extended-security action** {**allow** | **clear**}—Allows or clears the Extended Security (E-SEC) option.
- **imi-traffic-descriptor action** {**allow** | **clear**}—Allows or clears the IMI Traffic Descriptor (IMITD) option.
- **nop action** {**allow** | **clear**}—Allows or clears the No Operation option.
- **quick-start action** {**allow** | **clear**}—Allows or clears the Quick-Start (QS) option.
- **record-route action** {**allow** | **clear**}—Allows or clears the Record Route (RR) option.
- **router-alert action** {**allow** | **clear**}—Allows or clears the Router Alert (RTRALT) option.
- **timestamp action** {**allow** | **clear**}—Allows or clears the Time Stamp (TS) option.
- {**0-255**} **action** {**allow** | **clear**}—Allows or clears the option identified by the option type number. The number is the whole option type octet (copy, class, and option number), not just the option number portion of the octet. These option types might not represent real options. Non-standard options must be in the expected type-length-value format defined in the Internet Protocol RFC 791, http://tools.ietf.org/html/rfc791.

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

IPsec Pass Through Inspection

IPsec Pass Through inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. However, the default inspect class does include the default IPsec ports, so you can simply edit the default global inspection policy to add IPsec inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

The following sections describe the IPsec Pass Through inspection engine.

IPsec Pass Through Inspection Overview

Internet Protocol Security (IPsec) is a protocol suite for securing IP communications by authenticating and encrypting each IP packet of a data stream. IPsec also includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used to protect data flows between a pair of hosts (for example, computer users or servers), between a pair of security gateways (such as routers or firewalls), or between a security gateway and a host.

IPsec Pass Through application inspection provides convenient traversal of ESP (IP protocol 50) and AH (IP protocol 51) traffic associated with an IKE UDP port 500 connection. It avoids lengthy ACL configuration to permit ESP and AH traffic and also provides security using timeout and max connections.

Configure a policy map for IPsec Pass Through to specify the restrictions for ESP or AH traffic. You can set the per client max connections and the idle timeout.

NAT and non-NAT traffic is permitted. However, PAT is not supported.

Configure an IPsec Pass Through Inspection Policy Map

An IPsec Pass Through map lets you change the default configuration values used for IPsec Pass Through application inspection. You can use an IPsec Pass Through map to permit certain flows without using an ACL.

The configuration includes a default map, _default_ipsec_passthru_map, that sets no maximum limit on ESP connections per client, and sets the ESP idle timeout at 10 minutes. You need to configure an inspection policy map only if you want different values, or if you want to set AH values.

Procedure

Step 1 Create an IPsec Pass Through inspection policy map: **policy-map type inspect ipsec-pass-thru** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **esp per-client-max** *number* **timeout** *time*—Allows ESP tunnels and sets the maximum connections allowed per client and the idle timeout (in hh:mm:ss format). To allow an unlimited number of connections, specify 0 for the number.
- **ah per-client-max** *number* **timeout** *time*—Allows AH tunnels. The parameters have the same meaning as for the esp command.

Example

The following example shows how to use ACLs to identify IKE traffic, define an IPsec Pass Thru parameter map, define a policy, and apply the policy to the outside interface:

```
hostname(config)# access-list ipsecpassthruacl permit udp any any eq 500
hostname(config)# class-map ipsecpassthru-traffic
hostname(config-cmap)# match access-list ipsecpassthruacl
hostname(config)# policy-map type inspect ipsec-pass-thru iptmap
hostname(config-pmap)# parameters
hostname(config-pmap-p)# esp per-client-max 10 timeout 0:11:00
hostname(config-pmap-p)# ah per-client-max 5 timeout 0:06:00
hostname(config)# policy-map inspection_policy
hostname(config-pmap)# class ipsecpassthru-traffic
hostname(config-pmap-c)# inspect ipsec-pass-thru iptmap
hostname(config)# service-policy inspection_policy interface outside
```

IPv6 Inspection

IPv6 inspection lets you selectively log or drop IPv6 traffic based on the extension header. In addition, IPv6 inspection can check conformance to RFC 2460 for type and order of extension headers in IPv6 packets.

IPv6 inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. You can simply edit the default global inspection policy to add IPv6 inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Defaults for IPv6 Inspection

If you enable IPv6 inspection and do not specify an inspection policy map, then the default IPv6 inspection policy map is used, and the following actions are taken:

- Allows only known IPv6 extension headers. Non-conforming packets are dropped and logged.
- Enforces the order of IPv6 extension headers as defined in the RFC 2460 specification. Non-conforming packets are dropped and logged.
- Drops any packet with a routing type header.

Following is the policy map configuration:

```
policy-map type inspect ipv6 _default_ipv6_map
 description Default IPV6 policy-map
 parameters
  verify-header type
  verify-header order
 match header routing-type range 0 255
  drop log
```

Configure an IPv6 Inspection Policy Map

To identify extension headers to drop or log, or to disable packet verification, create an IPv6 inspection policy map to be used by the service policy.

Procedure

Step 1 Create an IPv6 inspection policy map: **policy-map type inspect ipv6** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 (Optional) Drop or log traffic based on the headers in IPv6 messages.

a) Identify the traffic based on the IPv6 header: **match header** *type*

Where *type* is one of the following:

- **ah**—Matches the IPv6 Authentication extension header.
- **count gt** *number*—Specifies the maximum number of IPv6 extension headers, from 0 to 255.
- **destination-option**—Matches the IPv6 destination-option extension header.
- **esp**—Matches the IPv6 Encapsulation Security Payload (ESP) extension header.
- **fragment**—Matches the IPv6 fragment extension header.
- **hop-by-hop**—Matches the IPv6 hop-by-hop extension header.
- **routing-address count gt** *number*—Sets the maximum number of IPv6 routing header type 0 addresses, greater than a number between 0 and 255.
- **routing-type** {**eq** | **range**} *number*—Matches the IPv6 routing header type, from 0 to 255. For a range, separate values by a space, for example, **30 40**.

b) Specify the action to perform on matching packets. You can drop the packet and optionally log it, or just log it. If you do not enter an action, the packet is logged.

- **drop** [**log**]—Drop all packets that match.
- **log**—Send a system log message. You can use this option alone or with one of the other actions.

c) Repeat the process until you identify all headers that you want to drop or log.

Step 4 Configure parameters that affect the inspection engine.

a) Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **verify-header type**—Allows only known IPv6 extension headers.
- **verify-header order**—Enforces the order of IPv6 extension headers as defined in RFC 2460.

Example

The following example creates an inspection policy map that will drop and log all IPv6 packets with the hop-by-hop, destination-option, routing-address, and routing type 0 headers. It also enforces header order and type.

```
policy-map type inspect ipv6 ipv6-pm
 parameters
  verify-header type
  verify-header order
 match header hop-by-hop
  drop log
 match header destination-option
  drop log
 match header routing-address count gt 0
  drop log
 match header routing-type eq 0
  drop log

policy-map global_policy
 class class-default
  inspect ipv6 ipv6-pm
!
service-policy global_policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

NetBIOS Inspection

NetBIOS application inspection performs NAT for the embedded IP address in the NetBIOS name service (NBNS) packets and NetBIOS datagram services packets. It also enforces protocol conformance, checking the various count and length fields for consistency.

NetBIOS inspection is enabled by default. You can optionally create a policy map to drop or log NetBIOS protocol violations. The following procedure explains how to configure a NetBIOS inspection policy map.

Procedure

Step 1 Create a NetBIOS inspection policy map: **policy-map type inspect netbios** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

Step 4 Specify the action to take for NETBIOS protocol violations: **protocol-violation action** {**drop** [**log**] | **log**}

Where the **drop** action drops the packet. The **log** action sends a system log message when this policy map matches traffic.

Example

```
hostname(config)# policy-map type inspect netbios netbios_map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# protocol-violation drop log

hostname(config)# policy-map netbios_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# no inspect netbios
hostname(config-pmap-c)# inspect netbios netbios_map
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

PPTP Inspection

PPTP is a protocol for tunneling PPP traffic. A PPTP session is composed of one TCP channel and usually two PPTP GRE tunnels. The TCP channel is the control channel used for negotiating and managing the PPTP GRE tunnels. The GRE tunnels carry PPP sessions between the two hosts.

When enabled, PPTP application inspection inspects PPTP protocol packets and dynamically creates the GRE connections and xlates necessary to permit PPTP traffic.

Specifically, the ASA inspects the PPTP version announcements and the outgoing call request/response sequence. Only PPTP Version 1, as defined in RFC 2637, is inspected. Further inspection on the TCP control channel is disabled if the version announced by either side is not Version 1. In addition, the outgoing-call request and reply sequence are tracked. Connections and xlates are dynamically allocated as necessary to permit subsequent secondary GRE data traffic.

The PPTP inspection engine must be enabled for PPTP traffic to be translated by PAT. Additionally, PAT is only performed for a modified version of GRE (RFC2637) and only if it is negotiated over the PPTP TCP control channel. PAT is not performed for the unmodified version of GRE (RFC 1701 and RFC 1702).

For information on enabling PPTP inspection, see Configure Application Layer Protocol Inspection, on page 253.

RSH Inspection

RSH inspection is enabled by default. The RSH protocol uses a TCP connection from the RSH client to the RSH server on TCP port 514. The client and server negotiate the TCP port number where the client listens for the STDERR output stream. RSH inspection supports NAT of the negotiated port number if necessary.

For information on enabling RSH inspection, see Configure Application Layer Protocol Inspection, on page 253.

SMTP and Extended SMTP Inspection

ESMTP inspection detects attacks, including spam, phising, malformed message attacks, and buffer overflow/underflow attacks. It also provides support for application security and protocol conformance, which enforces the sanity of the ESMTP messages as well as block senders/receivers, and block mail relay.

ESMTP inspection is enabled by default. You need to configure it only if you want different processing than that provided by the default inspection map.

The following sections describe the ESMTP inspection engine.

SMTP and ESMTP Inspection Overview

Extended SMTP (ESMTP) application inspection provides improved protection against SMTP-based attacks by restricting the types of SMTP commands that can pass through the ASA and by adding monitoring capabilities. ESMTP is an enhancement to the SMTP protocol and is similar is most respects to SMTP.

ESMTP application inspection controls and reduces the commands that the user can use as well as the messages that the server returns. ESMTP inspection performs three primary tasks:

- Restricts SMTP requests to seven basic SMTP commands and eight extended commands. Supported commands are the following:
  - Extended SMTP—AUTH, EHLO, ETRN, HELP, SAML, SEND, SOML, STARTTLS, and VRFY.
  - SMTP (RFC 821)—DATA, HELO, MAIL, NOOP, QUIT, RCPT, RSET.
- Monitors the SMTP command-response sequence.
- Generates an audit trail—Audit record 108002 is generated when an invalid character embedded in the mail address is replaced. For more information, see RFC 821.

ESMTP inspection monitors the command and response sequence for the following anomalous signatures:

- Truncated commands.
- Incorrect command termination (not terminated with <CR><LR>).

- The MAIL and RCPT commands specify who are the sender and the receiver of the mail. Mail addresses are scanned for strange characters. The pipeline character (|) is deleted (changed to a blank space) and “<” ,”>” are only allowed if they are used to define a mail address (“>” must be preceded by “<”).
- Unexpected transition by the SMTP server.
- For unknown or unsupported commands, the inspection engine changes all the characters in the packet to X, which are rejected by the internal server. This results in a message such as “500 Command unknown: 'XXX'.” Incomplete commands are discarded

  Unsupported ESMTP commands are ATRN, ONEX, VERB, CHUNKING, and private extensions..
- TCP stream editing.
- Command pipelining.



Note With ESMTP inspection enabled, a Telnet session used for interactive SMTP may hang if the following rules are not observed: SMTP commands must be at least four characters in length; they must be terminated with carriage return and line feed; and you must wait for a response before issuing the next reply.

Defaults for ESMTP Inspection

ESMTP inspection is enabled by default, using the _default_esmtp_map inspection policy map.

- The server banner is masked. The ESMTP inspection engine changes the characters in the server SMTP banner to asterisks except for the “2”, “0”, “0” characters. Carriage return (CR) and linefeed (LF) characters are ignored.
- Encrypted connections are allowed but not inspected.
- Special characters in sender and receiver address are not noticed, no action is taken.
- Connections with command line length greater than 512 are dropped and logged.
- Connections with more than 100 recipients are dropped and logged.
- Messages with body length greater than 998 bytes are logged.
- Connections with header line length greater than 998 are dropped and logged.
- Messages with MIME filenames greater than 255 characters are dropped and logged.
- EHLO reply parameters matching “others” are masked.

Following is the policy map configuration:

```
policy-map type inspect esmtp _default_esmtp_map
 description Default ESMTP policy-map
 parameters
  mask-banner
  no mail-relay
  no special-character
  allow-tls
 match cmd line length gt 512
  drop-connection log
```

```
match cmd RCPT count gt 100
 drop-connection log
match body line length gt 998
 log
match header line length gt 998
 drop-connection log
match sender-address length gt 320
 drop-connection log
match MIME filename length gt 255
 drop-connection log
match ehlo-reply-parameter others
 mask
```

Configure an ESMTP Inspection Policy Map

To specify actions when a message violates a parameter, create an ESMTP inspection policy map. You can then apply the inspection policy map when you enable ESMTP inspection.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 Create an ESMTP inspection policy map: **policy-map type inspect esmtp** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **body** {**length** | **line length**} **gt** *bytes*—Matches messages where the length or length of a line in an ESMTP body message is greater than the specified number of bytes.
- **match** [**not**] **cmd verb** *verb1* [*verb2*...]—Matches the command verb in the message. You can specify one or more of the following commands: auth, data, ehlo, etrn, helo, help, mail, noop, quit, rcpt, rset, saml, soml, vrfy.
- **match** [**not**] **cmd line length gt** *bytes*—Matches messages where the length of a line in the command verb is greater than the specified number of bytes.
- **match** [**not**] **cmd rcpt count gt** *count*—Matches messages where the number of recipients is greater than the specified count.
- **match** [**not**] **ehlo-reply-parameter** *parameter* [*parameter2*...]—Matches ESMTP EHLO reply parameters. You can specify one or more of the following parameters: 8bitmime, auth, binaryname, checkpoint, dsn, etrn, others, pipelining, size, vrfy.

- **match** [**not**] **header** {**length** | **line length**} **gt** *bytes*—Matches messages where the length or length of a line in an ESMTP header is greater than the specified number of bytes.
- **match** [**not**] **header to-fields count gt** *count*—Matches messages where the number of To fields in the header is greater than the specified number.
- **match** [**not**] **invalid-recipients count gt** *number*—Matches messages where the number of invalid recipients is greater than the specified count.
- **match** [**not**] **mime filetype regex** {*regex_name* | **class** *class_name*}—Matches the MIME or media file type against the specified regular expression or regular expression class.
- **match** [**not**] **mime filename length gt** *bytes*—Matches messages where a file name is longer than the specified number of bytes.
- **match** [**not**] **mime encoding** *type* [*type2...*]—Matches the MIME encoding type. You can specify one or more of the following types: 7bit, 8bit, base64, binary, others, quoted-printable.
- **match** [**not**] **sender-address regex** {*regex_name* | **class** *class_name*}—Matches the sender email address against the specified regular expression or regular expression class.
- **match** [**not**] **sender-address length gt** *bytes*—Matches messages where the sender address is greater than the specified number of bytes.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop-connection** [**log**]—Drop the packet and close the connection.
- **mask** [**log**]—Mask out the matching portion of the packet. This action is available for **ehlo-reply-parameter** and **cmd verb** only.
- **reset** [**log**]—Drop the packet, close the connection, and send a TCP reset to the server or client.
- **log**—Send a system log message. You can use this option alone or with one of the other actions.
- **rate-limit** *message_rate*—Limit the rate of messages in packets per second. This option is available with **cmd verb** only, where you can use it as the only action, or you can use it in conjunction with the **mask** action.

You can specify multiple **match** commands in the policy map. For information about the order of **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

Step 4 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **mail-relay** *domain-name* **action** {**drop-connection** [**log**] | **log**}—Identifies a domain name for mail relay. You can either drop the connection and optionally log it, or log it.
- **mask-banner**—Masks the banner from the ESMTP server.

- **special-character action** {**drop-connection** [**log**] | **log**}—Identifies the action to take for messages that include the special characters pipe (|), back quote, and NUL in the sender or receiver email addresses. You can either drop the connection and optionally log it, or log it.
- **allow-tls** [**action log**]—Whether to allow ESMTP over TLS (encrypted connections) without inspection. You can optionally log encrypted connections. The default is to allow TLS sessions without inspection. If you specify **no allow-tls**, the system strips the STARTTLS indication from the session connection and forces a plain-text connection.

Example

The following example shows how to define an ESMTP inspection policy map.

```
hostname(config)# regex user1 "user1@cisco.com"
hostname(config)# regex user2 "user2@cisco.com"
hostname(config)# regex user3 "user3@cisco.com"
hostname(config)# class-map type regex senders_black_list
hostname(config-cmap)# description "Regular expressions to filter out undesired senders"
hostname(config-cmap)# match regex user1
hostname(config-cmap)# match regex user2
hostname(config-cmap)# match regex user3

hostname(config)# policy-map type inspect esmtp advanced_esmtp_map
hostname(config-pmap)# match sender-address regex class senders_black_list
hostname(config-pmap-c)# drop-connection log

hostname(config)# policy-map outside_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect esmtp advanced_esmtp_map

hostname(config)# service-policy outside_policy interface outside
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

SNMP Inspection

SNMP application inspection is applied to both to-the-device and through-the-device traffic. This inspection is necessary if you configure SNMP v3 where users are limited to specific SNMP hosts. Without the inspection, a defined v3 user can poll the device from any allowed host. SNMP inspection is enabled by default for the default ports, so you need to configure it only if you use non-default ports. The default ports are UDP/161, 162 (for all device types) and UDP/4161 for devices that also run FXOS, as FXOS listens on UDP/161.

By default, the SNMP inspection limits the polling to the configured version.

Note If you configure SNMP on a device that also runs FXOS, SNMP inspection is mandatory, and is re-enabled if you disable it. SNMP inspection is enabled on a traffic class map that includes port UDP/4161.

Optionally, you can further restrict SNMP traffic to a specific version of SNMP. Earlier versions of SNMP are less secure; therefore, denying certain SNMP versions may be required by your security policy. The system can deny SNMP versions 1, 2, 2c, or 3. You control the versions permitted by creating an SNMP map, as explained below. If you do not need to control the versions, simply enable SNMP inspection without a map.

Procedure

Create an SNMP map.

Use the **snmp-map** *map_name* command to create the map and enter SNMP map configuration mode, then the **deny version** *version* command to identify the versions to disallow. The version can be 1, 2, 2c, or 3.

Example:

The following example denies SNMP Versions 1 and 2:

```
hostname(config)# snmp-map sample_map
hostname(config-snmp-map)# deny version 1
hostname(config-snmp-map)# deny version 2
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

SQL*Net Inspection

SQL*Net inspection is enabled by default. The inspection engine supports SQL*Net versions 1 and 2, but only the Transparent Network Substrate (TNS) format. Inspection does not support the Tabular Data Stream (TDS) format. SQL*Net messages are scanned for embedded addresses and ports, and NAT rewrite is applied when necessary.

The default port assignment for SQL*Net is 1521. This is the value used by Oracle for SQL*Net, but this value does not agree with IANA port assignments for Structured Query Language (SQL). If your application uses a different port, apply the SQL*Net inspection to a traffic class that includes that port.

Disable SQL*Net inspection when:

- SQL data transfer occurs on the same port as the SQL control TCP port 1521. The security appliance acts as a proxy when SQL*Net inspection is enabled and reduces the client window size from 65000 to about 16000 causing data transfer issues.
- The inspection of high rates of SQL traffic causes unacceptable spikes in CPU usage.

After disabling SQL*Net inspection, use the **clear conn port 1521** command so that connections can be rebuilt without inspection.

For information on enabling SQL*Net inspection, see Configure Application Layer Protocol Inspection, on page 253.

Sun RPC Inspection

This section describes Sun RPC application inspection.

Sun RPC Inspection Overview

Sun RPC protocol inspection is enabled by default. You simply need to manage the Sun RPC server table to identify which services are allowed to traverse the firewall. However, pinholing for NFS is done for any server even without the server table configuration.

Sun RPC is used by NFS and NIS. Sun RPC services can run on any port. When a client attempts to access a Sun RPC service on a server, it must learn the port that service is running on. It does this by querying the port mapper process, usually rpcbind, on the well-known port of 111.

The client sends the Sun RPC program number of the service and the port mapper process responds with the port number of the service. The client sends its Sun RPC queries to the server, specifying the port identified by the port mapper process. When the server replies, the ASA intercepts this packet and opens both embryonic TCP and UDP connections on that port.

NAT or PAT of Sun RPC payload information is not supported.

Manage Sun RPC Services

Use the Sun RPC services table to control Sun RPC traffic based on established Sun RPC sessions.

Procedure

Step 1 Configure the Sun RPC service properties.

sunrpc-server *interface_name ip_address mask* **service** *service_type* **protocol** {**tcp** | **udp**} *port*[*-port*] **timeout** *hh:mm:ss*

Where:

- *interface_name*—The interface through which traffic to the server flows.
- *ip_address mask*—The address of the Sun RPC server.
- **service** *service_type* —The service type on the server, which is the mapping between a specific service type and the port number used for the service. To determine the service type (for example, 100003), use the **sunrpcinfo** command at the UNIX or Linux command line on the Sun RPC server machine.
- **protocol** {**tcp** | **udp**}—Whether the service uses TCP or UDP.
- *port*[*-port*]—The port or range of ports used by the service. To specify a range of ports, separate the starting and ending port numbers in the range with a hyphen (for example, 111-113).

- **timeout** *hh:mm:ss*—The idle timeout for the pinhole opened for the connection by Sun RPC inspection.

Example:

For example, to create a timeout of 30 minutes to the Sun RPC server with the IP address 192.168.100.2, enter the following command. In this example, the Sun RPC server is on the inside interface using TCP port 111.

```
hostname(config)# sunrpc-server inside 192.168.100.2 255.255.255.255
service 100003 protocol tcp 111 timeout 00:30:00
```

Step 2 (Optional.) Monitor the pinholes created for these services.

To display the pinholes open for Sun RPC services, enter the **show sunrpc-server active** command. For example:

```
hostname# show sunrpc-server active
LOCAL FOREIGN SERVICE TIMEOUT
-----------------------------------------------
1 209.165.200.5/0 192.168.100.2/2049 100003 0:30:00
2 209.165.200.5/0 192.168.100.2/2049 100003 0:30:00
3 209.165.200.5/0 192.168.100.2/647 100005 0:30:00
4 209.165.200.5/0 192.168.100.2/650 100005 0:30:00
```

The entry in the LOCAL column shows the IP address of the client or server on the inside interface, while the value in the FOREIGN column shows the IP address of the client or server on the outside interface.

If necessary, you can clear these services using the **clear sunrpc-server active**

TFTP Inspection

TFTP inspection is enabled by default.

TFTP, described in RFC 1350, is a simple protocol to read and write files between a TFTP server and client.

The inspection engine inspects TFTP read request (RRQ), write request (WRQ), and error notification (ERROR), and dynamically creates connections and translations, if necessary, to permit file transfer between a TFTP client and server.

A dynamic secondary channel and a PAT translation, if necessary, are allocated on a reception of a valid read (RRQ) or write (WRQ) request. This secondary channel is subsequently used by TFTP for file transfer or error notification.

Only the TFTP server can initiate traffic over the secondary channel, and at most one incomplete secondary channel can exist between the TFTP client and server. An error notification from the server closes the secondary channel.

TFTP inspection must be enabled if static PAT is used to redirect TFTP traffic.

For information on enabling TFTP inspection, see Configure Application Layer Protocol Inspection, on page 253.

XDMCP Inspection

XDMCP is a protocol that uses UDP port 177 to negotiate X sessions, which use TCP when established.

For successful negotiation and start of an XWindows session, the ASA must allow the TCP back connection from the Xhosted computer. To permit the back connection, you can use access rules to allow the TCP ports. Alternatively, you can use the **established** command on the ASA. Once XDMCP negotiates the port to send the display, the **established** command is consulted to verify if this back connection should be permitted.

During the XWindows session, the manager talks to the display Xserver on the well-known port 6000 | n. Each display has a separate connection to the Xserver, as a result of the following terminal setting.

```
setenv DISPLAY Xserver:n
```

where *n* is the display number.

When XDMCP is used, the display is negotiated using IP addresses, which the ASA can NAT if needed. XDCMP inspection does not support PAT.

For information on enabling XDMCP inspection, see Configure Application Layer Protocol Inspection, on page 253.

VXLAN Inspection

Virtual Extensible Local Area Network (VXLAN) inspection works on VXLAN encapsulated traffic that passes through the ASA. It ensures that the VXLAN header format conforms to standards, dropping any malformed packets. VXLAN inspection is not done on traffic for which the ASA acts as a VXLAN Tunnel End Point (VTEP) or a VXLAN gateway, as those checks are done as a normal part of decapsulating VXLAN packets.

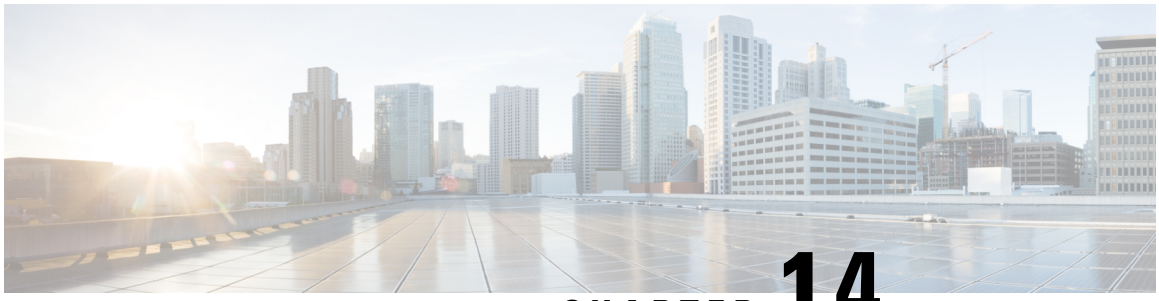
VXLAN packets are UDP, normally on port 4789. This port is part of the default-inspection-traffic class, so you can simply add VXLAN inspection to the inspection_default service policy rule. Alternatively, you can create a class for it using port or ACL matching.

History for Basic Internet Protocol Inspection

Feature Name	Releases	Feature Information
DCERPC inspection support for ISystemMapper UUID message RemoteGetClassObject opnum3.	9.4(1)	The ASA started supporting non-EPM DCERPC messages in release 8.3, supporting the ISystemMapper UUID message RemoteCreateInstance opnum4. This change extends support to the RemoteGetClassObject opnum3 message. We did not modify any commands.
VXLAN packet inspection	9.4(1)	The ASA can inspect the VXLAN header to enforce compliance with the standard format. We introduced the following command: **inspect vxlan**.

Feature Name	Releases	Feature Information
ESMTP inspection change in default behavior for TLS sessions.	9.4(1)	The default for ESMTP inspection was changed to allow TLS sessions, which are not inspected. However, this default applies to new or reimaged systems. If you upgrade a system that includes **no allow-tls**, the command is not changed. The change in default behavior was also made in these older versions: 8.4(7.25), 8.5(1.23), 8.6(1.16), 8.7(1.15), 9.0(4.28), 9.1(6.1), 9.2(3.2) 9.3(1.2), 9.3(2.2).
IP Options inspection improvements.	9.5(1)	IP Options inspection now supports all possible IP options. You can tune the inspection to allow, clear, or drop any standard or experimental options, including those not yet defined. You can also set a default behavior for options not explicitly defined in an IP options inspection map. We added the following commands: **basic-security**, **commercial-security**, **default**, **exp-flow-control**, **exp-measure**, **extended-security**, **imi-traffic-description**, **quick-start**, **record-route**, **timestamp**, and {**0-255**} (which indicates an IP option type number).
DCERPC inspection improvements and UUID filtering	9.5(2)	DCERPC inspection now supports NAT for OxidResolver ServerAlive2 opnum5 messages. You can also now filter on DCERPC message universally unique identifiers (UUIDs) to reset or log particular message types. There is a new DCERPC inspection class map for UUID filtering. We introduced the following command: **match** [**not**] **uuid**. We modified the following command: **class-map type inspect**.
DNS over TCP inspection.	9.6(2)	You can now inspect DNS over TCP traffic (TCP/53). We added the following command: **tcp-inspection**.
Cisco Umbrella support.	9.10(1)	You can configure the device to redirect DNS requests to Cisco Umbrella, so that your Enterprise Security policy defined in Cisco Umbrella can be applied to user connections. You can allow or block connections based on FQDN, or for suspicious FQDNs, you can redirect the user to the Cisco Umbrella intelligent proxy, which can perform URL filtering. The Umbrella configuration is part of the DNS inspection policy. We added or modified the following commands: **umbrella** (global and policy-map parameters configuration modes), **token**, **public-key**, **timeout edns**, **dnscrypt**, **show service-policy inspect dns detail**.

Feature Name	Releases	Feature Information
Cisco Umbrella Enhancements.	9.12(1)	You can now identify local domain names that should bypass Cisco Umbrella. DNS requests for these domains go directly to the DNS servers without Umbrella processing. You can also identify which Umbrella servers to use for resolving DNS requests. Finally, you can define the Umbrella inspection policy to fail open, so that DNS requests are not blocked if the Umbrella server is unavailable. We added or changed the following commands: **local-domain-bypass**, **resolver**, **umbrella fail-open**.
XDMCP inspection disabled by default in new installations.	9.15(1)	Previously, XDMCP inspection was enabled by default for all traffic. Now, on new installations, which includes new systems and reimaged systems, XDMCP is off by default. If you need this inspection, please enable it. Note that on upgrades, your current settings for XDMCP inspection are retained, even if you simply had it enabled by way of the default inspection settings.



CHAPTER 14

Inspection for Voice and Video Protocols

The following topics explain application inspection for voice and video protocols. For basic information on why you need to use inspection for certain protocols, and the overall methods for applying inspection, see Getting Started with Application Layer Protocol Inspection, on page 245.

- CTIQBE Inspection, on page 305
- H.323 Inspection, on page 306
- MGCP Inspection, on page 311
- RTSP Inspection, on page 314
- SIP Inspection, on page 317
- Skinny (SCCP) Inspection, on page 323
- STUN Inspection, on page 326
- History for Voice and Video Protocol Inspection, on page 326

CTIQBE Inspection

CTIQBE protocol inspection supports NAT, PAT, and bidirectional NAT. This enables Cisco IP SoftPhone and other Cisco TAPI/JTAPI applications to work successfully with Cisco CallManager for call setup across the ASA.

TAPI and JTAPI are used by many Cisco VoIP applications. CTIQBE is used by Cisco TSP to communicate with Cisco CallManager.

For information on enabling CTIQBE inspection, see Configure Application Layer Protocol Inspection, on page 253.

Limitations for CTIQBE Inspection

Stateful failover of CTIQBE calls is not supported.

The following summarizes special considerations when using CTIQBE application inspection in specific scenarios:

- If two Cisco IP SoftPhones are registered with different Cisco CallManagers, which are connected to different interfaces of the ASA, calls between these two phones fail.
- When Cisco CallManager is located on the higher security interface compared to Cisco IP SoftPhones, if NAT or outside NAT is required for the Cisco CallManager IP address, the mapping must be static as

Cisco IP SoftPhone requires the Cisco CallManager IP address to be specified explicitly in its Cisco TSP configuration on the PC.

- When using PAT or Outside PAT, if the Cisco CallManager IP address is to be translated, its TCP port 2748 must be statically mapped to the same port of the PAT (interface) address for Cisco IP SoftPhone registrations to succeed. The CTIQBE listening port (TCP 2748) is fixed and is not user-configurable on Cisco CallManager, Cisco IP SoftPhone, or Cisco TSP.

H.323 Inspection

H.323 inspection supports RAS, H.225, and H.245, and its functionality translates all embedded IP addresses and ports. It performs state tracking and filtering and can do a cascade of inspect function activation. H.323 inspection supports phone number filtering, dynamic T.120 control, H.245 tunneling control, HSI groups, protocol state tracking, H.323 call duration enforcement, T.38 Fax, and audio/video control.

H.323 inspection is enabled by default. You need to configure it only if you want non-default processing.

The following sections describe the H.323 application inspection.

H.323 Inspection Overview

H.323 inspection provides support for H.323 compliant applications such as Cisco CallManager. H.323 is a suite of protocols defined by the International Telecommunication Union for multimedia conferences over LANs. The ASA supports H.323 through Version 6, including H.323 v3 feature Multiple Calls on One Call Signaling Channel.

With H.323 inspection enabled, the ASA supports multiple calls on the same call signaling channel, a feature introduced with H.323 Version 3. This feature reduces call setup time and reduces the use of ports on the ASA.

The two major functions of H.323 inspection are as follows:

- NAT the necessary embedded IPv4 addresses in the H.225 and H.245 messages. Because H.323 messages are encoded in PER encoding format, the ASA uses an ASN.1 decoder to decode the H.323 messages.
- Dynamically allocate the negotiated H.245 and RTP/RTCP connections. The H.225 connection can also be dynamically allocated when using RAS.

How H.323 Works

The H.323 collection of protocols collectively may use up to two TCP connection and four to eight UDP connections. FastConnect uses only one TCP connection, and RAS uses a single UDP connection for registration, admissions, and status.

An H.323 client can initially establish a TCP connection to an H.323 server using TCP port 1720 to request Q.931 call setup. As part of the call setup process, the H.323 terminal supplies a port number to the client to use for an H.245 TCP connection. In environments where H.323 gatekeeper is in use, the initial packet is transmitted using UDP.

H.323 inspection monitors the Q.931 TCP connection to determine the H.245 port number. If the H.323 terminals are not using FastConnect, the ASA dynamically allocates the H.245 connection based on the inspection of the H.225 messages. The H.225 connection can also be dynamically allocated when using RAS.

Within each H.245 message, the H.323 endpoints exchange port numbers that are used for subsequent UDP data streams. H.323 inspection inspects the H.245 messages to identify these ports and dynamically creates connections for the media exchange. RTP uses the negotiated port number, while RTCP uses the next higher port number.

The H.323 control channel handles H.225 and H.245 and H.323 RAS. H.323 inspection uses the following ports.

- 1718—Gate Keeper Discovery UDP port
- 1719—RAS UDP port
- 1720—TCP Control Port

You must permit traffic for the well-known H.323 port 1719 for RAS signaling. Additionally, you must permit traffic for the well-known H.323 port 1720 for the H.225 call signaling; however, the H.245 signaling ports are negotiated between the endpoints in the H.225 signaling. When an H.323 gatekeeper is used, the ASA opens an H.225 connection based on inspection of the ACF and RCF messages.

After inspecting the H.225 messages, the ASA opens the H.245 channel and then inspects traffic sent over the H.245 channel as well. All H.245 messages passing through the ASA undergo H.245 application inspection, which translates embedded IP addresses and opens the media channels negotiated in H.245 messages.

Each UDP connection with a packet going through H.323 inspection is marked as an H.323 connection and times out with the H.323 timeout as configured with the **timeout** command.

Note You can enable call setup between H.323 endpoints when the Gatekeeper is inside the network. The ASA includes options to open pinholes for calls based on the RegistrationRequest/RegistrationConfirm (RRQ/RCF) messages. Because these RRQ/RCF messages are sent to and from the Gatekeeper, the calling endpoint's IP address is unknown and the ASA opens a pinhole through source IP address/port 0/0. By default, this option is disabled. To enable call setup between H.323 endpoint, enter the **ras-rcf-pinholes enable** command during parameter configuration mode while creating an H.323 Inspection policy map.

H.239 Support in H.245 Messages

The ASA sits between two H.323 endpoints. When the two H.323 endpoints set up a telepresentation session so that the endpoints can send and receive a data presentation, such as spreadsheet data, the ASA ensure successful H.239 negotiation between the endpoints.

H.239 is a standard that provides the ability for H.300 series endpoints to open an additional video channel in a single call. In a call, an endpoint (such as a video phone), sends a channel for video and a channel for data presentation. The H.239 negotiation occurs on the H.245 channel.

The ASA opens pinholes for the additional media channel and the media control channel. The endpoints use open logical channel message (OLC) to signal a new channel creation. The message extension is part of H.245 version 13.

The decoding and encoding of the telepresentation session is enabled by default. H.239 encoding and decoding is preformed by ASN.1 coder.

Limitations for H.323 Inspection

H.323 inspection is tested and supported for Cisco Unified Communications Manager (CUCM) 7.0. It is not supported for CUCM 8.0 and higher. H.323 inspection might work with other releases and products.

The following are some of the known issues and limitations when using H.323 application inspection:

- PAT is supported except for extended PAT or per-session PAT.
- Static PAT may not properly translate IP addresses embedded in optional fields within H.323 messages. If you experience this kind of problem, do not use static PAT with H.323.
- Not supported with NAT between same-security-level interfaces.
- Not supported with NAT64.
- NAT with H.323 inspection is not compatible with NAT when done directly on the endpoints. If you perform NAT on the endpoints, disable H.323 inspection.

Configure H.323 Inspection Policy Map

You can create an H.323 inspection policy map to customize H.323 inspection actions if the default inspection behavior is not sufficient for your network.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create an H.323 inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect h323** [**match-all** | **match-any**] *class_map_name*

Where *class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The **match-any** keyword specifies that the traffic matches the class map if it matches at least one of the criteria. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **called-party regex** {*regex_name* | **class** *class_name*}—Matches the called party against the specified regular expression or regular expression class.
- **match** [**not**] **calling-party regex** {*regex_name* | **class** *class_name*}—Matches the calling party against the specified regular expression or regular expression class.
- **match** [**not**] **media-type** {**audio** | **data** | **video**}—Matches the media type.

Step 2 Create an H.323 inspection policy map: **policy-map type inspect h323** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created an H.323 class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for H.323 class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop** [**log**]—Drop the packet. For media type matches, you can include the **log** keyword to send a system log message.
- **drop-connection**—Drop the packet and close the connection. This option is available for called or calling party matching.
- **reset**—Drop the packet, close the connection, and send a TCP reset to the server and/or client. This option is available for called or calling party matching.

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **ras-rcf-pinholes enable**—Enables call setup between H.323 endpoints. You can enable call setup between H.323 endpoints when the Gatekeeper is inside the network. Use this option to open pinholes for calls based on the RegistrationRequest/RegistrationConfirm (RRQ/RCF) messages. Because these RRQ/RCF messages are sent to and from the Gatekeeper, the calling endpoint's IP address is unknown and the ASA opens a pinhole through source IP address/port 0/0. By default, this option is disabled.
- **timeout users** *time*—Sets the H.323 call duration limit (in hh:mm:ss format). To have no timeout, specify 00:00:00. Range is from 0:0:0 to 1193:0;0.
- **call-party-number**—Enforces sending call party number during call setup.
- **h245-tunnel-block action** {**drop-connection** | **log**}—Enforces H.245 tunnel blocking. Specify whether you want to drop the connection or simply log it.
- **rtp-conformance** [**enforce-payloadtype**]—Checks RTP packets flowing on the pinholes for protocol conformance. The optional enforce-payloadtype keyword enforces the payload type to be audio or video based on the signaling exchange.
- **state-checking** {**h225** | **ras**}—Enables state checking validation. You can enter the command separately to enable state checking for H.225 and RAS.
- **early-message** *message_type*—Whether to allow the specified type of H.225 messages before the H.225 SETUP message. You can allow the **facility** message to arrive early, in compliance with H.460.18.

  If you encounter call setup issues, where connections are being closed before being completed when using H.323/H.225, use this command to allow early messages. Also, ensure that you enable inspection for both H.323 RAS and H.225 (they are both enabled by default).

Step 6 While still in parameter configuration mode, you can configure HSI groups.

a) Define an HSI group and enter HSI group configuration mode: **hsi-group** *id*

   Where *id* is the HSI group ID. Range is from 0 to 2147483647.

b) Add an HSI to the HSI group using the IP address: **hsi** *ip_address*

   You can add a maximum of five hosts per HSI group.

c) Add an endpoint to the HSI group: **endpoint** *ip_address if_name*

   Where *ip_address* is the endpoint to add and *if_name* is the interface through which the endpoint is connected to the ASA. You can add a maximum of ten endpoints per HSI group.

Example

The following example shows how to configure phone number filtering:

```
hostname(config)# regex caller 1 "5551234567"
hostname(config)# regex caller 2 "5552345678"
hostname(config)# regex caller 3 "5553456789"

hostname(config)# class-map type inspect h323 match-all h323_traffic
hostname(config-pmap-c)# match called-party regex caller1
```

```
hostname(config-pmap-c)# match calling-party regex caller2

hostname(config)# policy-map type inspect h323 h323_map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# class h323_traffic
hostname(config-pmap-c)# drop
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

MGCP Inspection

MGCP inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. However, the default inspect class does include the default MGCP ports, so you can simply edit the default global inspection policy to add MGCP inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

The following sections describe MGCP application inspection.

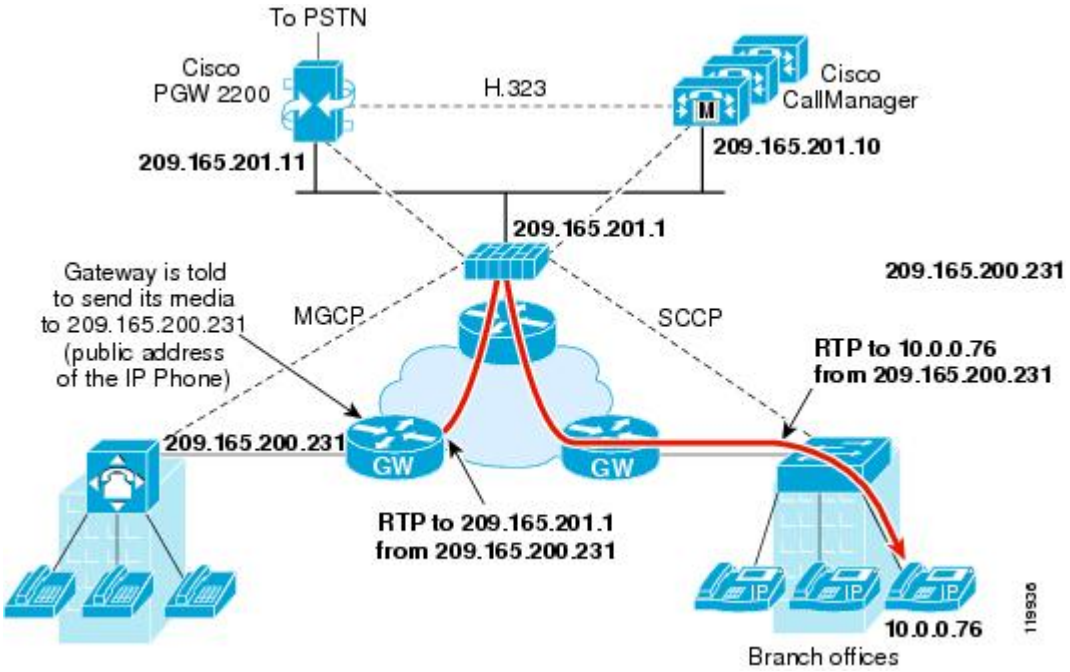
MGCP Inspection Overview

MGCP is used to control media gateways from external call control elements called media gateway controllers or call agents. A media gateway is typically a network element that provides conversion between the audio signals carried on telephone circuits and data packets carried over the Internet or over other packet networks. Using NAT and PAT with MGCP lets you support a large number of devices on an internal network with a limited set of external (global) addresses. Examples of media gateways are:

- Trunking gateways, that interface between the telephone network and a Voice over IP network. Such gateways typically manage a large number of digital circuits.
- Residential gateways, that provide a traditional analog (RJ11) interface to a Voice over IP network. Examples of residential gateways include cable modem/cable set-top boxes, xDSL devices, broad-band wireless devices.
- Business gateways, that provide a traditional digital PBX interface or an integrated soft PBX interface to a Voice over IP network.

MGCP messages are transmitted over UDP. A response is sent back to the source address (IP address and UDP port number) of the command, but the response may not arrive from the same address as the command was sent to. This can happen when multiple call agents are being used in a failover configuration and the call agent that received the command has passed control to a backup call agent, which then sends the response. The following figure illustrates how you can use NAT with MGCP.

Figure 41: Using NAT with MGCP



MGCP endpoints are physical or virtual sources and destinations for data. Media gateways contain endpoints on which the call agent can create, modify and delete connections to establish and control media sessions with other multimedia endpoints. Also, the call agent can instruct the endpoints to detect certain events and generate signals. The endpoints automatically communicate changes in service state to the call agent.

- Gateways usually listen to UDP port 2427 to receive commands from the call agent.
- The port on which the call agent receives commands from the gateway. Call agents usually listen to UDP port 2727 to receive commands from the gateway.

Note MGCP inspection does not support the use of different IP addresses for MGCP signaling and RTP data. A common and recommended practice is to send RTP data from a resilient IP address, such as a loopback or virtual IP address; however, the ASA requires the RTP data to come from the same address as MGCP signaling.

Configure an MGCP Inspection Policy Map

If the network has multiple call agents and gateways for which the ASA has to open pinholes, create an MGCP map. You can then apply the MGCP map when you enable MGCP inspection.

Procedure

Step 1 To create an MGCP inspection policy map: **policy-map type inspect mgcp** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

Step 4 Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option.

- **call-agent** *ip_address group_id*—Configures the call agent groups that can manage one or more gateways. The call agent group information is used to open connections for the call agents in the group (other than the one a gateway sends a command to) so that any of the call agents can send the response. Call agents with the same group_id belong to the same group. A call agent may belong to more than one group. The group_id option is a number from 0 to 4294967295. The ip_address option specifies the IP address of the call agent.

  Note
  MGCP call agents send AUEP messages to determine if MGCP end points are present. This establishes a flow through the ASA and allows MGCP end points to register with the call agent.

- **gateway** *ip_address group_id*—Identifies which group of call agents is managing a particular gateway. The IP address of the gateway is specified with the ip_address option. The group_id option is a number from 0 to 4294967295 that must correspond with the group_id of the call agents that are managing the gateway. A gateway may only belong to one group.
- **command-queue** *command_limit*—Sets the maximum number of commands allowed in the MGCP command queue, from 1 to 2147483647. The default is 200.

Example

The following example shows how to define an MGCP map:

```
hostname(config)# policy-map type inspect mgcp sample_map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# call-agent 10.10.11.5 101
hostname(config-pmap-p)# call-agent 10.10.11.6 101
hostname(config-pmap-p)# call-agent 10.10.11.7 102
hostname(config-pmap-p)# call-agent 10.10.11.8 102
hostname(config-pmap-p)# gateway 10.10.10.115 101
hostname(config-pmap-p)# gateway 10.10.10.116 102
hostname(config-pmap-p)# gateway 10.10.10.117 102
hostname(config-pmap-p)# command-queue 150
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

RTSP Inspection

RTSP inspection is enabled by default. You need to configure it only if you want non-default processing. The following sections describe RTSP application inspection.

RTSP Inspection Overview

The RTSP inspection engine lets the ASA pass RTSP packets. RTSP is used by RealAudio, RealNetworks, Apple QuickTime 4, RealPlayer, and Cisco IP/TV connections.



Note For Cisco IP/TV, use RTSP TCP ports 554 and 8554.

RTSP applications use the well-known port 554 with TCP (rarely UDP) as a control channel. The ASA only supports TCP, in conformity with RFC 2326. This TCP control channel is used to negotiate the data channels that are used to transmit audio/video traffic, depending on the transport mode that is configured on the client.

The supported RDT transports are: rtp/avp, rtp/avp/udp, x-real-rdt, x-real-rdt/udp, and x-pn-tng/udp.

The ASA parses Setup response messages with a status code of 200. If the response message is traveling inbound, the server is outside relative to the ASA and dynamic channels need to be opened for connections coming inbound from the server. If the response message is outbound, then the ASA does not need to open dynamic channels.

RTSP inspection does not support PAT or dual-NAT. Also, the ASA cannot recognize HTTP cloaking where RTSP messages are hidden in the HTTP messages.

RealPlayer Configuration Requirements

When using RealPlayer, it is important to properly configure transport mode. For the ASA, add an **access-list** command from the server to the client or vice versa. For RealPlayer, change transport mode by clicking **Options**>**Preferences**>**Transport**>**RTSP Settings**.

If using TCP mode on the RealPlayer, select the **Use TCP to Connect to Server** and **Attempt to use TCP for all content** check boxes. On the ASA, there is no need to configure the inspection engine.

If using UDP mode on the RealPlayer, select the **Use TCP to Connect to Server** and **Attempt to use UDP for static content** check boxes, and for live content not available via multicast. On the ASA, add an **inspect rtsp** command.

Limitations for RSTP Inspection

The following restrictions apply to the RSTP inspection.

- The ASA does not support multicast RTSP or RTSP messages over UDP.
- The ASA does not have the ability to recognize HTTP cloaking where RTSP messages are hidden in the HTTP messages.

- With Cisco IP/TV, the number of translates the ASA performs on the SDP part of the message is proportional to the number of program listings in the Content Manager (each program listing can have at least six embedded IP addresses).
- You can configure NAT for Apple QuickTime 4 or RealPlayer. Cisco IP/TV only works with NAT if the Viewer and Content Manager are on the outside network and the server is on the inside network.

Configure RTSP Inspection Policy Map

You can create an RTSP inspection policy map to customize RTSP inspection actions if the default inspection behavior is not sufficient for your network.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create an RTSP inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string "example.com," then any traffic that includes "example.com" does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect rtsp** [**match-all** | **match-any**] *class_map_name*

Where *class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The **match-any** keyword specifies that the traffic matches the class map if it matches at least one of the criteria. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **request-method** *method*—Matches an RTSP request method. The methods are: announce, describe, get_parameter, options, pause, play, record, redirect, setup, set_parameter, teardown.

- **match** [**not**] **url-filter regex** {*regex_name* | **class** *class_name*}—Matches the URL against the specified regular expression or regular expression class.

Step 2 To create an RTSP inspection policy map: **policy-map type inspect rtsp** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created an RTSP class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for RTSP class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop-connection** [**log**]—Drop the packet, close the connection, and optionally send a system log message. This option is available for URL matching.
- **log**—Send a system log message.
- **rate-limit** *message_rate*—Limit the rate of messages per second. This option is available for request method matching.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **reserve-port-protect**—Restricts the use of reserve ports during media negotiation.
- **url-length-limit** *bytes*—Sets a limit on the URL length allowed in the message, from 0 to 6000 bytes.

Example

The following example shows a how to define an RTSP inspection policy map.

```
hostname(config)# regex badurl1 www.url1.com/rtsp.avi
hostname(config)# regex badurl2 www.url2.com/rtsp.rm
```

```
hostname(config)# regex badurl3 www.url3.com/rtsp.asp

hostname(config)# class-map type regex match-any badurl-list
hostname(config-cmap)# match regex badurl1
hostname(config-cmap)# match regex badurl2
hostname(config-cmap)# match regex badurl3

hostname(config)# policy-map type inspect rtsp rtsp-filter-map
hostname(config-pmap)# match url-filter regex class badurl-list
hostname(config-pmap-p)# drop-connection

hostname(config)# class-map rtsp-traffic-class
hostname(config-cmap)# match default-inspection-traffic

hostname(config)# policy-map rtsp-traffic-policy
hostname(config-pmap)# class rtsp-traffic-class
hostname(config-pmap-c)# inspect rtsp rtsp-filter-map

hostname(config)# service-policy rtsp-traffic-policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

SIP Inspection

SIP is a widely used protocol for Internet conferencing, telephony, presence, events notification, and instant messaging. Partially because of its text-based nature and partially because of its flexibility, SIP networks are subject to a large number of security threats.

SIP application inspection provides address translation in message header and body, dynamic opening of ports and basic sanity checks. It also supports application security and protocol conformance, which enforce the sanity of the SIP messages, as well as detect SIP-based attacks.

SIP inspection is enabled by default. You need to configure it only if you want non-default processing, or if you want to identify a TLS proxy to enable encrypted traffic inspection. The following topics explain SIP inspection in more detail.

SIP Inspection Overview

SIP, as defined by the IETF, enables call handling sessions, particularly two-party audio conferences, or “calls.” SIP works with SDP for call signaling. SDP specifies the ports for the media stream. Using SIP, the ASA can support any SIP VoIP gateways and VoIP proxy servers. SIP and SDP are defined in the following RFCs:

- SIP: Session Initiation Protocol, RFC 3261
- SDP: Session Description Protocol, RFC 2327

To support SIP calls through the ASA, signaling messages for the media connection addresses, media ports, and embryonic connections for the media must be inspected, because while the signaling is sent over a well-known destination port (UDP/TCP 5060), the media streams are dynamically allocated. Also, SIP embeds

IP addresses in the user-data portion of the IP packet. Note that the maximum length of the SIP Request URI that the ASA supports is 255.

Instant Messaging (IM) applications also use SIP extensions (defined in RFC 3428) and SIP-specific event notifications (RFC 3265). After users initiate a chat session (registration/subscription), the IM applications use the MESSAGE/INFO methods and 202 Accept responses when users chat with each other. For example, two users can be online at any time, but not chat for hours. Therefore, the SIP inspection engine opens pinholes that time out according to the configured SIP timeout value. This value must be configured at least five minutes longer than the subscription duration. The subscription duration is defined in the Contact Expires value and is typically 30 minutes.

Because MESSAGE/INFO requests are typically sent using a dynamically allocated port other than port 5060, they are required to go through the SIP inspection engine.



Note SIP inspection supports the Chat feature only. Whiteboard, File Transfer, and Application Sharing are not supported. RTC Client 5.0 is not supported.

Limitations for SIP Inspection

SIP inspection is tested and supported for Cisco Unified Communications Manager (CUCM) 7.0, 8.0, 8.6, and 10.5. It is not supported for CUCM 8.5, or 9.x. SIP inspection might work with other releases and products.

If you find that SIP phones are not connecting to the call manager, you can try increasing the maximum number of unprocessed TCP segments in the CLI using the following command: **sysopt connection tcp-max-unprocessed-seg** *6-24*. The default is 6, so try a higher number.

SIP inspection does not support the T.38 MIME Internet Facsimile Protocol(IFP). SIP inspection drops SIP invitations that use the T.38 MIME audio sub-type. If you need to allow this type, disable SIP inspection and write an access control rule that allows the RTP streams.

NAT Limitations for SIP Inspection

- SIP inspection applies NAT for embedded IP addresses. However, if you configure NAT to translate both source and destination addresses, the external address ("from" in the SIP header for the "trying" response message) is not rewritten. Thus, you should use object NAT when working with SIP traffic so that you avoid translating the destination address.
- Do not configure NAT or PAT for interfaces with the same, or lower (source) to higher (destination), security levels. This configuration is not supported.
- If you configure SIP inspection for a targeted traffic class (that is, not the inspection_default traffic class), ensure that you use a bi-directional ACL and that you specify the 5060 destination port only. Otherwise, you might have NAT problems where the IP address in the SIP header is not translated even though the IP packet is correctly translated.
- If you hard-code the mapped address in the VIA header of the SIP invitation, do not enable SIP inspection. You can run into problems if you use static NAT to translate the source client address, and the interface for the default route is different than the interface for the connected route used by the client.

PAT Limitations for SIP Inspection

The following limitations and restrictions apply when using PAT with SIP:

- If a remote endpoint tries to register with a SIP proxy on a network protected by the ASA, the registration fails under very specific conditions, as follows:
  - PAT is configured for the remote endpoint.
  - The SIP registrar server is on the outside network.
  - The port is missing in the contact field in the REGISTER message sent by the endpoint to the proxy server.
- If a SIP device transmits a packet in which the SDP portion has an IP address in the owner/creator field (o=) that is different than the IP address in the connection field (c=), the IP address in the o= field may not be properly translated. This is due to a limitation in the SIP protocol, which does not provide a port value in the o= field. Because PAT needs a port to translate, the translation fails.
- When using PAT, any SIP header field which contains an internal IP address without a port might not be translated and hence the internal IP address will be leaked outside. If you want to avoid this leakage, configure NAT instead of PAT.

Default SIP Inspection

SIP inspection is enabled by default using the default inspection map, which includes the following:

- SIP instant messaging (IM) extensions: Enabled.
- Non-SIP traffic on SIP port: Dropped.
- Hide server's and endpoint's IP addresses: Disabled.
- Mask software version and non-SIP URIs: Disabled.
- Ensure that the number of hops to destination is greater than 0: Enabled.
- RTP conformance: Not enforced.
- SIP conformance: Do not perform state checking and header validation.

Also note that inspection of encrypted traffic is not enabled. You must configure a TLS proxy to inspect encrypted traffic.

Configure SIP Inspection Policy Map

You can create a SIP inspection policy map to customize SIP inspection actions if the default inspection behavior is not sufficient for your network.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create a SIP inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string "example.com," then any traffic that includes "example.com" does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect sip** [**match-all** | **match-any**] *class_map_name*

Where *the class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one **match** statement. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **called-party regex** {*regex_name* | **class** *class_name*}—Matches the called party, as specified in the To header, against the specified regular expression or regular expression class.
- **match** [**not**] **calling-party regex** {*regex_name* | **class** *class_name*}—Matches the calling party, as specified in the From header, against the specified regular expression or regular expression class.
- **match** [**not**] **content length gt** *bytes*—Matches messages where the content length in the SIP header is greater than the specified number of bytes, from 0 to 65536.
- **match** [**not**] **content type** {**sdp** | **regex** {*regex_name* | **class** *class_name*}—Matches the content type as SDP or against the specified regular expression or regular expression class.
- **match** [**not**] **im-subscriber regex** {*regex_name* | **class** *class_name*}—Matches the SIP IM subscriber against the specified regular expression or regular expression class.
- **match** [**not**] **message-path regex** {*regex_name* | **class** *class_name*}—Matches the SIP via header against the specified regular expression or regular expression class.
- **match** [**not**] **request-method** *method*—Matches a SIP request method: ack, bye, cancel, info, invite, message, notify, options, prack, refer, register, subscribe, unknown, update.
- **match** [**not**] **third-party-registration regex** {*regex_name* | **class** *class_name*}—Matches the requester of a third-party registration against the specified regular expression or regular expression class.

- **match** [**not**] **uri {sip | tel} length gt** *bytes*—Matches a URI in the SIP headers of the selected type (SIP or TEL) that is greater than the length specified, between 0 and 65536 bytes.

d) Enter **exit** to leave class map configuration mode.

Step 2 Create a SIP inspection policy map: **policy-map type inspect sip** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created a SIP class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for SIP class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop**—Drop all packets that match.
- **drop-connection**—Drop the packet and close the connection.
- **reset**—Drop the packet, close the connection, and send a TCP reset to the server and/or client.
- **log**—Send a system log message. You can use this option alone or with one of the other actions.
- **rate-limit** *message_rate*—Limits the rate of messages. Rate limiting is available for request method matches to “invite” and “register” only.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **im**—Enables instant messaging.
- **ip-address-privacy**—Enables IP address privacy, which hides the server and endpoint IP addresses.
- **max-forwards-validation action** {**drop** | **drop-connection** | **reset** | **log**} [**log**]—Checks the value of the Max-Forwards header, which cannot be zero before reaching the destination. You must also choose the action to take for non-conforming traffic (drop packet, drop connection, reset, or log) and whether to enable or disable logging.
- **rtp-conformance** [**enforce-payloadtype**]—Checks RTP packets flowing on the pinholes for protocol conformance. The optional enforce-payloadtype keyword enforces the payload type to be audio or video based on the signaling exchange.

- **software-version action** {**mask [log]** | **log**}—Identifies the software version using the Server and User-Agent (endpoint) header fields. You can mask the software version in the SIP messages and optionally log it, or simply log it.
- **state-checking action** {**drop** | **drop-connection** | **reset** | **log**} [**log**]—Enables state transition checking. You must also choose the action to take for non-conforming traffic (drop packet, drop connection, reset, or log) and whether to enable or disable logging.
- **strict-header-validation action** {**drop** | **drop-connection** | **reset** | **log**} [**log**]—Enables strict verification of the header fields in the SIP messages according to RFC 3261. You must also choose the action to take for non-conforming traffic (drop packet, drop connection, reset, or log) and whether to enable or disable logging.
- **traffic-non-sip**—Allows non-SIP traffic on the well-known SIP signaling port.
- **trust-verification-server ip** *ip_address*—Identifies Trust Verification Services servers, which enable Cisco Unified IP Phones to authenticate application servers during HTTPS establishment. You can enter the command up to four times to identify four servers. SIP inspection opens pinholes to each server for each registered phone, and the phone decides which to use. Configure the Trust Verification Services server on the CUCM server.
- **trust-verification-server port** *number*—Identifies the Trust Verification Services port. The default port is 2445, so use this command only if the server uses a different port. The allowed port range is 1026 to 32768.
- **uri-non-sip action** {**mask [log]** | **log**}—Identifies the non-SIP URIs present in the Alert-Info and Call-Info header fields. You can mask the information in the SIP messages and optionally log it, or simply log it.

Examples

The following example shows how to disable instant messaging over SIP:

```
hostname(config)# policy-map type inspect sip mymap
hostname(config-pmap)# parameters
hostname(config-pmap-p)# no im

hostname(config)# policy-map global_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect sip mymap

hostname(config)# service-policy global_policy global
```

The following example shows how to identify four Trust Verification Services servers.

```
hostname(config)# policy-map type inspect sip sample_sip_map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# trust-verification-server ip 10.1.1.1
hostname(config-pmap-p)# trust-verification-server ip 10.1.1.2
hostname(config-pmap-p)# trust-verification-server ip 10.1.1.3
hostname(config-pmap-p)# trust-verification-server ip 10.1.1.4
hostname(config-pmap-p)# trust-verification-server port 2445
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

Skinny (SCCP) Inspection

SCCP (Skinny) application inspection performs translation of embedded IP address and port numbers within the packet data, and dynamic opening of pinholes. It also performs additional protocol conformance checks and basic state tracking.

SCCP inspection is enabled by default. You need to configure it only if you want non-default processing, or if you want to identify a TLS proxy to enable encrypted traffic inspection.

The following sections describe SCCP application inspection.

SCCP Inspection Overview

Skinny (SCCP) is a simplified protocol used in VoIP networks. Cisco IP Phones using SCCP can coexist in an H.323 environment. When used with Cisco CallManager, the SCCP client can interoperate with H.323 compliant terminals.

The ASA supports PAT and NAT for SCCP. PAT is necessary if you have more IP phones than global IP addresses for the IP phones to use. By supporting NAT and PAT of SCCP Signaling packets, Skinny application inspection ensures that all SCCP signaling and media packets can traverse the ASA.

Normal traffic between Cisco CallManager and Cisco IP Phones uses SCCP and is handled by SCCP inspection without any special configuration. The ASA also supports DHCP options 150 and 66, which it accomplishes by sending the location of a TFTP server to Cisco IP Phones and other DHCP clients. Cisco IP Phones might also include DHCP option 3 in their requests, which sets the default route.

Note The ASA supports inspection of traffic from Cisco IP Phones running SCCP protocol version 22 and earlier.

Supporting Cisco IP Phones

In topologies where Cisco CallManager is located on the higher security interface with respect to the Cisco IP Phones, if NAT is required for the Cisco CallManager IP address, the mapping must be static as a Cisco IP Phone requires the Cisco CallManager IP address to be specified explicitly in its configuration. A static identity entry allows the Cisco CallManager on the higher security interface to accept registrations from the Cisco IP Phones.

Cisco IP Phones require access to a TFTP server to download the configuration information they need to connect to the Cisco CallManager server.

When the Cisco IP Phones are on a lower security interface compared to the TFTP server, you must use an ACL to connect to the protected TFTP server on UDP port 69. While you do need a static entry for the TFTP server, this does not have to be an identity static entry. When using NAT, an identity static entry maps to the same IP address. When using PAT, it maps to the same IP address and port.

When the Cisco IP Phones are on a higher security interface compared to the TFTP server and Cisco CallManager, no ACL or static entry is required to allow the Cisco IP Phones to initiate the connection.

Limitations for SCCP Inspection

SCCP inspection is tested and supported for Cisco Unified Communications Manager (CUCM) 7.0, 8.0, 8.6, and 10.5. It is not supported for CUCM 8.5, or 9.x. SCCP inspection might work with other releases and products.

If the address of an internal Cisco CallManager is configured for NAT or PAT to a different IP address or port, registrations for external Cisco IP Phones fail because the ASA does not support NAT or PAT for the file content transferred over TFTP. Although the ASA supports NAT of TFTP messages and opens a pinhole for the TFTP file, the ASA cannot translate the Cisco CallManager IP address and port embedded in the Cisco IP Phone configuration files that are transferred by TFTP during phone registration.

Note The ASA supports stateful failover of SCCP calls except for calls that are in the middle of call setup.

Default SCCP Inspection

SCCP inspection is enabled by default using these defaults:

- Registration: Not enforced.
- Maximum message ID: 0x181.
- Minimum prefix length: 4
- Media timeout: 00:05:00
- Signaling timeout: 01:00:00.
- RTP conformance: Not enforced.

Configure a Skinny (SCCP) Inspection Policy Map

To specify actions when a message violates a parameter, create an SCCP inspection policy map. You can then apply the inspection policy map when you enable SCCP inspection.

Procedure

Step 1 Create an SCCP inspection policy map: **policy-map type inspect skinny** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 (Optional) Drop traffic based on the station message ID field in SCCP messages.

a) Identify the traffic based on the station message ID value in hexadecimal, from 0x0 to 0xffff. You can either specify a single ID, or a range of IDs, using the **match** [**not**] **message-id** command. If you use a

match not command, then any traffic that does not match the criterion in the **match not** command has the action applied.

match message-id {*value* | **range** *start_value end_value*}

Example:

```
hostname(config-pmap)# match message-id 0x181
hostname(config-pmap)# match message-id range 0x200 0xffff
```

b) Specify the action to perform on matching packets. You can drop the packet and optionally log it: **drop** [**log**]

c) Repeat the process until you identify all message IDs that you want to drop.

Step 4 Configure parameters that affect the inspection engine.

a) Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **enforce-registration**—Enforces registration before calls can be placed.
- **message-ID max** *hex_value*—Sets the maximum SCCP station message ID allowed. The message ID is in hex, and the default maximum is 0x181.
- **rtp-conformance** [**enforce-payloadtype**]—Checks RTP packets flowing on the pinholes for protocol conformance. The optional enforce-payloadtype keyword enforces the payload type to be audio or video based on the signaling exchange.
- **sccp-prefix-len** {**max** | **min**} *length*—Sets the maximum or minimum SCCP prefix length value allowed. Enter the command twice to set both a minimum and maximum value. The default minimum is 4, there is no default maximum.
- **timeout** {**media** | **signaling**} *time*—Sets the timeouts for media and signaling connections (in hh:mm:ss format). To have no timeout, specify 0 for the number. The default media timeout is 5 minutes, the default signaling timeout is one hour.

Example

The following example shows how to define an SCCP inspection policy map.

```
hostname(config)# policy-map type inspect skinny skinny-map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# enforce-registration
hostname(config-pmap-p)# match message-id range 200 300
hostname(config-pmap-p)# drop log
hostname(config)# class-map inspection_default
hostname(config-cmap)# match default-inspection-traffic
hostname(config)# policy-map global_policy
```

```
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect skinny skinny-map
hostname(config)# service-policy global_policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection, on page 253.

STUN Inspection

Session Traversal Utilities for NAT (STUN), defined in RFC 5389, is used by WebRTC clients for browser-based real-time communications so that plug-ins are not necessary. WebRTC clients often use cloud STUN servers to learn their public IP addresses and ports. WebRTC uses Interactive Connectivity Establishment (ICE, RFC 5245) to verify connectivity between clients. These clients typically use UDP, although they can also use TCP or other protocols.

Because firewalls often block outgoing UDP traffic, WebRTC products such as Cisco Spark can have problems completing connections. STUN inspection opens pinholes for STUN endpoints, and enforces basic STUN and ICE compliance, to allow communications for clients if the connectivity check is acknowledged by both sides. Thus, you can avoid opening new ports in your access rules to enable these applications.

When you enable STUN inspection on the default inspection class, TCP/UDP port 3478 is watched for STUN traffic. The inspection supports IPv4 addresses and TCP/UDP only.

There are some NAT limitations for STUN inspection. For WebRTC traffic, static NAT/PAT44 are supported. Cisco Spark can support additional types of NAT, because Spark does not require pinholes. You can also use NAT/PAT64, including dynamic NAT/PAT, with Cisco Spark.

STUN inspection is supported in failover and cluster modes, as pinholes are replicated. However, the transaction ID is not replicated among nodes. In the case where a node fails after receiving a STUN Request and another node received the STUN Response, the STUN Response will be dropped.

Note STUN inspection uses transaction IDs to match requests and responses. If you use debug to troubleshoot connection drops, note that the system changes the format (endianness) of the IDs for the debug output, so they do not compare directly to those you might see in a pcap.

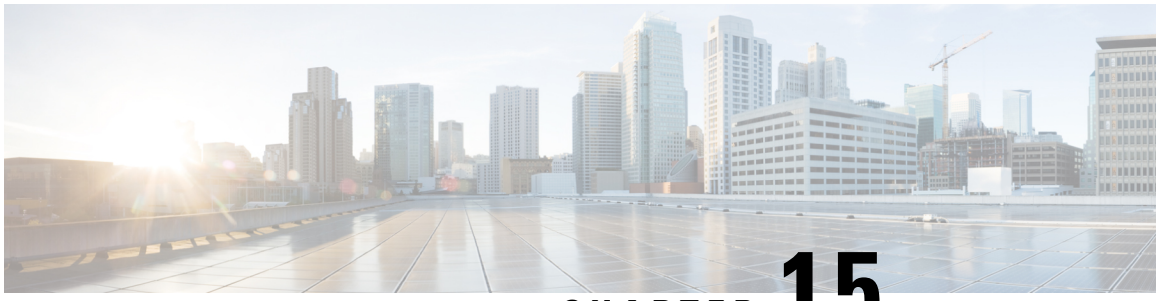
For information on enabling STUN inspection, see Configure Application Layer Protocol Inspection, on page 253.

History for Voice and Video Protocol Inspection

Feature Name	Releases	Feature Information
SIP, SCCP, and TLS Proxy support for IPv6	9.3(1)	You can now inspect IPv6 traffic when using SIP, SCCP, and TLS Proxy (using SIP or SCCP). We did not modify any commands.

Feature Name	Releases	Feature Information
SIP support for Trust Verification Services, NAT66, CUCM 10.5, and model 8831 phones.	9.3(2)	You can now configure Trust Verification Services servers in SIP inspection. You can also use NAT66. SIP inspection has been tested with CUCM 10.5. We added the **trust-verification-server** parameter command.
Improved SIP inspection performance on multiple core ASA.	9.4(1)	If you have multiple SIP signaling flows going through an ASA with multiple cores, SIP inspection performance has been improved. However, you will not see improved performance if you are using a TLS, phone, or IME proxy. We did not modify any commands.
SIP inspection support in ASA clustering	9.4(1)	You can now configure SIP inspection on the ASA cluster. A control flow can be created on any unit (due to load balancing), but its child data flows must reside on the same unit. TLS Proxy configuration is not supported. We introduced the following command: **show cluster service-policy**.
SIP inspection support for Phone Proxy and UC-IME Proxy was removed.	9.4(1)	You can no longer use Phone Proxy or UC-IME Proxy when configuring SIP inspection. Use TLS Proxy to inspect encrypted traffic. We removed the following commands: **phone-proxy**, **uc-ime**. We removed the **phone-proxy** and **uc-ime** keywords from the **inspect sip** command.
H.323 inspection support for the H.255 FACILITY message coming before the H.225 SETUP message for H.460.18 compatibility.	9.6(1)	You can now configure an H.323 inspection policy map to allow for H.225 FACILITY messages to come before the H.225 SETUP message, which can happen when endpoints comply with H.460.18. We introduced the following command: **early-message**.
Session Traversal Utilities for NAT (STUN) inspection.	9.6(2)	You can now inspect STUN traffic for WebRTC applications including Cisco Spark. Inspection opens pinholes required for return traffic. We added or modified the following commands: **inspect stun**, **show asp drop**, **show conn detail**, **show service-policy inspect stun**.
Support for TLSv1.2 in TLS proxy and Cisco Unified Communications Manager 10.5.2.	9.7(1)	You can now use TLSv1.2 with TLS proxy for encrypted SIP or SCCP inspection with the Cisco Unified Communications Manager 10.5.2. The TLS proxy supports the additional TLSv1.2 cipher suites added as part of the **client cipher-suite** command. We modified the following commands: **client cipher-suite**.
TLS proxy deprecated for SCCP (Skinny) inspection.	9.13(1)	The **tls-proxy** keyword, and support for SCCP/Skinny encrypted inspection, was deprecated. The keyword will be removed from the **inspect skinny** command in a future release.

Feature Name	Releases	Feature Information
TLS proxy support eliminated for SCCP (Skinny) inspection.	9.14(1)	The **tls-proxy** keyword, and support for SCCP/Skinny encrypted inspection, was removed.
The default SIP inspection policy map drops non-SIP traffic.	9.16(1)	For SIP-inspected traffic, the default is now to drop non-SIP traffic. The previous default was to allow non-SIP traffic on ports inspected for SIP. We changed the default SIP policy map to include the **no traffic-non-sip** command.



CHAPTER 15

Inspection for Mobile Networks

The following topics explain application inspection for protocols used in mobile networks such as LTE. These inspections require the Carrier license. For information on why you need to use inspection for certain protocols, and the overall methods for applying inspection, see Getting Started with Application Layer Protocol Inspection, on page 245.

- Mobile Network Inspection Overview, on page 329
- Licensing for Mobile Network Protocol Inspection, on page 336
- Defaults for GTP Inspection, on page 336
- Configure Mobile Network Inspection, on page 337
- Monitoring Mobile Network Inspection, on page 367
- History for Mobile Network Inspection, on page 372

Mobile Network Inspection Overview

The following topics explain the inspections available for protocols used in mobile networks such as LTE. There are other services available for SCTP traffic in addition to inspection.

GTP Inspection Overview

GPRS Tunneling Protocol is used in GSM, UMTS and LTE networks for general packet radio service (GPRS) traffic. GTP provides a tunnel control and management protocol to provide GPRS network access for a mobile station by creating, modifying, and deleting tunnels. GTP also uses a tunneling mechanism for carrying user data packets.

Service provider networks use GTP to tunnel multi-protocol packets through the GPRS backbone between endpoints. In GTPv0-1, GTP is used for signaling between gateway GPRS support nodes (GGSN) and serving GPRS support nodes (SGSN). In GTPv2, the signaling is between Packet Data Network Gateways (PGW) and the Serving Gateway (SGW) as well as other endpoints. The GGSN/PGW is the interface between the GPRS wireless data network and other networks. The SGSN/SGW performs mobility, data session management, and data compression.

You can use the ASA to provide protection against rogue roaming partners. Place the device between the home GGSN/PGW and visited SGSN/SGW endpoints and use GTP inspection on the traffic. GTP inspection works only on traffic between these endpoints. In GTPv2, this is known as the S5/S8 interface.

GTP and associated standards are defined by 3GPP (3rd Generation Partnership Project). For detailed information, see http://www.3gpp.org.

Tracking Location Changes for Mobile Stations

You can use GTP inspection to track location changes for mobile stations. Tracking location changes might help you identify fraudulent roaming charges, for example, if you see a mobile station move from one location to another within an unlikely time window, such as moving from a cell in the United States to one in Europe within 30 minutes.

When you enable location logging, the system generates syslog messages for new or changed location for each International Mobile Subscriber Identity (IMSI):

- 324010 indicates the creation of a new PDP context, and includes the Mobile Country Code (MCC), Mobile Network Code (MNC), the information elements, and optionally the cell ID where the user currently is registered. The cell ID is extracted from the Cell Global Identification (CGI) or E-UTRAN Cell Global Identifier (ECGI).
- 324011 indicates that the IMSI has moved from the one stored during the PDP context creation. The message shows the previous and current MCC/MNC, information elements, and optionally, cell ID.

By default, syslog messages do not include timestamp information. If you plan to analyze these messages to identify improbable roaming, you must also enable timestamps. Timestamp logging is not part of the GTP inspection map. Use the **logging timestamp** command.

For information on enabling location logging, see Configure a GTP Inspection Policy Map, on page 337.

GTP Inspection Limitations

Following are some limitations on GTP inspection:

- GTPv2 piggybacking messages are not supported. They are always dropped.
- GTPv2 emergency UE attach is supported only if it contains IMSI (International Mobile Subscriber Identity).
- GTP inspection does not inspect early data. That is, data sent from a PGW or SGW right after a Create Session Request but before the Create Session Response.
- For GTPv2, inspection supports up to 3GPP 29.274 V15.5.0. For GTPv1, support is up to 3GPP 29.060 V15.2.0. For GTPv0, support is up to release 8.
- GTP inspection does not support inter-SGSN handoff to the secondary PDP context. Inspection needs to do the handoff for both primary and secondary PDP contexts.
- When you enable GTP inspection, connections that use GTP-in-GTP encapsulation are always dropped.

Stream Control Transmission Protocol (SCTP) Inspection and Access Control

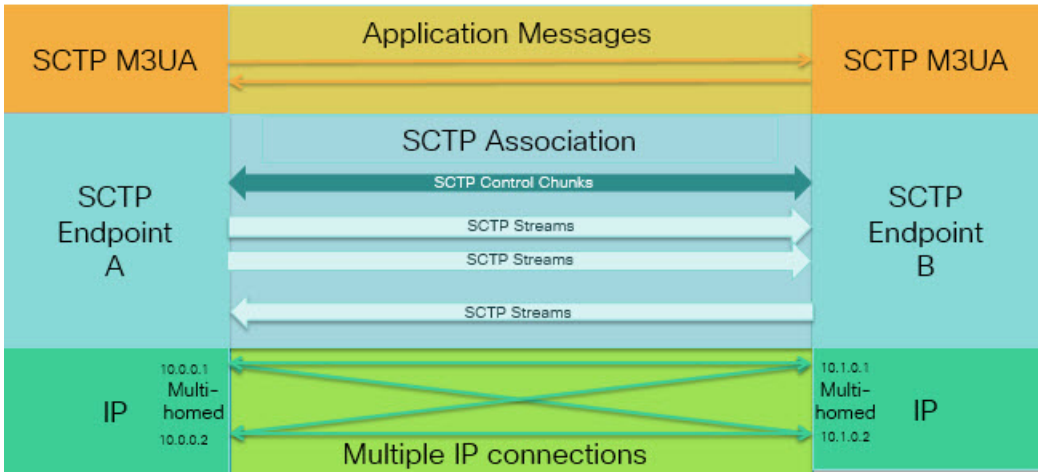
SCTP (Stream Control Transmission Protocol) is described in RFC 4960. The protocol supports the telephony signaling protocol SS7 over IP and is also a transport protocol for several interfaces in the 4G LTE mobile network architecture.

SCTP is a transport-layer protocol operating on top of IP in the protocol stack, similar to TCP and UDP. However, SCTP creates a logical communication channel, called an association, between two end nodes over one or more source or destination IP addresses. This is called multi-homing. An association defines a set of IP addresses on each node (source and destination) and a port on each node. Any IP address in the set can be used as either a source or a destination IP address of data packets associated to this association to form multiple

connections. Within each connection, multiple streams may exist to send messages. A stream in SCTP represents a logical application data channel.

The following figure illustrates the relationship between an association and its streams.

Figure 42: Relationship Between SCTP Association and Streams



If you have SCTP traffic going through the ASA, you can control access based on SCTP ports, and implement application layer inspection to enable connections and to optionally filter on payload protocol ID to selectively drop, log, or rate limit applications.

Note Each node can have up to three IP addresses. Any addresses over the limit of three are ignored and not included in the association. Pinholes for secondary IP addresses are opened automatically. You do not need to write access control rules to allow them.

Heartbeats and Association State

To maintain the association and ensure accurate reporting, the inspection engine monitors SCTP heartbeats. The system links the connection duration to the heartbeat signals. If the heartbeat is lost or the association is explicitly terminated, the device closes the connection and finalizes the event duration.

SCTP Association and Port Reporting

SCTP inspection is association-based rather than flow-based. Because the device tracks the entire association as a single unit, it does not map individual stream ports to traditional port fields in the logging output. As a result, when viewing connection details, you may observe the source and destination ports displayed as `0`. This is because the association is uniquely identified by the IP addresses and the association ID, rather than by individual source/destination port pairs.

SCTP Stateful Inspection

Similar to TCP, SCTP traffic is automatically inspected at layer 4 to ensure well-structured traffic and limited RFC 4960 enforcement. The following protocol elements are inspected and enforced:

- Chunk types, flags, and length.

- Verification tags.
- Source and destination ports, to prevent association redirect attacks.
- IP addresses.

SCTP stateful inspection accepts or rejects packets based on the association state:

- Validating the 4-way open and close sequences for initial association establishment.
- Verifying the forward progression of TSN within an association and a stream.
- Terminating an association when seeing the ABORT chunk due to heartbeat failure. SCTP endpoints might send the ABORT chunk in response to bombing attacks.

If you decide you do not want these enforcement checks, you can configure SCTP state bypass for specific traffic classes, as explained in Configure Connection Settings for Specific Traffic Classes (All Services), on page 400.

SCTP Access Control

You can create access rules for SCTP traffic. These rules are similar to TCP/UDP port-based rules, where you simply use **sctp** as the protocol, and the port numbers are SCTP ports. You can create service objects or groups for SCTP, or specify the ports directly. See the following topics.

- Configure Service Objects and Service Groups, on page 12
- Add an Extended ACE for Port-Based Matching, on page 35

SCTP NAT

You can apply static network object NAT to the addresses in SCTP association establishment messages. Although you can configure static twice NAT, this is not recommended because the topology of the destination part of the SCTP association is unknown. You cannot use dynamic NAT/PAT.

NAT for SCTP depends upon SCTP stateful inspection rather than SCTP application-layer inspection. Thus, you cannot NAT traffic if you configure SCTP state bypass.

SCTP Application Layer Inspection

You can further refine your access rules by enabling SCTP inspection and filtering on SCTP applications. You can selectively drop, log, or rate limit SCTP traffic classes based on the payload protocol identifier (PPID).

If you decide to filter on PPID, keep the following in mind:

- PPIDs are in data chunks, and a given packet can have multiple data chunks or even a control chunk. If a packet includes a control chunk or multiple data chunks, the packet will not be dropped even if the assigned action is drop.
- If you use PPID filtering to drop or rate-limit packets, be aware that the transmitter will resend any dropped packets. Although a packet for a rate-limited PPID might make it through on the next attempt, a packet for a dropped PPID will again be dropped. You might want to evaluate the eventual consequence of these repeated drops on your network.

SCTP Limitations

SCTP support includes the following limitations.

- Each node can have up to three IP addresses. Any addresses over the limit of three are ignored and not included in the association. Pinholes for secondary IP addresses are opened automatically. You do not need to write access control rules to allow them.
- Unused pinholes time out in 5 minutes.
- Dual stack IPv4 and IPv6 addresses on multi-homed endpoints is not supported.
- Network object static NAT is the only supported type of NAT. Also, NAT46 and NAT64 are not supported.
- Fragmentation and reassembly of SCTP packets is done only for traffic handled by Diameter, M3UA, and SCTP PPID-based inspection.
- ASCONF chunks, which are used to dynamically add or delete IP addresses in SCTP, are not supported.
- The Hostname parameter in INIT and INIT-ACK SCTP messages, which is used to specify a hostname which can then be resolved to an IP address, is not supported.
- SCTP/M3UA does not support equal-cost multipath routing (ECMP), whether configured on the ASA or elsewhere in the network. With ECMP, packets can be routed to a destination over multiple best paths. However, an SCTP/M3UA packet response to a single destination has to come back on the same interface that it exited. Even though the response can come from any M3UA server, it must always come back on the same interface that it exited. The symptom for this problem is that SCTP INIT-ACK packets are dropped, which you can see in the **show asp drop flow sctp-chunk-init-timeout** counter:

  ```
  Flow drop:
  SCTP INIT timed out (not receiving INIT ACK)(sctp-chunk-init-timeout)
  ```

  If you encounter this problem, you can resolve it by configuring static routes to the M3UA servers, or by configuring policy-based routing to implement a network design that ensures that INIT-ACK packets go through the same interface as the INIT packets.

Diameter Inspection

Diameter is an Authentication, Authorization, and Accounting (AAA) protocol used in next-generation mobile and fixed telecom networks such as EPS (Evolved Packet System) for LTE (Long Term Evolution) and IMS (IP Multimedia Subsystem). It replaces RADIUS and TACACS in these networks.

Diameter uses TCP and SCTP as the transport layer, and secures communications using TCP/TLS and SCTP/DTLS. It can optionally provide data object encryption as well. For detailed information on Diameter, see RFC 6733.

Diameter applications perform service management tasks such as deciding user access, service authorization, quality of service, and rate of charging. Although Diameter applications can appear on many different control-plane interfaces in the LTE architecture, the ASA inspects Diameter command codes and attribute-value pairs (AVP) for the following interfaces only:

- S6a: Mobility Management Entity (MME) - Home Subscription Service (HSS).
- S9: PDN Gateway (PDG) - 3GPP AAA Proxy/Server.
- Rx: Policy Charging Rules Function (PCRF) - Call Session Control Function (CSCF).

Diameter inspection opens pinholes for Diameter endpoints to allow communication. The inspection supports 3GPP version 12 and is RFC 6733 compliant. You can use it for TCP/TLS (by specifying a TLS proxy when you enable inspection) and SCTP, but not SCTP/DTLS. Use IPsec to provide security to SCTP Diameter sessions.

You can optionally use a Diameter inspection policy map to filter traffic based on application ID, command codes, and AVP, to apply special actions such as dropping packets or connections, or logging them. You can create custom AVP for newly-registered Diameter applications. Filtering lets you fine-tune the traffic you allow on your network.

Note Diameter messages for applications that run on other interfaces will be allowed and passed through by default. However, you can configure a Diameter inspection policy map to drop these applications by application ID, although you cannot specify actions based on the command codes or AVP for these unsupported applications.

M3UA Inspection

MTP3 User Adaptation (M3UA) is a client/server protocol that provides a gateway to the SS7 network for IP-based applications that interface with the SS7 Message Transfer Part 3 (MTP3) layer. M3UA makes it possible to run the SS7 User Parts (such as ISUP) over an IP network. M3UA is defined in RFC 4666.

M3UA uses SCTP as the transport layer. SCTP port 2905 is the default port.

The MTP3 layer provides networking functions such as routing and node addressing, but uses point codes to identify nodes. The M3UA layer exchanges Originating Point Codes (OPC) and Destination Point Codes (DPC). This is similar to how IP uses IP addresses to identify nodes.

M3UA inspection provides limited protocol conformance. You can optionally implement strict application server process (ASP) state checking and additional message validation for select messages. Strict ASP state checking is required if you want stateful failover or if you want to operate within a cluster. However, strict ASP state checking works in Override mode only, it does not work if you are running in Loadsharing or Broadcast mode (per RFC 4666). The inspection assumes there is one and only one ASP per endpoint.

You can optionally apply access policy based on point codes or Service Indicators (SI). You can also apply rate limiting based on message class and type.

M3UA Protocol Conformance

M3UA inspection provides the following limited protocol enforcement. Inspection drops and logs packets that do not meet requirements.

- Common message header. Inspection validates all fields in the common header.
  - Version 1 only.
  - Message length must be correct.
  - Message type class with a reserved value is not allowed.
  - Invalid message ID within the message class is not allowed.
- Payload data message.
  - Only one parameter of a given type is allowed.

- Data messages on SCTP stream 0 are not allowed.

- The Affected Point Code field must be present in the following messages or the message is dropped: Destination Available (DAVA), Destination Unavailable (DUNA), Destination State Audit (DAUD), Signaling Congestion (SCON), Destination User Part Unavailable (DUPU), Destination Restricted (DRST).
- If you enable message tag validation for the following messages, the content of certain fields are checked and validated. Messages that fail validation are dropped.
  - Destination User Part Unavailable (DUPU)—The User/Cause field must be present, and it must contain only valid cause and user codes.
  - Error—All mandatory fields must be present and contain only allowed values. Each error message must contain the required fields for that error code.
  - Notify—The status type and status information fields must contain allowed values only.
- If you enable strict application server process (ASP) state validation, the system maintains the ASP states of M3UA sessions and allows or drops ASP messages based on the validation result. If you do not enable strict ASP state validation, all ASP messages are forwarded uninspected.

M3UA Inspection Limitations

Following are some limitations on M3UA inspection.

- NAT is not supported for IP addresses that are embedded in M3UA data.
- M3UA strict application server process (ASP) state validation depends on SCTP stateful inspection. Do not implement SCTP state bypass and M3UA strict ASP validation on the same traffic.
- Strict ASP state checking is required if you want stateful failover or if you want to operate within a cluster. However, strict ASP state checking works in Override mode only, it does not work if you are running in Loadsharing or Broadcast mode (per RFC 4666). The inspection assumes there is one and only one ASP per endpoint.

RADIUS Accounting Inspection Overview

The purpose of RADIUS accounting inspection is to prevent over-billing attacks on GPRS networks that use RADIUS servers. Although you do not need the Carrier license to implement RADIUS accounting inspection, it has no purpose unless you are implementing GTP inspection and you have a GPRS setup.

The over-billing attack in GPRS networks results in consumers being billed for services that they have not used. In this case, a malicious attacker sets up a connection to a server and obtains an IP address from the SGSN. When the attacker ends the call, the malicious server will still send packets to it, which gets dropped by the GGSN, but the connection from the server remains active. The IP address assigned to the malicious attacker gets released and reassigned to a legitimate user who will then get billed for services that the attacker will use.

RADIUS accounting inspection prevents this type of attack by ensuring the traffic seen by the GGSN is legitimate. With the RADIUS accounting feature properly configured, the ASA tears down a connection based on matching the Framed IP attribute in the Radius Accounting Request Start message with the Radius

Accounting Request Stop message. When the Stop message is seen with the matching IP address in the Framed IP attribute, the ASA looks for all connections with the source matching the IP address.

You have the option to configure a secret pre-shared key with the RADIUS server so the ASA can validate the message. If the shared secret is not configured, the ASA will only check that the source IP address is one of the configured addresses allowed to send the RADIUS messages.



Note When using RADIUS accounting inspection with GPRS enabled, the ASA checks for the 3GPP-Session-Stop-Indicator in the Accounting Request STOP messages to properly handle secondary PDP contexts. Specifically, the ASA requires that the Accounting Request STOP messages include the 3GPP-SGSN-Address attribute before it will terminate the user sessions and all associated connections. Some third-party GGSNs might not send this attribute by default.

Licensing for Mobile Network Protocol Inspection

Inspection of the following protocols requires the license listed in the table below.

- GTP
- SCTP.
- Diameter
- M3UA

Model	License Requirement
ASA Virtual (all models)	Carrier license (enabled by default)
Secure Firewall 3100	Carrier license
Firepower 4100	Carrier license
Firepower 9300	Carrier license
All other models	The Carrier license is not available on other models. You cannot inspect these protocols.

Defaults for GTP Inspection

GTP inspection is not enabled by default. However, if you enable it without specifying your own inspection map, a default map is used that provides the following processing. You need to configure a map only if you want different values.

- Errors are not permitted.
- The maximum number of requests is 200.
- The maximum number of tunnels is 500. This is equivalent to the number of PDP contexts (endpoints).

- The GTP endpoint timeout is 30 minutes. Endpoints include GSNs (GTPv0,1) and SGW/PGW (GTPv2).
- The PDP context timeout is 30 minutes. In GTPv2, this is the bearer context timeout.
- The request timeout is 1 minute.
- The signaling timeout is 30 minutes.
- The tunneling timeout is 1 hour.
- The T3 response timeout is 20 seconds.
- Unknown message IDs are allowed. You can configure **match message v1/v2 id range** commands to drop and log any commands that you do not support or want to allow. Messages are considered unknown if they are either undefined or are defined in GTP releases that the system does not support.

Configure Mobile Network Inspection

Inspections for protocols used in mobile networks are not enabled by default. You must configure them if you want to support mobile networks.

Procedure

Step 1 (Optional.) Configure a GTP Inspection Policy Map, on page 337.

Step 2 (Optional.) Configure an SCTP Inspection Policy Map, on page 342.

Step 3 (Optional.) Configure a Diameter Inspection Policy Map, on page 343.

If you want to filter on attribute-value pairs (AVP) that are not yet supported in the software, you can create custom AVP for use in the Diameter inspection policy map. See Create a Custom Diameter Attribute-Value Pair (AVP), on page 347.

Step 4 (Optional.) If you want to inspect encrypted Diameter TCP/TLS traffic, create the required TLS proxy as described in Inspecting Encrypted Diameter Sessions, on page 348

Step 5 (Optional.) Configure an M3UA Inspection Policy Map, on page 359

Step 6 Configure the Mobile Network Inspection Service Policy , on page 363.

Step 7 (Optional.) Configure RADIUS Accounting Inspection, on page 364.

RADIUS accounting inspection protects against over-billing attacks.

Configure a GTP Inspection Policy Map

If you want to enforce additional parameters on GTP traffic, and the default map does not meet your needs, create and configure a GTP map.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 Create a GTP inspection policy map: **policy-map type inspect gtp** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **apn regex** {*regex_name* | **class** *class_name*}—Matches the access point name (APN) against the specified regular expression or regular expression class.
- **match** [**not**] **message** {**v1** | **v2**} **id** {*message_id* | **range** *message_id_1 message_id_2*}—Matches the message ID, which can be 1 to 255. You can specify a single ID or a range of IDs. You must specify whether the message is for GTPv0/1 (**v1**) or GTPv2 (**v2**).
- **match** [**not**] **message length min** *bytes* **max** *bytes*—Matches messages where the length of the UDP payload (GTP header plus the rest of the message) is between the minimum and maximum values, from 1 to 65536.
- **match** [**not**] **msisdn regex** {*regex_name* | **class** *class_name*}—Matches the Mobile Station International Subscriber Directory Number (MSISDN) information element in the Create PDP Context request, Create session request, and Modify Bearer Response messages against the specified regular expression or regular expression class. The regular expression can identify a specific MSISDN, or a range of MSISDNs based on the first x number of digits. MSISDN filtering is supported for GTPv1 and GTPv2 only.
- **match** [**not**] **selection-mode** *mode_value*—Matches the Selection Mode information element in the Create PDP Context request. The selection mode specifies the origin of the Access Point Name (APN) in the message, and can be one of the following. Selection Mode filtering is supported for GTPv1 and GTPv2 only.
  - 0—Verified. The APN was provided by the mobile station or network, and the subscription is verified.
  - 1—Mobile Station. The APN was provided by the mobile station, and the subscription is not verified.
  - 2—Network. The APN was provided by the network, and the subscription is not verified.
  - 3—Reserved, not used.
- **match** [**not**] **version** {*version_id* | **range** *version_id_1 version_id_2*}—Matches the GTP version, which can be 0 to 255. You can specify a single version or a range of versions.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop** [**log**]—Drop all packets that match. Add the **log** keyword to also send a system log message.
- **rate-limit** *message_rate*—Limit the rate of messages. This option is available with **message id** only.

You can specify multiple **match** commands in the policy map. For information about the order of **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

Step 4 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **anti-replay** [*window_size*]—Enable anti-replay by specifying a sliding window for GTP-U messages. The size of the sliding window is in number of messages and can be 128, 256, 512, or 1024. If you do not specify a size, you get the default, 512. As valid messages appear, the window moves to the new sequence numbers. Sequence numbers are in the range 0-65535, wrapping when they reach the maximum, and they are unique per PDP context. Messages are considered valid if their sequence numbers are within the window. Anti-replay helps prevent session hijacking or DoS attacks, which can occur when a hacker captures GTP data packets and replays them.
- **permit errors**—Allows all packets, including invalid GTP packets or packets that otherwise would fail parsing and be dropped. Packets can still be dropped based on the actions you define in the policy map.
- **request-queue** *max_requests*—Sets the maximum number of GTP requests that will be queued waiting for a response. The default is 200. When the limit has been reached and a new request arrives, the request that has been in the queue for the longest time is removed. The Error Indication, the Version Not Supported and the SGSN Context Acknowledge messages are not considered as requests and do not enter the request queue to wait for a response.
- **tunnel-limit** *max_tunnels*—Sets the maximum number of active GTP tunnels allowed. This is equivalent to the number of PDP contexts or endpoints. The default is 500. New requests will be dropped once the number of tunnels specified by this command is reached.
- **timeout** {**endpoint** | **pdp-context** | **request** | **signaling** | **t3-response** | **tunnel**} *time*—Sets the idle timeout for the specified service (in hh:mm:ss format). To have no timeout, specify 0 for the number. Enter the command separately for each timeout.
  - **endpoint**—The maximum period of inactivity before a GTP endpoint is removed.
  - **pdp-context**—The maximum period of inactivity before removing the PDP Context for a GTP session. In GTPv2, this is the bearer context.
  - **request**—The maximum period of inactivity after which a request is removed from the request queue. Any subsequent responses to a dropped request will also be dropped.
  - **signaling**—The maximum period of inactivity before GTP signaling is removed.
  - **t3-response**—The maximum wait time for a response before removing the connection.
  - **tunnel**—The maximum period of inactivity for the GTP tunnel before it is torn down.

Step 5 While still in parameter configuration mode, configure GTP-U checking for IP packets and anti-spoofing.

gtp-u-header-check [**anti-spoofing** [**gtpv2-dhcp-bypass** | **gtpv2-dhcp-drop**]]

Without keywords, this command checks whether the inner payload of a GTP data packet is a valid IP packet, and drops the packet if it has a non-IP header.

If you include the **anti-spoofing** keyword, the system also checks whether the mobile user IP address in the IP header of the inner payload matches the IP address assigned in GTP control messages such as Create Session Response, and drops the GTP-U message if the IP addresses do not match. This check supports IPv4, IPv6, and IPv4v6 PDN Types. If the mobile station gets its address using DHCP, the end-user IP address in GTPv2 is 0.0.0.0 (IPv4) or *prefix*::0 (IPv6), so in this case, the system updates the end-user IP address with the first IP address found in the inner packets. You can change the default behavior for DHCP-obtained addresses using the following keywords:

- **gtpv2-dhcp-bypass**—Do not update the 0.0.0.0 or *prefix*::0 address. Instead, allow packets where the end-user IP address is 0.0.0.0 or *prefix*::0. This option bypasses the anti-spoofing check when DHCP is used to obtain the IP address.
- **gtpv2-dhcp-drop**—Do not update the 0.0.0.0 or *prefix*::0 address. Instead, drop all packets where the end-user IP address is 0.0.0.0 or *prefix*::0. This option prevents access for users that use DHCP to obtain the IP address.

Step 6 While still in parameter configuration mode, configure IMSI prefix filtering, if desired:

mcc *country_code* **mnc** *network_code*

drop mcc *country_code* **mnc** *network_code*

You can enter the command as many times as necessary to specify all targeted MCC/MNC pairs, but all commands within the policy map must be either **mcc** or **drop mcc**. You cannot combine these commands.

By default, GTP inspection does not check for valid Mobile Country Code (MCC)/Mobile Network Code (MNC) combinations. If you configure IMSI prefix filtering, the MCC and MNC in the IMSI of the received packet is compared with the configured MCC/MNC combinations. The system then takes one of the following actions based on the command:

- **mcc** command—The packet is dropped if it does not match.
- **drop mcc** command—The packet is dropped if it does match.

The Mobile Country Code is a non-zero, three-digit value; add zeros as a prefix for one- or two-digit values. The Mobile Network Code is a two- or three-digit value.

Add all MCC and MNC combinations you want to either permit or to drop. By default, the ASA does not check the validity of MNC and MCC combinations, so you must verify the validity of the combinations configured. To find more information about MCC and MNC codes, see the ITU E.212 recommendation, *Identification Plan for Land Mobile Stations*.

Step 7 While still in parameters configuration mode, enable location logging, if desired.

location-logging [**cell-id**]

Log the location of subscribers to track location changes for mobile stations. Tracking location changes can help you identify fraudulent roaming charges. When you enable location logging, the system generates syslog messages for new (message 324010) or changed (message 324011) location for each International Mobile Subscriber Identity (IMSI).

Specify the **cell-id** parameter if you want the log messages to include the the cell ID where the user currently is registered. The cell ID is extracted from the Cell Global Identification (CGI) or E-UTRAN Cell Global Identifier (ECGI).

Step 8 While still in parameter configuration mode, configure GSN or PGW pooling, if desired.

permit-response to-object-group *SGSN-SGW_name* **from-object-group** *GSN-PGW_pool*

When the ASA performs GTP inspection, by default the ASA drops GTP responses from GSNs or PGWs that were not specified in the GTP request. This situation occurs when you use load-balancing among a pool of GSNs or PGWs to provide efficiency and scalability of GPRS.

To configure GSN/PGW pooling and thus support load balancing, create a network object group that specifies the GSN/PGW endpoints and specify this on the **from-object-group** parameter. Likewise, create a network object group for the SGSN/SGW and select it on the **to-object-group** parameter. If the GSN/PGW responding belongs to the same object group as the GSN/PGW that the GTP request was sent to and if the SGSN/SGW is in an object group that the responding GSN/PGW is permitted to send a GTP response to, the ASA permits the response.

The network object group can identify the endpoints by host address or by the subnet that contains them.

Example:

The following is an example of GSN/PGW pooling. An entire Class C network is defined as the GSN/PGW pool but you can identify multiple individual IP addresses, one per **network-object** command, instead of identifying whole networks. The example then modifies a GTP inspection map to permit responses from the pool to the SGSN/SgW.

```
hostname(config)# object-group network gsnpool32
hostname(config-network)# network-object 192.168.100.0 255.255.255.0
hostname(config)# object-group network sgsn32
hostname(config-network)# network-object host 192.168.50.100

hostname(config)# policy-map type inspect gtp gtp-policy
hostname(config-pmap)# parameters
hostname(config-pmap-p)# permit-response to-object-group sgsn32
from-object-group gsnpool32
```

Example

The following example shows how to limit the number of tunnels in the network:

```
hostname(config)# policy-map type inspect gtp gmap
hostname(config-pmap)# parameters
hostname(config-pmap-p)# tunnel-limit 3000

hostname(config)# policy-map global_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect gtp gmap

hostname(config)# service-policy global_policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure the Mobile Network Inspection Service Policy , on page 363.

Configure an SCTP Inspection Policy Map

To apply alternative actions to SCTP traffic based on the application-specific payload protocol identifier (PPID), such as rate limiting, create an SCTP inspection policy map to be used by the service policy.



Note PPIDs are in data chunks, and a given packet can have multiple data chunks or even a control chunk. If a packet includes a control chunk or multiple data chunks, the packet will not be dropped even if the assigned action is drop. For example, if you configure an SCTP inspection policy map to drop PPID 26, and a PPID 26 data chunk is combined in a packet with a Diameter PPID data chunk, that packet will not be dropped.

Procedure

Step 1 Create an SCTP inspection policy map: **policy-map type inspect sctp** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 Drop, rate limit, or log traffic based on the PPID in SCTP data chunks.

a) Identify the traffic based on the PPID.

match [**not**] **ppid** *ppid_1* [*ppid_2*]

Where *ppid_1* is the PPID number (0-4294967295) or name (see the CLI help for the available names). You can include a second (higher) PPID, *ppid_2*, to specify a range of PPIDs. Use **match not ppid** to identify traffic that does not match the PPID or range.

You can find the current list of SCTP PPIDs at http://www.iana.org/assignments/sctp-parameters/sctp-parameters.xhtml#sctp-parameters-25.

b) Specify the action to perform on matching packets.

- **drop**—Drop and log all packets that match.
- **log**—Send a system log message.
- **rate-limit** *rate*—Limit the rate of messages. The rate is in kilobits per second (kbps).

c) Repeat the process until you identify all PPIDs you want to selectively handle.

Example

The following example creates an inspection policy map that will drop unassigned PPIDs (unassigned at the time this example was written), rate limit PPIDs 32-40, and log the Diameter PPID. The service policy applies the inspection to the inspection_default class, which matches all SCTP traffic.

```
policy-map type inspect sctp sctp-pmap
 match ppid 58 4294967295
  drop
 match ppid 26
  drop
 match ppid 49
  drop
 match ppid 32 40
  rate-limit 1000
 match ppid diameter
  log

policy-map global_policy
 class inspection_default
  inspect sctp sctp-pmap
!
service-policy global_policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure the Mobile Network Inspection Service Policy , on page 363.

Configure a Diameter Inspection Policy Map

You can create a Diameter inspection policy map to filter on various Diameter protocol elements. You can then selectively drop or log connections.

To configure Diameter message filtering, you must have a good knowledge of these protocol elements as they are defined in RFCs and technical specifications. For example, the IETF has a list of registered applications, command codes, and attribute-value pairs at http://www.iana.org/assignments/aaa-parameters/aaa-parameters.xhtml, although Diameter inspection does not support all listed items. See the 3GPP web site for their technical specifications.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create a Diameter inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect diameter** [**match-all** | **match-any**] *class_map_name*

   Where *class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one **match** statement. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

   Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

   - **match** [**not**] **application-id** *app_id* [*app_id_2*]—Matches the application identifier, where *app_id* is the Diameter application name or number (0-4294967295). If there is a range of consecutively-numbered applications that you want to match, you can include a second ID. You can define the range by application name or number, and it applies to all the numbers between the first and second IDs.

     These applications are registered with the IANA. Following are the core supported applications, but you can filter on other applications. Use the CLI help for a list of application names.

     - **3gpp-rx-ts29214** (16777236)
     - **3gpp-s6a** (16777251)
     - **3gpp-s9** (16777267)
     - **common-message** (0). This is the base Diameter protocol.

   - **match** [**not**] **command-code** *code* [*code_2*]—Matches the command code, where *code* is the Diameter command code name or number (0-4294967295). If there is a range of consecutively-numbered command codes that you want to match, you can include a second code. You can define the range by command code name or number, and it applies to all the numbers between the first and second codes.

     For example, the following command matches the Capability Exchange Request/Answer command code:

     ```
     match command-code cer-cea
     ```

- Match the attribute-value pair (AVP).

  To match AVP by attribute only:

  match [**not**] **avp** *code* [*code_2*] [**vendor-id** *id_number*]

  To match an AVP based on the value of the attribute:

  match [**not**] **avp** *code* [**vendor-id** *id_number*] *value*

  Where:

  - *code*—The name or number (1-4294967295) of an attribute-value pair. For the first code, you can specify the name of a custom AVP or one that is registered in RFCs or 3GPP technical specifications and is directly supported in the software. If you want to match a range of AVP, specify the second code by number only. If you want to match an AVP by its value, you cannot specify a second code. See the CLI help for a list of AVP names.
  - **vendor-id** *id_number*—(Optional.) The ID number of the vendor to also match, from 0-4294967295. For example, the 3GPP vendor ID is 10415, the IETF is 0.
  - *value*—The value portion of the AVP. You can configure this only if the data type of the AVP is supported. For example, you can specify an IP address for AVP that have the address data type. Following are the specific syntax of the value option for the supported data types:
    - Diameter Identity, Diameter URI, Octet String—Use regular expression or regular expression class objects to match these data types.

      {**regex** *regex_name* | **class** *regex_class*}

    - Address—Specify the IPv4 or IPv6 address to match. For example, 10.100.10.10 or 2001:DB8::0DB8:800:200C:417A.
    - Time—Specify the start and end dates and time. Both are required. Time is in 24-hour format.

      date *year month day* **time** *hh:mm:ss* **date** *year month day* **time** *hh:mm:ss*

      For example:

      ```
      date 2015 feb 5 time 12:00:00 date 2015 mar 9 time 12:00:00
      ```

    - Numeric—Specify a range of numbers:

      range *number_1 number_2*

      The valid number range depends on the data type:

      - Integer32: -2147483647 to 2147483647
      - Integer64: -9223372036854775807 to 9223372036854775807
      - Unsigned32: 0 to 4294967295
      - Unsigned64: 0 to 18446744073709551615
      - Float32: decimal point representation with 8 digit precision
      - Float64: decimal point representation with 16 digit precision

d) Enter **exit** to leave class map configuration mode.

Step 2 Create a Diameter inspection policy map: **policy-map type inspect diameter** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created a Diameter class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for Diameter class maps.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop**—Drop all packets that match.
- **drop-connection**—Drop the packet and close the connection.
- **log**—Send a system log message.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled, on page 246.

Example:

```
hostname(config)# policy-map type inspect diameter diameter-map
hostname(config-pmap)# class diameter-class-map
hostname(config-pmap-c)# drop
hostname(config-pmap-c)# match command-code cer-cea
hostname(config-pmap-c)# log
```

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the no form of the command to disable the option.

- **unsupported** {**application-id** | **command-code** | **avp**} **action log**—Enables logging for unsupported Diameter elements. These options specify application IDs, command codes, and AVP that are not directly supported by the software. The default is to allow the elements without logging them. You can enter the command three times to enable logging for all elements.
- **strict-diameter** {**state** | **session**}—Enables strict Diameter protocol conformance to RFC 6733. By default, inspection ensures that Diameter frames comply with the RFC. You can add **state** machine validation or **session**-related message validation, or both by entering the command twice.

Example:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)# unsupported application-id action log
hostname(config-pmap-p)# unsupported command-code action log
hostname(config-pmap-p)# unsupported avp action log
hostname(config-pmap-p)# strict-diameter state
hostname(config-pmap-p)# strict-diameter session
```

Example

The following example shows a how to log some applications and block a specific IP address.

```
class-map type inspect diameter match-any log_app
  match application-id 3gpp-s6a
  match application-id 3gpp-s13

class-map type inspect diameter match-all block_ip
  match command-code cer-cea
  match avp host-ip-address 1.1.1.1

policy-map type inspect diameter diameter_map
  parameters
    unsupported application-id log
  class log_app
    log
  class block_ip
    drop-connection

policy-map global_policy
  class inspection_default
    inspect diameter diameter_map

service-policy global_policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure the Mobile Network Inspection Service Policy , on page 363.

Create a Custom Diameter Attribute-Value Pair (AVP)

As new attribute-value pairs (AVP) are defined and registered, you can create custom Diameter AVP to define them and use them in your Diameter inspection policy map. You would get the information you need to create the AVP from the RFC or other source that defines the AVP.

Create custom AVP only if you want to use them in a Diameter inspection policy map or class map for AVP matching.

Procedure

Create a custom Diameter AVP.

diameter avp *name* **code** *value* **data-type** *type* [**vendor-id** *id_number*] [**description** *text*]

Where:

- *name*—The name of the custom AVP you are creating, up to 32 characters. You would refer to this name on the match avp command in a Diameter inspection policy map or class map.
- **code** *value*—The custom AVP code value, from 256-4294967295. You cannot enter a code and vendor-id combination that is already defined in the system.
- **data-type** *type*—The data type of the AVP. You can define AVP of the following types. If the new AVP is of a different type, you cannot create a custom AVP for it.
  - **address**—For IP addresses.
  - **diameter-identity**—Diameter identity data.
  - **diameter-uri**—Diameter uniform resource identifier (URI).
  - **float32**—32-bit floating point number.
  - **float64**—64-bit floating point number.
  - **int32**—32-bit integer.
  - **int64**—64-bit integer.
  - **octetstring**—Octet string.
  - **time**—Time value.
  - **uint32**—32-bit unsigned integer.
  - **uint64**—64-bit unsigned integer.
- **vendor-id** *id_number*—(Optional.) The ID number of the vendor who defined the AVP, from 0-4294967295. For example, the 3GPP vendor ID is 10415, the IETF is 0.
- **description** *text*—(Optional.) A description of the AVP, up to 80 characters. Enclose the description in quotation marks if you include spaces.

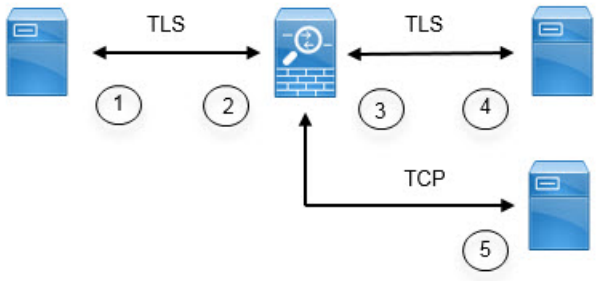
Inspecting Encrypted Diameter Sessions

If a Diameter application uses encrypted data over TCP, inspection cannot see inside the packets to implement your message filtering rules. Thus, if you create filtering rules, and you want them to also apply to encrypted TCP traffic, you must configure a TLS proxy. You also need a proxy if you want strict protocol enforcement on encrypted traffic. This configuration does not apply to SCTP/DTLS traffic.

The TLS proxy acts as a man-in-the-middle. It decrypts traffic, inspects it, then encrypts it again and sends it to the intended destination. Thus, both sides of the connection, the Diameter server and Diameter client, must trust the ASA, and all parties must have the required certificates. You must have a good understanding of digital certificates to implement TLS proxy. Please read the chapter on digital certificates in the ASA general configuration guide.

The following illustration shows the relationship among the Diameter client and server, and the ASA, and the certification requirements to establish trust. In this model, a Diameter client is an MME (Mobility Management Entity), not an end user. The CA certificate on each side of a link is the one used to sign the certificate on the other side of the link. For example, the ASA proxy TLS server CA certificate is the one used to sign the Diameter/TLS client certificate.

Figure 43: Diameter TLS Inspection



1	Diameter TLS client (MME) • Client identity certificate • CA certificate used to sign the ASA TLS proxy server's identity certificate	2	ASA proxy TLS server • Server identity certificate • CA certificate used to sign the Diameter TLS client's identity certificate
3	ASA proxy TLS client • Client identity (static or LDC) certificate • CA certificate used to sign the Diameter TLS server identity certificate	4	Diameter TLS server (full proxy) • Server identity certificate • CA certificate used to sign the ASA proxy TLS client's identity certificate
5	Diameter TCP server (TLS offload).	—	—

You have the following options for configuring TLS proxy for Diameter inspection:

- Full TLS proxy—Encrypt traffic between the ASA and Diameter clients and the ASA and Diameter server. You have the following options for establishing the trust relationship with the TLS server:
  - Use a static proxy client trustpoint. The ASA presents the same certificate for every Diameter client when communicating with the Diameter server. Because all clients look the same, the Diameter server cannot provide differential services per client. On the other hand, this option is faster than the LDC method.
  - Use local dynamic certificates (LDC). With this option, the ASA presents unique certificates per Diameter client when communicating with the Diameter server. The LDC retains all fields from the

received client identity certificate except its public key and a new signature from the ASA. This method gives the Diameter server better visibility into client traffic, which makes it possible to provide differential services based on client certificate characteristics.

- TLS offload—Encrypt traffic between the ASA and Diameter client, but use a clear-text connection between the ASA and Diameter server. This option is viable if the Diameter server is in the same data center as the ASA, where you are certain that the traffic between the devices will not leave the protected area. Using TLS offload can improve performance, because it reduces the amount of encryption processing required. It should be the fastest of the options. The Diameter server can apply differential services based on client IP address only.

All three options use the same configuration for the trust relationship between the ASA and Diameter clients.

Note TLS proxy uses TLSv1.0 - 1.2. You can configure the TLS version and the cipher suite.

The following topics explain how to configure TLS proxy for Diameter inspection.

Configure Server Trust Relationship with Diameter Clients

The ASA acts as a TLS proxy server in relation to the Diameter clients. To establish the mutual trust relationship:

- You need to import the Certificate Authority (CA) certificate used to sign the ASA's server certificate into the Diameter client. This might be in the client's CA certificate store or some other location that the client uses. Consult the client documentation for exact details on certificate usage.
- You need to import the CA certificate used to sign the Diameter TLS client's certificate so the ASA can trust the client.

The following procedure explains how to import the CA certificate used to sign the Diameter client's certificate, and import an identity certificate to use for the ASA TLS proxy server. Instead of importing an identity certificate, you could create a self-signed certificate on the ASA.

Procedure

Step 1 Import the CA certificate that is used to sign the Diameter client's certificate into an ASA trustpoint.

This step allows the ASA to trust the Diameter clients.

a) Create the trustpoint for the Diameter client.

In this example, **enrollment terminal** indicates you will paste the certificate into the CLI. The trustpoint is called **diameter-clients**.

```
ciscoasa(config)# crypto ca trustpoint diameter-clients
ciscoasa(ca-trustpoint)# revocation-check none
ciscoasa(ca-trustpoint)# enrollment terminal
```

b) Add the certificate.

```
ciscoasa(config)# crypto ca authenticate diameter-clients
Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself
MIIDRTCCAu+gAwIBAgIQKVcqP/KW74VP0NZzL+JbRTANBgkqhkiG9w0BAQUFADCB
[certificate data omitted]
/7QEM8izy0EOTSErKu7Nd76jwf5e4qttkQ==
quit

INFO: Certificate has the following attributes:
Fingerprint: 24b81433 409b3fd5 e5431699 8d490d34
Do you accept this certificate? [yes/no]: y
Trustpoint CA certificate accepted.

% Certificate successfully imported
```

Step 2 Import the certificate and create a trustpoint for the ASA proxy server's identity certificate and keypair.

This step allows the Diameter clients to trust the ASA.

a) Import the certificate in pkcs12 format.

In the following example, **tls-proxy-server-tp** is the trustpoint name and **"123"** is the decryption pass phrase. Use your own trustpoint name and pass phrase.

```
ciscoasa (config)# crypto ca import tls-proxy-server-tp pkcs12 "123"

Enter the base 64 encoded pkcs12.
End with a blank line or the word "quit" on a line by itself:
[PKCS12 data omitted]

quit

INFO: Import PKCS12 operation completed successfully

ciscoasa (config)#
```

b) Configure the trustpoint.

```
ciscoasa(config)# crypto ca trustpoint tls-proxy-server-tp
ciscoasa(ca-trustpoint)# revocation-check none
```

Configure Full TLS Proxy with Static Client Certificate for Diameter Inspection

If the Diameter server can accept the same certificate for all clients, you can set up a static client certificate for the ASA to use when communicating with the Diameter server.

With this configuration, you need to establish the mutual trust relationship between the ASA and clients (as explained in Configure Server Trust Relationship with Diameter Clients, on page 350), and the ASA and Diameter server. Following are the ASA and Diameter server trust requirements.

- You need to import the CA certificate used to sign the Diameter Server's identity certificate so the ASA can validate the server's identity certificate during the TLS handshake.

- You need to import the client certificate, one that the Diameter server also trusts. If the Diameter server does not already trust the certificate, import the CA certificate used to sign it into the server. Consult the Diameter server's documentation for details.

Procedure

Step 1 Import the CA certificate that is used to sign the Diameter server's certificate into an ASA trustpoint.

This step allows the ASA to trust the Diameter server.

a) Create the trustpoint for the Diameter server.

In this example, **enrollment terminal** indicates you will paste the certificate into the CLI. You could also use enrollment url to specify automatic enrollment (SCEP) with the CA. The trustpoint is called **diameter-server**.

```
ciscoasa(config)# crypto ca trustpoint diameter-server
ciscoasa(ca-trustpoint)# revocation-check none
ciscoasa(ca-trustpoint)# enrollment terminal
```

b) Add the certificate.

```
ciscoasa(config)# crypto ca authenticate diameter-server
Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself
MIIDRTCCAu+gAwIBAgIQKVcqP/KW74VP0NZzL+JbRTANBgkqhkiG9w0BAQUFADCB
[certificate data omitted]
/7QEM8izy0EOTSErKu7Nd76jwf5e4qttkQ==
quit

INFO: Certificate has the following attributes:
Fingerprint: 24b81433 409b3fd5 e5431699 8d490d34
Do you accept this certificate? [yes/no]: y
Trustpoint CA certificate accepted.

% Certificate successfully imported
```

Step 2 Import the certificate and create a trustpoint for the ASA proxy client's identity certificate and keypair.

This step allows the Diameter server to trust the ASA.

a) Import the certificate in pkcs12 format.

In the following example, **tls-proxy-client-tp** is the trustpoint name and **"123"** is the decryption pass phrase. Use your own trustpoint name and pass phrase.

```
ciscoasa (config)# crypto ca import tls-proxy-client-tp pkcs12 "123"

Enter the base 64 encoded pkcs12.
End with a blank line or the word "quit" on a line by itself:
[PKCS12 data omitted]

quit
```

```
INFO: Import PKCS12 operation completed successfully

ciscoasa (config)#
```

b) Configure the trustpoint.

```
ciscoasa(config)# crypto ca trustpoint tls-proxy-client-tp
ciscoasa(ca-trustpoint)# revocation-check none
```

Step 3 Configure the TLS proxy.

a) Name the TLS proxy and enter TLS proxy configuration mode.

tls-proxy *name*

b) Identify the trustpoint used when the ASA acts as the proxy server in relationship to the Diameter clients.

server trust-point *trustpoint_name*

Note
For testing purposes, or if you are certain that you can trust the Diameter clients, you can skip this step and include the **no server authenticate-client** command in the TLS proxy configuration.

c) Identify the trustpoint used when the ASA acts as the proxy client in relationship to the Diameter server.

client trust-point *name*

d) (Optional.) Define the ciphers that the client can use.

client cipher-suite *cipher-list*

Where *cipher-list* can include any combination of the following:

- **3des-sha1**
- **aes128-sha1**
- **aes256-sha1**
- **des-sha1**
- **null-sha1**
- **rc4-sha1**

Separate multiple options with spaces.

If you do not define the ciphers the TLS proxy can use, the proxy server uses the global cipher suite defined by the **ssl cipher** command. By default, the global cipher level is medium, which means all ciphers are available except for NULL-SHA, DES-CBC-SHA, and RC4-MD5. Specify the **client cipher-suite** command only if you want to use a different suite than the one generally available on the ASA.

To set the minimum TLS version for all SSL client connections on the ASA, see the **ssl client-version** command. The default is TLS v1.0.

e) (Optional.) Define the ciphers that the server can use.

server cipher-suite *cipher-list*

Where *cipher-list* can include any combination of the following:

- **3des-sha1**
- **aes128-sha1**
- **aes256-sha1**
- **des-sha1**
- **null-sha1**
- **rc4-sha1**

Separate multiple options with spaces.

If you do not define the ciphers the TLS proxy can use, the proxy server uses the global cipher suite defined by the **ssl cipher** command. By default, the global cipher level is medium, which means all ciphers are available except for NULL-SHA, DES-CBC-SHA, and RC4-MD5. Specify the **server cipher-suite** command only if you want to use a different suite than the one generally available on the ASA.

To set the minimum TLS version for all SSL server connections on the ASA, see the **ssl server-version** command. The default is TLS v1.0.

Example:

```
ciscoasa(config)# tls-proxy diameter-tls-static-proxy
ciscoasa(config-tlsp)# server trust-point tls-proxy-server-tp
ciscoasa(config-tlsp)# client trust-point tls-proxy-client-tp
```

What to do next

You can now use the TLS proxy in Diameter inspection. See Configure the Mobile Network Inspection Service Policy , on page 363.

Configure Full TLS Proxy with Local Dynamic Certificates for Diameter Inspection

If the Diameter server needs unique certificates for each client, you can configure the ASA to generate local dynamic certificates (LDC). These certificates exist for the duration of the client's connection and are then destroyed.

With this configuration, you need to establish the mutual trust relationship between the ASA and clients (as explained in Configure Server Trust Relationship with Diameter Clients, on page 350), and the ASA and Diameter server. The configuration is similar to the one described in Configure Full TLS Proxy with Static Client Certificate for Diameter Inspection, on page 351, except instead of importing a Diameter client certificate, you set up the LDC on the ASA. Following are the ASA and Diameter server trust requirements.

- You need to import the CA certificate used to sign the Diameter Server's identity certificate so the ASA can validate the server's identity certificate during the TLS handshake.
- You need to create the LDC trustpoint. You need to export the LDC server's CA certificate and import it into the Diameter server. The export step is explained below. Consult the Diameter server's documentation for information on importing certificates.

Procedure

Step 1 Import the CA certificate that is used to sign the Diameter server's certificate into an ASA trustpoint.

This step allows the ASA to trust the Diameter server.

a) Create the trustpoint for the Diameter server.

In this example, **enrollment terminal** indicates you will paste the certificate into the CLI. You could also use enrollment url to specify automatic enrollment (SCEP) with the CA. The trustpoint is called **diameter-server**.

```
ciscoasa(config)# crypto ca trustpoint diameter-server
ciscoasa(ca-trustpoint)# revocation-check none
ciscoasa(ca-trustpoint)# enrollment terminal
```

b) Add the certificate.

```
ciscoasa(config)# crypto ca authenticate diameter-server
Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself
MIIDRTCCAu+gAwIBAgIQKVcqP/KW74VP0NZzL+JbRTANBgkqhkiG9w0BAQUFADCB
[certificate data omitted]
/7QEM8izy0EOTSErKu7Nd76jwf5e4qttkQ==
quit

INFO: Certificate has the following attributes:
Fingerprint: 24b81433 409b3fd5 e5431699 8d490d34
Do you accept this certificate? [yes/no]: y
Trustpoint CA certificate accepted.

% Certificate successfully imported
```

Step 2 Create the local CA to sign local dynamic certificates (LDC).

a) Create an RSA keypair for the trustpoint.

In this example, the keypair name is ldc-signer-key.

```
ciscoasa(config)# crypto key generate rsa label ldc-signer-key
INFO: The name for the keys will be: ldc-signer-key
Keypair generation process
ciscoasa(config)#
```

b) Create the LDC issuer trustpoint.

In this example, the trustpoint name is **ldc-server**, the keypair created above is used, self-signed enrollment is specified (**enrollment self**, which is required), and the common name of the ASA is included as the subject name. Check whether the Diameter application has specific requirements for the subject name.

The **proxy-ldc-issuer** command defines the local CA role for the trustpoint to issue dynamic certificates for TLS proxy.

```
ciscoasa(config)# crypto ca trustpoint ldc-server
ciscoasa(ca-trustpoint)# keypair ldc-signer-key
ciscoasa(ca-trustpoint)# subject-name CN=asa3
ciscoasa(ca-trustpoint)# enrollment self
ciscoasa(ca-trustpoint)# proxy-ldc-issuer
ciscoasa(ca-trustpoint)# exit
```

c) Enroll the trustpoint.

```
ciscoasa(config)# crypto ca enroll ldc-server
```

Step 3 Configure the TLS proxy.

a) Name the TLS proxy and enter TLS proxy configuration mode.

tls-proxy *name*

b) Identify the trustpoint used when the ASA acts as the server in relationship to the Diameter clients.

server trust-point *trustpoint_name*

Note
For testing purposes, or if you are certain that you can trust the Diameter clients, you can skip this step and include the **no server authenticate-client** command in the TLS proxy configuration.

c) Identify the LDC trustpoint used when the ASA issues dynamic certificates and acts as the client in relationship to the Diameter server.

client ldc issuer *name*

d) Identify the LDC keypair. Specify the same key that is defined in the LDC trustpoint.

client ldc key-pair *name*

e) (Optional.) Define the ciphers that the client can use.

client cipher-suite *cipher-list*

Where *cipher-list* can include any combination of the following:

- **3des-sha1**
- **aes128-sha1**
- **aes256-sha1**
- **des-sha1**
- **null-sha1**
- **rc4-sha1**

Separate multiple options with spaces.

If you do not define the ciphers the TLS proxy can use, the proxy server uses the global cipher suite defined by the **ssl cipher** command. By default, the global cipher level is medium, which means all ciphers are available except for NULL-SHA, DES-CBC-SHA, and RC4-MD5. Specify the **client cipher-suite** command only if you want to use a different suite than the one generally available on the ASA.

To set the minimum TLS version for all SSL client connections on the ASA, see the **ssl client-version** command. The default is TLS v1.0.

f) (Optional.) Define the ciphers that the server can use.

server cipher-suite *cipher-list*

Where *cipher-list* can include any combination of the following:

- **3des-sha1**
- **aes128-sha1**
- **aes256-sha1**
- **des-sha1**
- **null-sha1**
- **rc4-sha1**

Separate multiple options with spaces.

If you do not define the ciphers the TLS proxy can use, the proxy server uses the global cipher suite defined by the **ssl cipher** command. By default, the global cipher level is medium, which means all ciphers are available except for NULL-SHA, DES-CBC-SHA, and RC4-MD5. Specify the **server cipher-suite** command only if you want to use a different suite than the one generally available on the ASA.

To set the minimum TLS version for all SSL server connections on the ASA, see the **ssl server-version** command. The default is TLS v1.0.

Example:

```
ciscoasa(config)# tls-proxy diameter-tls-ldc-proxy
ciscoasa(config-tlsp)# server trust-point tls-proxy-server-tp
ciscoasa(config-tlsp)# client ldc issuer ldc-server
ciscoasa(config-tlsp)# client ldc key-pair ldc-signer-key
```

Step 4 Export the LDC CA certificate and import it into the Diameter server.

a) Export the certificate.

In the following example, the LDC trustpoint is ldc-server; specify your own LDC trustpoint name.

```
ciscoasa(config)# crypto ca export ldc-server identity-certificate
-----BEGIN CERTIFICATE-----
MIIDbDCCAlSgAwIBAgIQfWOQvGFpj7hCCB49+kS4CjANBgkqhkiG9w0BAQUFADAT
MREwDwYDVQQDEwhIdW5ueUJlZTAeFw0xMzA2MjUwMTE5MzJaFw00ODA2MjUwMTI5
...[data omitted]...
lJZ48NoI64RqfGC/KHUsOQ==
-----END CERTIFICATE-----
```

b) Copy the certificate data and save it into a file.

You can now import it into the Diameter server. Consult the Diameter server's documentation for the procedure. Note that the data is in Base64 format. If your server requires binary or DER format, you will need to use OpenSSL tools to convert formats.

What to do next

You can now use the TLS proxy in Diameter inspection. See Configure the Mobile Network Inspection Service Policy , on page 363.

Configure TLS Proxy with TLS Offload for Diameter Inspection

If you are certain the network path between the ASA and Diameter server is secure, you can avoid the performance cost of encrypting data between the ASA and server. With TLS offload, the TLS proxy encrypts/decrypts sessions between the Diameter client and the ASA, but uses clear text with the Diameter server.

With this configuration, you need to establish the mutual trust relationship between the ASA and clients only, which simplifies the configuration. Before doing the following procedure, complete the steps in Configure Server Trust Relationship with Diameter Clients, on page 350.

Procedure

Step 1 Configure the TLS proxy with TLS offload.

a) Name the TLS proxy and enter TLS proxy configuration mode.

tls-proxy *name*

b) Identify the trustpoint used when the ASA acts as the server in relationship to the Diameter clients.

server trust-point *trustpoint_name*

Note
For testing purposes, or if you are certain that you can trust the Diameter clients, you can skip this step and include the **no server authenticate-client** command in the TLS proxy configuration.

c) (Optional.) Define the ciphers that the server can use.

server cipher-suite *cipher-list*

Where *cipher-list* can include any combination of the following:

- **3des-sha1**
- **aes128-sha1**
- **aes256-sha1**
- **des-sha1**
- **null-sha1**
- **rc4-sha1**

Separate multiple options with spaces.

If you do not define the ciphers the TLS proxy can use, the proxy server uses the global cipher suite defined by the **ssl cipher** command. By default, the global cipher level is medium, which means all ciphers are available except for NULL-SHA, DES-CBC-SHA, and RC4-MD5. Specify the **server cipher-suite** command only if you want to use a different suite than the one generally available on the ASA.

To set the minimum TLS version for all SSL server connections on the ASA, see the **ssl server-version** command. The default is TLS v1.0.

d) Specify that communication between the ASA and the Diameter server should be in clear text. In this relationship, the ASA acts as a client of the Diameter server.

client clear-text

Example:

```
ciscoasa(config)# tls-proxy diameter-tls-offload-proxy
ciscoasa(config-tlsp)# server trust-point tls-proxy-server-tp
ciscoasa(config-tlsp)# client clear-text
```

Step 2 Because the Diameter ports differ for TCP and TLS, configure a NAT rule to translate the TCP port to the TLS port for traffic going from the Diameter server to the client.

Create an object NAT rule for each Diameter server. Each rule should:

- Perform static identity NAT for the Diameter server address. That is, the IP address in the object should be the same as the translated address in the NAT rule.
- Translate the real port 3868, which is the default Diameter TCP port number, to 5868, the default Diameter TLS port number.
- The source interface should be the one that connects to the Diameter server, and the destination interface the one that connects to the Diameter client.

The following example translates TCP traffic on port 3868 coming to the outside interface from the 10.29.29.29 Diameter server to port 5868 on the inside interface.

```
ciscoasa(config)# object network diameter-client
ciscoasa(config-network-object)# host 10.29.29.29
ciscoasa(config-network-object)# nat (outside,inside) static 10.29.29.29
service tcp 3868 5868
```

What to do next

You can now use the TLS proxy in Diameter inspection. See Configure the Mobile Network Inspection Service Policy , on page 363.

Configure an M3UA Inspection Policy Map

Use an M3UA inspection policy map to configure access control based on point codes. You can also drop and rate limit messages by class and type.

The default point code format is ITU. If you use a different format, specify the required format in the policy map.

If you do not want to apply policy based on point code or message class, you do not need to configure an M3UA policy map. You can enable inspection without a map.

Procedure

Step 1 Create an M3UA inspection policy map: **policy-map type inspect m3ua** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **message class** *class_id* [**id** *message_id*]—Matches the M3UA message class and type. The following table lists the possible values. Consult M3UA RFCs and documentation for detailed information about these messages.

M3UA Message Class	Message ID Type
0 (Management Messages)	0-1
1 (Transfer Messages)	1
2 (SS7 Signaling Network Management Messages)	1-6
3 (ASP State Maintenance Messages)	1-6
4 (ASP Traffic Maintenance Messages)	1-4
9 (Routing Key Management Messages)	1-4

- **match** [**not**] **opc** *code*—Matches the originating point code in the data message, that is, the traffic source. Point code is in *zone-region-sp* format, where the possible values for each element depend on the SS7 variant:
  - **ITU**—Point codes are 14 bit in 3-8-3 format. The value ranges are [0-7]-[0-255]-[0-7].
  - **ANSI**—Point codes are 24 bit in 8-8-8 format. The value ranges are [0-255]-[0-255]-[0-255].
  - **Japan**—Point codes are 16 bit in 5-4-7 format. The value ranges are [0-31]-[0-15]-[0-127].
  - **China**—Point codes are 24 bit in 8-8-8 format. The value ranges are [0-255]-[0-255]-[0-255].
- **match** [**not**] **dpc** *code*—Matches the destination point code in the data message. Point code is in *zone-region-sp* format, as explained for **match opc**.

- **match** [**not**] **service-indicator** *number*—Matches the service indicator number, 0-15. Following are the available service indicators. Consult M3UA RFCs and documentation for detailed information about these service indicators.
  - 0—Signaling Network Management Messages
  - 1—Signaling Network Testing and Maintenance Messages
  - 2—Signaling Network Testing and Maintenance Special Messages
  - 3—SCCP
  - 4—Telephone User Part
  - 5—ISDN User Part
  - 6—Data User Part (call and circuit-related messages)
  - 7—Data User Part (facility registration and cancellation messages)
  - 8—Reserved for MTP Testing User Part
  - 9—Broadband ISDN User Part
  - 10—Satellite ISDN User Part
  - 11—Reserved
  - 12—AAL type 2 Signaling
  - 13—Bearer Independent Call Control
  - 14—Gateway Control Protocol
  - 15—Reserved

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop** [**log**]—Drop all packets that match. Optionally, send a system log message.
- **rate-limit** *message_rate*—Limit the rate of messages. This option is available with **match message class** only.

You can specify multiple **match** commands in the policy map. For information about the order of match commands, see How Multiple Traffic Classes are Handled, on page 246.

Step 4 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **message-tag-validation** {**dupu** | **error** | **notify**}—Ensures that the content of certain fields are checked and validated for the specified message type. Messages that fail validation are dropped. Validation differs by message type.

  - Destination User Part Unavailable (DUPU)—The User/Cause field must be present, and it must contain only valid cause and user codes.
  - Error—All mandatory fields must be present and contain only allowed values. Each error message must contain the required fields for that error code.
  - Notify—The status type and status information fields must contain allowed values only.

- **ss7 variant** {**ITU** | **ANSI** | **JAPAN** | **CHINA**}—Identifies the variant of SS7 used in your network. This option determines the valid format for point codes. After you configure the option and deploy an M3UA policy, you cannot change it unless you first remove the policy. The default variant is ITU.
- **strict-asp-state**—Performs application server process (ASP) state validation. The system maintains the ASP states of M3UA sessions and allows or drops ASP messages based on the validation result. If you do not enable strict ASP state validation, all ASP messages are forwarded uninspected. Strict ASP state checking is required if you want stateful failover or if you want to operate within a cluster. However, strict ASP state checking works in Override mode only, it does not work if you are running in Loadsharing or Broadcast mode (per RFC 4666). The inspection assumes there is one and only one ASP per endpoint.
- **timeout endpoint** *time*—Sets the idle timeout to remove statistics for an M3UA endpoint, in hh:mm:ss format. To have no timeout, specify 0. The default is 30 minutes (00:30:00).
- **timeout session** *time*—Sets the idle timeout to remove an M3UA session if you enable strict ASP state validation, in hh:mm:ss format. To have no timeout, specify 0. The default is 30 minutes (00:30:00). Disabling this timeout can prevent the system from removing stale sessions.

Example

The following is an example of an M3UA policy map and service policy.

```
hostname(config)# policy-map type inspect m3ua m3ua-map
hostname(config-pmap)# match message class 2 id 6
hostname(config-pmap-c)# drop
hostname(config-pmap-c)# match message class 9
hostname(config-pmap-c)# drop
hostname(config-pmap-c)# match dpc 1-5-1
hostname(config-pmap-c)# drop log
hostname(config-pmap-c)# parameters
hostname(config-pmap-p)# ss7 variant ITU
hostname(config-pmap-p)# timeout endpoint 00:45:00

hostname(config)# policy-map global_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect m3ua m3ua-map

hostname(config)# service-policy global_policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure the Mobile Network Inspection Service Policy , on page 363.

Configure the Mobile Network Inspection Service Policy

Inspections for the protocols used in mobile networks are not enabled in the default inspection policy, so you must enable them if you need these inspections. You can simply edit the default global inspection policy to add these inspections. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Procedure

Step 1 If necessary, create an L3/L4 class map to identify the traffic for which you want to apply the inspection.

```
class-map name
   match parameter
```

Example:

```
hostname(config)# class-map mobile_class_map
hostname(config-cmap)# match access-list mobile
```

In the default global policy, the inspection_default class map is a special class map that includes default ports for all inspection types (**match default-inspection-traffic**). If you are using this class map in either the default policy or for a new service policy, you can skip this step.

For information on matching statements, see Create a Layer 3/4 Class Map for Through Traffic, on page 234.

Step 2 Add or edit a policy map that sets the actions to take with the class map traffic: **policy-map** *name*

Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

Step 3 Identify the L3/L4 class map you are using for the inspection: **class** *name*

Example:

```
hostname(config-pmap)# class inspection_default
```

To edit the default policy, or to use the special inspection_default class map in a new policy, specify **inspection_default** for the *name*. Otherwise, you are specifying the class you created earlier in this procedure.

Step 4 Enable the inspections.

In the following commands, the inspection policy maps are optional. If you created any of these maps to customize the inspection, specify their names on the appropriate command. For Diameter, you can also specify a TLS proxy to enable inspection of encrypted messages.

- **inspect gtp** [*map_name*]—To enable GTP inspection.
- **inspect sctp** [*map_name*]—To enable SCTP inspection.

- **inspect diameter** [*map_name*] [**tls-proxy** *proxy_name*]—To enable Diameter inspection.

  Note
  If you specify a TLS proxy for Diameter inspection, and you apply NAT port redirection to Diameter server traffic (for example, redirect server traffic from port 5868 to 3868), configure inspection globally or on the ingress interface only. If you apply the inspection to the egress interface, NATed Diameter traffic bypasses inspection.

- **inspect m3ua** [*map_name*]—To enable M3UA inspection.

Example:

```
hostname(config-class)# inspect gtp
hostname(config-class)# inspect sctp
hostname(config-class)# inspect diameter
hostname(config-class)# inspect m3ua
```

Note
If you are editing the default global policy (or any in-use policy) to use a different inspection policy map, you must remove the inspection with the **no inspect** version of the command, and then re-add it with the new inspection policy map name. For example, to change the policy map for GTP:

```
hostname(config-class)# no inspect gtp
hostname(config-class)# inspect gtp gtp-map
```

Step 5 If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy global_policy global
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Configure RADIUS Accounting Inspection

RADIUS accounting inspection is not enabled by default. You must configure it if you want RADIUS accounting inspection.

Procedure

Step 1 Configure a RADIUS Accounting Inspection Policy Map, on page 365.

Step 2 Configure the RADIUS Accounting Inspection Service Policy, on page 366.

Configure a RADIUS Accounting Inspection Policy Map

You must create a RADIUS accounting inspection policy map to configure the attributes needed for the inspection.

Procedure

Step 1 Create a RADIUS accounting inspection policy map: **policy-map type inspect radius-accounting** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

Step 4 Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option.

- **send response**—Instructs the ASA to send Accounting-Request Start and Stop messages to the sender of those messages (which are identified in the **host** command).
- **enable gprs**—Implement GPRS over-billing protection. The ASA checks for the 3GPP VSA 26-10415 attribute in the Accounting-Request Stop and Disconnect messages in order to properly handle secondary PDP contexts. If this attribute is present, then the ASA tears down all connections that have a source IP matching the User IP address on the configured interface.
- **validate-attribute** *number*—Additional criteria to use when building a table of user accounts when receiving Accounting-Request Start messages. These attributes help when the ASA decides whether to tear down connections.

  If you do not specify additional attributes to validate, the decision is based solely on the IP address in the Framed IP Address attribute. If you configure additional attributes, and the ASA receives a start accounting message that includes an address that is currently being tracked, but the other attributes to validate are different, then all connections started using the old attributes are torn down, on the assumption that the IP address has been reassigned to a new user.

  Values range from 1-191, and you can enter the command multiple times. For a list of attribute numbers and their descriptions, see http://www.iana.org/assignments/radius-types.
- **host** *ip_address* [**key** *secret*]—The IP address of the RADIUS server or GGSN. You can optionally include a secret key so that the ASA can validate the message. Without the key, only the IP address is checked. You can repeat this command to identify multiple RADIUS and GGSNs hosts. The ASA receives a copy of the RADIUS accounting messages from these hosts.

- **timeout users** *time*—Sets the idle timeout for users (in hh:mm:ss format). To have no timeout, specify 00:00:00. The default is one hour.

Example

```
policy-map type inspect radius-accounting radius-acct-pmap
  parameters
    send response
    enable gprs
    validate-attribute 31
    host 10.2.2.2 key 123456789
    host 10.1.1.1 key 12345
class-map type management radius-class
  match port udp eq radius-acct
policy-map global_policy
  class radius-class
    inspect radius-accounting radius-acct-pmap
```

Configure the RADIUS Accounting Inspection Service Policy

RADIUS accounting inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. Because RADIUS accounting inspection is for traffic directed to the ASA, you must configure it as a management inspection rule rather than a standard rule.

Procedure

Step 1 Create an L3/L4 management class map to identify the traffic for which you want to apply the inspection, and identify the matching traffic.

```
class-map type management name
match {port | access-list} parameter
```

Example:

```
hostname(config)# class-map type management radius-class-map
hostname(config-cmap)# match port udp eq radius-acct
```

In this example, the match is for the radius-acct UDP port, which is 1646. You can specify a different port, a range of ports (**match port udp range** *number1 number2*) or use **match access-list** *acl_name* and use an ACL.

Step 2 Add or edit a policy map that sets the actions to take with the class map traffic: **policy-map** *name*

Example:

```
hostname(config)# policy-map global_policy
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name.

Step 3 Identify the L3/L4 management class map you are using for RADIUS accounting inspection: **class** *name*

Example:

```
hostname(config-pmap)# class radius-class-map
```

Step 4 Configure RADIUS accounting inspection: **inspect radius-accounting** [*radius-accounting_policy_map*]

Where *radius_accounting_policy_map* is the RADIUS accounting inspection policy map you created in Configure a RADIUS Accounting Inspection Policy Map, on page 365.

Example:

```
hostname(config-class)# no inspect radius-accounting
hostname(config-class)# inspect radius-accounting radius-class-map
```

Note
If you are editing an in-use policy to use a different inspection policy map, you must remove the RADIUS accounting inspection with the **no inspect radius-accounting** command, and then re-add it with the new inspection policy map name.

Step 5 If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy global_policy global
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Monitoring Mobile Network Inspection

The following topics explain how to monitor mobile network inspection.

Monitoring GTP Inspection

To display the GTP configuration, enter the **show service-policy inspect gtp** command in privileged EXEC mode.

Use the **show service-policy inspect gtp statistics** command to show the statistics for GTP inspection. The following is sample output:

```
firewall(config)# show service-policy inspect gtp statistics
```

```
GPRS GTP Statistics:
  version_not_support               0     msg_too_short             0
  unknown_msg                       0     unexpected_sig_msg        0
  unexpected_data_msg               0     ie_duplicated             0
  mandatory_ie_missing              0     mandatory_ie_incorrect    0
  optional_ie_incorrect             0     ie_unknown                0
  ie_out_of_order                   0     ie_unexpected             0
  total_forwarded                  67     total_dropped             1
  signalling_msg_dropped            1     data_msg_dropped          0
  signalling_msg_forwarded         67     data_msg_forwarded        0
  total created_pdp                33     total deleted_pdp         32
  total created_pdpmcb             31     total deleted_pdpmcb      30
  total dup_sig_mcbinfo             0     total dup_data_mcbinfo    0
  no_new_sgw_sig_mcbinfo            0     no_new_sgw_data_mcbinfo   0
  pdp_non_existent                  1
```

You can get statistics for a specific GTP endpoint by entering the IP address on the **show service-policy inspect gtp statistics** *ip_address* command.

```
firewall(config)# show service-policy inspect gtp statistics 10.9.9.9
1 in use, 1 most used, timeout 0:30:00
 GTP GSN Statistics for 10.9.9.9, Idle 0:00:34, restart counter 0
  Tunnels Active                    0
  Tunnels Created                   1
  Tunnels Destroyed                 0
  Total Messages Received           1
                         Signalling Messages        Data Messages
  total received                   1                          0
  dropped                          0                          0
  forwarded                        1                          0
```

Use the **show service-policy inspect gtp pdp-context** command to display PDP context-related information. For GTPv2, this is the bearer context. For example:

```
ciscoasa(config)# show service-policy inspect gtp pdp-context
4 in use, 5 most used

Version v1,   TID 050542012151705f,  MS Addr 2005:a00::250:56ff:fe96:eec,
SGSN Addr 10.0.203.22,      Idle 0:52:01,   Timeout 3:00:00,   APN ssenoauth146

Version v2,   TID 0505420121517056,  MS Addr 100.100.100.102,
SGW Addr 10.0.203.24,      Idle 0:00:05,   Timeout 3:00:00,   APN ssenoauth146

Version v2,   TID 0505420121517057,  MS Addr 100.100.100.103,
SGW Addr 10.0.203.25,      Idle 0:00:04,   Timeout 3:00:00,   APN ssenoauth146

Version v2,   TID 0505420121517055,  MS Addr 100.100.100.101,
SGW Addr 10.0.203.23,      Idle 0:00:06,   Timeout 3:00:00,   APN ssenoauth146

ciscoasa(config)# show service-policy inspect gtp pdp-context detail
1 in use, 1 most used

Version v1,   TID 050542012151705f,  MS Addr 2005:a00::250:56ff:fe96:eec,
SGSN Addr 10.0.203.22,      Idle 0:06:14,   Timeout 3:00:00,   APN ssenoauth146

    user_name (IMSI):  50502410121507    MS address: 2005:a00::250:56ff:fe96:eec
    nsapi: 5                linked nsapi: 5
    primary pdp: Y         sgsn is Remote
    sgsn_addr_signal: 10.0.203.22    sgsn_addr_data: 10.0.203.22
    ggsn_addr_signal: 10.0.202.22    ggsn_addr_data: 10.0.202.22
    sgsn control teid:     0x00000001    sgsn data teid:        0x000003e8
```

```
    ggsn control teid:     0x000f4240    ggsn data teid:        0x001e8480
    signal_sequence:               18     state:    Ready
...
```

The PDP or bearer context is identified by the tunnel ID (TID), which is a combination of the values for IMSI and NSAPI (GTPv0-1) or IMSI and EBI (GTPv2). A GTP tunnel is defined by two associated contexts in different GSN or SGW/PGW nodes and is identified with a Tunnel ID. A GTP tunnel is necessary to forward packets between an external packet data network and a mobile subscriber (MS) user.

Monitoring SCTP

You can use the following commands to monitor SCTP.

- **show service-policy inspect sctp**

  Displays SCTP inspection statistics. The sctp-drop-override counter increments each time a PPID is matched to a drop action, but the packet was not dropped because it contained data chunks with different PPIDs. For example:

```
ciscoasa# show service-policy inspect sctp
Global policy:
  Service-policy: global_policy
    Class-map: inspection_default
      Inspect: sctp sctp, packet 153302, lock fail 0, drop 20665, reset-drop 0,
5-min-pkt-rate 0 pkts/sec, v6-fail-close 0, sctp-drop-override 4910
        Match ppid 30 35
          rate-limit 1000 kbps, chunk 2354, dropped 10, bytes 21408, dropped-bytes 958

        Match: ppid 40
           drop, chunk 5849
        Match: ppid 55
           log, chunk 9546
```

- **show sctp** [**detail**]

  Displays current SCTP cookies and associations. Add the **detail** keyword to see detailed information about SCTP associations. The detailed view also shows information about multi-homing, multiple streams, and fragment reassembly.

```
ciscoasa# show sctp

AssocID: 71adeb15
Local:  192.168.107.12/50001 (ESTABLISHED)
Remote: 192.168.108.122/2905 (ESTABLISHED)
Secondary Conn List:
    192.168.108.12(192.168.108.12):2905 to 192.168.107.122(192.168.107.122):50001
    192.168.107.122(192.168.107.122):50001 to 192.168.108.12(192.168.108.12):2905
    192.168.108.122(192.168.108.122):2905 to 192.168.107.122(192.168.107.122):50001
    192.168.107.122(192.168.107.122):50001 to 192.168.108.122(192.168.108.122):2905
    192.168.108.12(192.168.108.12):2905 to 192.168.107.12(192.168.107.12):50001
    192.168.107.12(192.168.107.12):50001 to 192.168.108.12(192.168.108.12):2905
```

- **show conn protocol sctp**

  Displays information about current SCTP connections.

- **show local-host** [**connection sctp** *start*[*-end*]]

Displays information on hosts making SCTP connections through the ASA, per interface. Add the **connection sctp** keyword to see only those hosts with the specified number or range of SCTP connections.

- **show traffic**

  Displays SCTP connection and inspection statistics per interface if you enable the **sysopt traffic detailed-statistics** command.

Monitoring Diameter

You can use the following commands to monitor Diameter.

- **show service-policy inspect diameter**

  Displays Diameter inspection statistics. For example:

```
ciscoasa# show service-policy inspect diameter
Global policy:
  Service-policy: global_policy
    Class-map: inspection_default
      Inspect: Diameter Diameter_map, packet 0, lock fail 0, drop 0, -drop 0,
5-min-pkt-rate 0 pkts/sec, v6-fail-close 0
        Class-map: log_app
            Log: 5849
         Class-map: block_ip
             drop-connection: 2
```

- **show diameter**

  Displays state information for each Diameter connection. For example:

```
ciscoasa# show diameter
Total active diameter sessions: 5
Session 3638
       ==========
       ref_count: 1 val = .; 1096298391; 2461;
           Protocol : diameter Context id : 0
           From inside:211.1.1.10/45169 to outside:212.1.1.10/3868
...
```

- **show conn detail**

  Displays connection information. Diameter connections are marked with the Q flag.

- **show tls-proxy**

  Displays information about the TLS proxy if you use one in Diameter inspection.

Monitoring M3UA

You can use the following commands to monitor M3UA.

- **show service-policy inspect m3ua drops**

  Displays drop statistics for M3UA inspection.

- **show service-policy inspect m3ua endpoint** [*IP_address*]

  Displays statistics for M3UA endpoints. You can specify an endpoint IP address to see information for a specific endpoint. For high availability or clustered systems, the statistics are per unit, they are not synchronized across units. For example:

```
ciscoasa# sh service-policy inspect m3ua endpoint
 M3UA Endpoint Statistics for 10.0.0.100, Idle : 0:00:06 :
                   Forwarded        Dropped          Total Received
 All Messages      21                5                 26
 DATA Messages     9                 5                 14
 M3UA Endpoint Statistics for 10.0.0.110, Idle : 0:00:06 :
                   Forwarded        Dropped          Total Received
 All Messages      21                8                 29
 DATA Messages     9                 8                 17
```

- **show service-policy inspect m3ua session**

  Displays information about M3UA sessions if you enable strict application server process (ASP) state validation. Information includes source association ID, whether the session is single or double exchange, and in clustering, whether it is a cluster owner session or a backup session. In a cluster with 3 or more units, you might see stale backup sessions if a unit leaves and then returns to the cluster. These stale sessions are removed when they time out, unless you disabled session timeout.

```
Ciscoasa# show service-policy inspect m3ua session
0 in use, 0 most used
Flags: o - cluster owner session, b - cluster backup session
       d - double exchange      , s - single exchange
AssocID: cfc59fbe in Down state, idle:0:00:05, timeout:0:01:00, bd
AssocID: dac2e123 in Active state, idle:0:00:18, timeout:0:01:00, os
```

- **show service-policy inspect m3ua table**

  Displays the run-time M3UA inspection table, including classification rules.

- **show conn detail**

  Displays connection information. M3UA connections are marked with the v flag.

History for Mobile Network Inspection

Feature Name	Releases	Feature Information
GTPv2 inspection and improvements to GTPv0/1 inspection.	9.5(1)	GTP inspection can now handle GTPv2. In addition, GTP inspection for all versions now supports IPv6 addresses. We changed the **match message id** command to **match message** {**v1** | **v2**} **id** *message_id* . We replaced the **timeout gsn** command with **timeout endpoint**. We removed the **gsn** keyword from the **clear/show service-policy inspect gtp statistics** command; now, simply enter the endpoint ID to see or clear these statistics. The **clear/show service-policy inspect gtp request** and **pdpmcb** commands now include a **version** keyword, so you can display information about a specific GTP version.
SCTP inspection	9.5(2)	You can now apply application-layer inspection to Stream Control Transmission Protocol (SCTP) traffic to apply actions based on payload protocol identifier (PPID). We added or modified the following commands: **clear conn protocol sctp**, **inspect sctp**, **match ppid**, **policy-map type inspect sctp**, **show conn protocol sctp**, **show local-host connection sctp**, **show service-policy inspect sctp**.
Diameter inspection	9.5(2)	You can now apply application-layer inspection to Diameter traffic and also apply actions based on application ID, command code, and attribute-value pair (AVP) filtering. We added or modified the following commands: **class-map type inspect diameter**, **diameter**, **inspect diameter**, **match application-id**, **match avp**, **match command-code**, **policy-map type inspect diameter**, **show conn detail**, **show diameter**, **show service-policy inspect diameter**, **unsupported**.
Diameter inspection improvements	9.6(1)	You can now inspect Diameter over TCP/TLS traffic, apply strict protocol conformance checking, and inspect Diameter over SCTP in cluster mode. We added or modified the following commands: **client clear-text**, **inspect diameter**, **strict-diameter**.
SCTP stateful inspection in cluster mode	9.6(1)	SCTP stateful inspection now works in cluster mode. You can also configure SCTP stateful inspection bypass in cluster mode. We did not introduce or change any commands.

Feature Name	Releases	Feature Information
MTP3 User Adaptation (M3UA) inspection.	9.6(2)	You can now inspect M3UA traffic and also apply actions based on point code, service indicator, and message class and type. We added or modified the following commands: **clear service-policy inspect m3ua** {**drops** | **endpoint** [*IP_address*]}, **inspect m3ua**, **match dpc**, **match opc**, **match service-indicator**, **policy-map type inspect m3ua**, **show asp table classify domain inspect-m3ua**, **show conn detail**, **show service-policy inspect m3ua** {**drops** | **endpoint** [*IP_address*]}, **ss7 variant**, **timeout endpoint**.
Support for SCTP multi-streaming reordering and reassembly and fragmentation. Support for SCTP multi-homing, where the SCTP endpoints have more than one IP address.	9.7(1)	The system now fully supports SCTP multi-streaming reordering, reassembly, and fragmentation, which improves Diameter and M3UA inspection effectiveness for SCTP traffic. The system also supports SCTP multi-homing, where the endpoints have more than one IP address each. For multi-homing, the system opens pinholes for the secondary addresses so that you do not need to write access rules to allow them. SCTP endpoints must be limited to 3 IP addresses each. We modified the output of the following command: **show sctp detail**.
M3UA inspection improvements.	9.7(1)	M3UA inspection now supports stateful failover, semi-distributed clustering, and multihoming. You can also configure strict application server process (ASP) state validation and validation for various messages. Strict ASP state validation is required for stateful failover and clustering. We added or modified the following commands: **clear service-policy inspect m3ua session** [**assocID** *id*], **match port sctp**, **message-tag-validation**, **show service-policy inspect m3ua drop**, **show service-policy inspect m3ua endpoint**, **show service-policy inspect m3ua session**, **show service-policy inspect m3ua table**, **strict-asp-state**, **timeout session**.
Support for setting the TLS proxy server SSL cipher suite.	9.8(1)	You can now set the SSL cipher suite when the ASA acts as a TLS proxy server. Formerly, you could only set global settings for the ASA using the **ssl cipher** command. We introduced the following command: **server cipher-suite**
GTP inspection enhancements for MSISDN and Selection Mode filtering, anti-replay, and user spoofing protection.	9.10(1)	You can now configure GTP inspection to drop Create PDP Context messages based on Mobile Station International Subscriber Directory Number (MSISDN) or Selection Mode. You can also implement anti-replay and user spoofing protection. We added the following commands: **anti-replay**, **gtp-u-header-check**, **match msisdn**, **match selection-mode**.

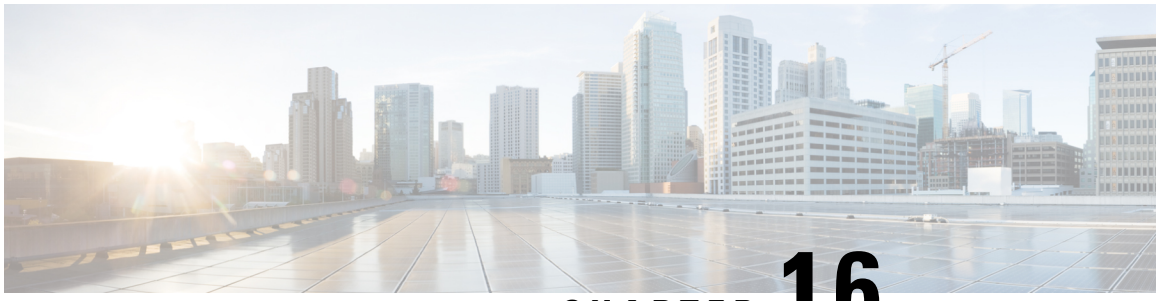
Feature Name	Releases	Feature Information
GTPv1 release 10.12 support.	9.12(1)	The system now supports GTPv1 release 10.12. Previously, the system supported release 6.1. The new support includes recognition of 25 additional GTPv1 messages and 66 information elements. In addition, there is a behavior change. Now, any unknown message IDs are allowed. Previously, unknown messages were dropped and logged. We did not add or change any commands.
Location logging for mobile stations (GTP inspection).	9.13(1)	You can configure GTP inspection to log the initial location of a mobile station and subsequent changes to the location. Tracking location changes can help you identify possibly fraudulent roaming charges. We added the following command: **location-logging**.
GTPv2 and GTPv1 release 15 support.	9.13(1)	The system now supports GTPv2 3GPP 29.274 V15.5.0. For GTPv1, support is up to 3GPP 29.060 V15.2.0. The new support includes recognition of 2 additional messages and 53 information elements. We did not add or change any commands.
Ability to specify the IMSI prefixes to be dropped in GTP inspection.	9.16(1)	GTP inspection lets you configure IMSI prefix filtering, to identify the Mobile Country Code/Mobile Network Code (MCC/MNC) combinations to allow. You can now do IMSI filtering on the MCC/MNC combinations that you want to drop. This way, you can list out the unwanted combinations, and default to allowing all other combinations. We added the following command: **drop mcc**.
Secure Firewall 3100 support for the Carrier license	9.18(1)	The Carrier license enables Diameter, GTP/GPRS, SCTP inspection. New/Modified commands: **feature carrier**



PART V

Connection Management and Threat Detection

- Connection Settings, on page 377
- Quality of Service, on page 411
- Threat Detection, on page 423



CHAPTER 16

Connection Settings

This chapter describes how to configure connection settings for connections that go through the ASA, or for management connections that go to the ASA.

- Connection settings, on page 377
- Configure Connection Settings, on page 378
- Monitoring Connections, on page 406
- History for Connection Settings, on page 407

Connection settings

Connection settings refer to a variety of features that manage connections through a device.

Connection setting types

Connection settings include these types:

- **Global timeouts for various protocols**—All global timeouts have default values. Change them only if you experience premature connection loss.
- **Connection timeouts per traffic class**—You can override the global timeouts for specific types of traffic using service policies. All traffic class timeouts have default values, so you do not have to set them.
- **Connection limits and TCP Intercept**—By default, there are no limits on how many connections can go through (or to) the ASA. Use service policy rules to set limits on particular traffic classes. This protects servers from denial of service attacks Particularly, you can set limits on embryonic connections (those that have not finished the TCP handshake), which protects against SYN flooding attacks. When embryonic limits are exceeded, the TCP Intercept component activates to proxy connections and throttle attacks.
- **Dead connection detection (DCD)**—Enable dead connection detection if you have persistent connections that are valid but often idle. These connections may get closed when they exceed idle timeout settings. Dead connection detection identifies idle but valid connections and keeps them alive by resetting their idle timers. Whenever idle times are exceeded, DCD probes both sides of the connection to see if both sides agree that the connection is valid.. The **show service-policy** command output includes counters to show the amount of activity from DCD. You can use the **show conn detail** command to get information about the initiator and responder and how often each has sent probes.
- **TCP sequence randomization**—Each TCP connection has two initial sequence numbers (ISN): one generated by the client and one generated by the server. By default, the ASA randomizes the ISN of the TCP SYN passing in both the inbound and outbound directions. Randomization prevents an attacker

from predicting the next ISN for a new connection and potentially hijacking the new session. However, TCP sequence randomization effectively breaks TCP SACK (Selective Acknowledgement), as the sequence numbers the client sees are different from what the server sees. You can disable randomization per traffic class if desired.

- **TCP normalization**—The TCP Normalizer protects against abnormal packets. You can configure how some types of packet abnormalities are handled by traffic class.
- **TCP state bypass**—You can bypass TCP state checking if you use asymmetrical routing in your network.
- **SCTP state bypass**—You can bypass Stream Control Transmission Protocol (SCTP) stateful inspection if you do not want SCTP protocol validation.
- **Flow offloading**—You can identify select traffic to be offloaded to a super fast path, where the flows are switched in the NIC itself. Offloading can help you improve performance for data-intensive applications such as large file transfers.
- **IPsec flow offload**—After the initial setup of an IPsec site-to-site VPN or remote access VPN security association (SA), IPsec connections are offloaded to the field-programmable gate array (FPGA) in the device, which should improve device performance. This feature is enabled by default on platforms that support it.

Configure Connection Settings

Connection limits, timeouts, TCP Normalization, TCP sequence randomization, and decrementing time-to-live (TTL) have default values that are appropriate for most networks. You need to configure these connection settings only if you have unusual requirements, your network has specific types of configuration, or if you are experiencing unusual connection loss due to premature idle timeouts.

Other connection-related features are not enabled. You would configure these services on specific traffic classes only, and not as a general service. These features include the following: TCP Intercept, TCP State Bypass, Dead Connection Detection (DCD), SCTP state bypass, flow offload.

The following general procedure covers the gamut of possible connection setting configurations. Pick and choose which to implement based on your needs.

Before you begin

When a packet enters the device, the firewall first evaluates access control rules and NAT to determine whether a connection entry needs to be created. A connection entry is created regardless of whether an ARP entry exists for the next hop. Thus, the existence of a connection does not mean that any packet within the scope of the connection made it through the device. Connection settings such as idle timeouts ensure that unwanted connections do not linger and take up system resources.

Procedure

Step 1 Configure Global Timeouts, on page 379. These settings change the default idle timeouts for various protocols for all traffic that passes through the device. If you are having problems with connections being reset due to premature timeouts, first try changing the global timeouts.

Step 2 Protect Servers from a SYN Flood DoS Attack (TCP Intercept), on page 381. Use this procedure to configure TCP Intercept.

Step 3 Customize Abnormal TCP Packet Handling (TCP Maps, TCP Normalizer), on page 384, if you want to alter the default TCP Normalization behavior for specific traffic classes.

Step 4 Bypass TCP state checks for asymmetrical routing (TCP state bypass), on page 387, if you have this type of routing environment.

Step 5 Disable TCP Sequence Randomization, on page 391, if the default randomization is scrambling data for certain connections.

Step 6 Offload Large Flows, on page 392, if you need to improve performance in a computing intensive data center.

Step 7 Configure Connection Settings for Specific Traffic Classes (All Services), on page 400. This is a catch-all procedure for connection settings. These settings can override the global defaults for specific traffic classes using service policy rules. You also use these rules to customize TCP Normalizer, change TCP sequence randomization, decrement time-to-live on packets, and implement other optional features.

Step 8 Configure TCP Options, on page 405, if you need to force resets or change some other standard TCP behavior.

Configure Global Timeouts

You can set the global idle timeout durations for the connection and translation slots of various protocols. If the slot has not been used for the idle time specified, the resource is returned to the free pool.

Changing the global timeout sets a new default timeout, which in some cases can be overridden for particular traffic flows through service policies.

For protocols that do not have a specific configurable timeout setting, such as GRE, the idle timeout is 2 minutes.

Procedure

Use the **timeout** command to set global timeouts.

All timeout values are in the format *hh*:*mm*:*ss*, with a maximum duration of 1193:0:0 in most cases. Use the **clear configure timeout** command to reset all timeouts to their default values. If you want to simply reset one timer to the default, enter the **timeout** command for that setting with the default value.

Use **0** for the value to disable a timer.

You can configure the following global timeouts.

- **timeout conn** *hh*:*mm*:*ss*—The idle time after which a connection closes, between 0:5:0 and 1193:0:0. The default is 1 hour (1:0:0).
- **timeout half-closed** *hh*:*mm*:*ss*—The idle time until a TCP half-closed connection closes. A connection is considered half-closed if both the FIN and FIN-ACK have been seen. If only the FIN has been seen, the regular **conn** timeout applies. The minimum is 30 seconds. The default is 10 minutes.
- **timeout udp** *hh*:*mm*:*ss*—The idle time until a UDP connection closes. This duration must be at least 1 minute. The default is 2 minutes.
- **timeout icmp** *hh*:*mm*:*ss*—The idle time for ICMP, between 0:0:2 and 1193:0:0. The default is 2 seconds (0:0:2).

- **timeout icmp-error** *hh*:*mm*:*ss*—The idle time before the ASA removes an ICMP connection after receiving an ICMP echo-reply packet, between 0:0:0 and 0:1:0 or the **timeout icmp** value, whichever is lower. The default is 0 (disabled). When this timeout is disabled, and you enable ICMP inspection, then the ASA removes the ICMP connection as soon as an echo-reply is received; thus any ICMP errors that are generated for the (now closed) connection are dropped. This timeout delays the removal of ICMP connections so you can receive important ICMP errors.
- **timeout sunrpc** *hh*:*mm*:*ss*—The idle time until a SunRPC slot is freed. This duration must be at least 1 minute. The default is 10 minutes.
- **timeout H323** *hh*:*mm*:*ss*—The idle time after which H.245 (TCP) and H.323 (UDP) media connections close, between 0:0:0 and 1193:0:0. The default is 5 minutes (0:5:0). Because the same connection flag is set on both H.245 and H.323 media connections, the H.245 (TCP) connection shares the idle timeout with the H.323 (RTP and RTCP) media connection.
- **timeout h225** *hh*:*mm*:*ss*—The idle time until an H.225 signaling connection closes. The H.225 default timeout is 1 hour (1:0:0). To close a connection immediately after all calls are cleared, a value of 1 second (0:0:1) is recommended.
- **timeout mgcp** *hh*:*mm*:*ss*—The idle time after which an MGCP media connection is removed, between 0:0:0 and 1193:0:0. The default is 5 minutes (0:5:0)
- **timeout mgcp-pat** *hh*:*mm*:*ss*—The absolute interval after which an MGCP PAT translation is removed, between 0:0:0 and 1193:0:0. The default is 5 minutes (0:5:0). The minimum time is 30 seconds.
- **timeout sctp** *hh*:*mm*:*ss*—The idle time until a Stream Control Transmission Protocol (SCTP) connection closes, between 0:1:0 and 1193:0:0. The default is 2 minutes (0:2:0).
- **timeout sip** *hh*:*mm*:*ss*—The idle time until a SIP signaling port connection closes, between 0:5:0 and 1193:0:0. The default is 30 minutes (0:30:0).
- **sip-callinit-timeout** *hh*:*mm*:*ss*—The idle time after which a SIP session in Call_Init call state (after the initial invite message is received) will be deleted, i.e. when no response is received, between 0:0:1 and 1193:0:0. The default is 32 seconds (0:00:32).
- **timeout sip_media** *hh*:*mm*:*ss*—The idle time until an SIP media port connection closes. This duration must be at least 1 minute. The default is 2 minutes. The SIP media timer is used for SIP RTP/RTCP with SIP UDP media packets, instead of the UDP inactivity timeout.
- **timeout sip-provisional-media** *hh*:*mm*:*ss*—The timeout value for SIP provisional media connections, between 0:1:0 and 0:30:0. The default is 2 minutes.
- **timeout sip-invite** *hh*:*mm*:*ss*—The idle time after which pinholes for PROVISIONAL responses and media xlates will be closed, between 0:1:0 and 00:30:0. The default is 3 minutes (0:3:0).
- **timeout sip-disconnect** *hh*:*mm*:*ss*—The idle time after which a SIP session is deleted if the 200 OK is not received for a CANCEL or a BYE message, between 0:0:1 and 00:10:0. The default is 2 minutes (0:2:0).
- **timeout uath** *hh*:*mm*:*ss* {**absolute** | **inactivity**}—The duration before the authentication and authorization cache times out and the user has to reauthenticate the next connection, between 0:0:0 and 1193:0:0. The default is 5 minutes (0:5:0). The default timer is **absolute**; you can set the timeout to occur after a period of inactivity by entering the **inactivity** keyword. The uauth duration must be shorter than the xlate duration. Set to 0 to disable caching. Do not use 0 if passive FTP is used for the connection or if the virtual http command is used for web authentication.

- **timeout xlate** *hh*:*mm*:*ss*—The idle time until a translation slot is freed. This duration must be at least 1 minute. The default is 3 hours.
- **timeout pat-xlate** *hh*:*mm*:*ss*—The idle time until a PAT translation slot is freed, between 0:0:30 and 0:5:0. The default is 30 seconds. You may want to increase the timeout if upstream routers reject new connections using a freed PAT port because the previous connection might still be open on the upstream device.
- **timeout tcp-proxy-reassembly** *hh*:*mm*:*ss*—The idle timeout after which buffered packets waiting for reassembly are dropped, between 0:0:10 and 1193:0:0. The default is 1 minute (0:1:0).
- **timeout floating-conn** *hh*:*mm*:*ss*—When multiple routes exist to a network with different metrics, the ASA uses the one with the best metric at the time of connection creation. If a better route becomes available, then this timeout lets connections be closed so a connection can be reestablished to use the better route. The default is 0 (the connection never times out). To make it possible to use better routes, set the timeout to a value between 0:0:30 and 1193:0:0.
- **timeout conn-holddown** *hh*:*mm*:*ss*—How long the system should maintain a connection when the route used by the connection no longer exists or is inactive. If the route does not become active within this holddown period, the connection is freed. The purpose of the connection holddown timer is to reduce the effect of route flapping, where routes might come up and go down quickly. You can reduce the holddown timer to make route convergence happen more quickly. The default is 15 seconds, the range is 00:00:00 to 00:00:15.
- **timeout igp stale-route** *hh*:*mm*:*ss*—How long to keep a stale route before removing it from the router information base. These routes are for interior gateway protocols such as OSPF. The default is 70 seconds (00:01:10), the range is 00:00:10 to 00:01:40.

Protect Servers from a SYN Flood DoS Attack (TCP Intercept)

A SYN-flooding denial of service (DoS) attack occurs when an attacker sends a series of SYN packets to a host. These packets usually originate from spoofed IP addresses. The constant flood of SYN packets keeps the server SYN queue full, which prevents it from servicing connection requests from legitimate users.

You can limit the number of embryonic connections to help prevent SYN flooding attacks. An embryonic connection is a connection request that has not finished the necessary handshake between source and destination.

When the embryonic connection threshold of a connection is crossed, the ASA acts as a proxy for the server and generates a SYN-ACK response to the client SYN request using the SYN cookie method, so that the connection is not added to the SYN queue of the targeted host. The SYN cookie is the initial sequence number returned in the SYN-ACK that is constructed from MSS, time stamp, and a mathematical hash of other items to essentially create a secret. If the ASA receives an ACK back from the client with the correct sequence number and within the valid time window, it can then authenticate that the client is real and allow the connection to the server. The component that performs the proxy is called TCP Intercept.

The end-to-end process for protecting a server from a SYN flood attack involves setting connection limits, enabling TCP Intercept statistics, and then monitoring the results.

Before you begin

- Ensure that you set the embryonic connection limit lower than the TCP SYN backlog queue on the server that you want to protect. Otherwise, valid clients can no longer access the server during a SYN attack.

To determine reasonable values for embryonic limits, carefully analyze the capacity of the server, the network, and server usage.

- Depending on the number of CPU cores on your ASA model, the maximum concurrent and embryonic connections can exceed the configured numbers due to the way each core manages connections. In the worst case scenario, the ASA allows up to *n*-1 extra connections and embryonic connections, where *n* is the number of cores. For example, if your model has 4 cores, if you configure 6 concurrent connections and 4 embryonic connections, you could have an additional 3 of each type. To determine the number of cores for your model, enter the **show cpu core** command.

Procedure

Step 1 Create an L3/L4 class map to identify the servers you are protecting. Use an access-list match.

```
class-map name
match parameter
```

Example:

```
hostname(config)# access-list servers extended permit tcp any host 10.1.1.5 eq http
hostname(config)# access-list servers extended permit tcp any host 10.1.1.6 eq http
hostname(config)# class-map protected-servers
hostname(config-cmap)# match access-list servers
```

Step 2 Add or edit a policy map that sets the actions to take with the class map traffic, and identify the class map.

```
policy-map name
class name
```

Example:

```
hostname(config)# policy-map global_policy
hostname(config-pmap)# class protected-servers
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name. For the class map, specify the class you created earlier in this procedure.

Step 3 Set the embryonic connection limits.

- **set connection embryonic-conn-max** *n*—The maximum number of simultaneous embryonic TCP connections allowed, from 0 and 2000000. The default is 0, which allows unlimited connections.
- **set connection per-client-embryonic-max** *n*—The maximum number of simultaneous embryonic TCP connections allowed per client, from 0 and 2000000. The default is 0, which allows unlimited connections.
- **set connection syn-cookie-mss** *n*—The server maximum segment size (MSS) for SYN-cookie generation for embryonic connections upon reaching the embryonic connections limit, from 48 to 65535 . The default is 1380. This setting is meaningful only if you configure **set connection embryonic-conn-max** or **per-client-embryonic-max**.

Example:

```
hostname(config-pmap-c)# set connection embryonic-conn-max 1000
hostname(config-pmap-c)# set connection per-client-embryonic-max 50
```

Step 4 If you are editing an existing service policy (such as the default global policy called global_policy), you can skip this step. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy global_policy global
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Step 5 Configure threat detection statistics for attacks intercepted by TCP Intercept.

threat-detection statistics tcp-intercept [**rate-interval** *minutes*] [**burst-rate** *attacks_per_sec*] [**average-rate** *attacks_per_sec*]

Where:

- **rate-interval** *minutes* sets the size of the history monitoring window, between 1 and 1440 minutes. The default is 30 minutes. During this interval, the ASA samples the number of attacks 30 times.
- **burst-rate** *attacks_per_sec* sets the threshold for syslog message generation, between 25 and 2147483647. The default is 400 per second. When the burst rate is exceeded, syslog message 733104 is generated.
- **average-rate** *attacks_per_sec* sets the average rate threshold for syslog message generation, between 25 and 2147483647. The default is 200 per second. When the average rate is exceeded, syslog message 733105 is generated.

Example:

```
hostname(config)# threat-detection statistics tcp-intercept
```

Step 6 Monitor the results with the following commands:

- **show threat-detection statistics top tcp-intercept** [**all** | **detail**]—View the top 10 protected servers under attack. The **all** keyword shows the history data of all the traced servers. The **detail** keyword shows history sampling data. The ASA samples the number of attacks 30 times during the rate interval, so for the default 30 minute period, statistics are collected every 60 seconds.
- **clear threat-detection statistics tcp-intercept**—Erases TCP Intercept statistics.

Example:

```
hostname(config)# show threat-detection statistics top tcp-intercept
Top 10 protected servers under attack (sorted by average rate)
Monitoring window size: 30 mins    Sampling interval: 30 secs
<Rank> <Server IP:Port> <Interface> <Ave Rate> <Cur Rate> <Total> <Source IP (Last Attack
Time)>
--------------------------------------------------------------------------------
```

```
1    10.1.1.5:80 inside 1249 9503 2249245 <various> Last: 10.0.0.3 (0 secs ago)
2    10.1.1.6:80 inside 10 10 6080 10.0.0.200 (0 secs ago)
```

Customize Abnormal TCP Packet Handling (TCP Maps, TCP Normalizer)

The TCP normalizer identifies abnormal packets that the ASA can act on when they are detected; for example, the ASA can allow, drop, or clear the packets. TCP normalization helps protect the ASA from attacks. TCP normalization is always enabled, but you can customize how some features behave.

The default configuration includes the following settings:

```
no check-retransmission
no checksum-verification
exceed-mss allow
queue-limit 0 timeout 4
reserved-bits allow
syn-data allow
synack-data drop
invalid-ack drop
seq-past-window drop
tcp-options range 6 7 clear
tcp-options range 9 18 clear
tcp-options range 20 255 clear
tcp-options md5 allow
tcp-options mss allow
tcp-options selective-ack allow
tcp-options timestamp allow
tcp-options window-scale allow
ttl-evasion-protection
urgent-flag clear
window-variation allow-connection
```

To customize the TCP normalizer, first define the settings using a TCP map. Then, you can apply the map to selected traffic classes using service policies.

Procedure

Step 1 Create a TCP map to specify the TCP normalization criteria that you want to look for: **tcp-map** *tcp-map-name*

Step 2 Configure the TCP map criteria by entering one or more of the following commands. The defaults are used for any commands you do not enter. Use the **no** form of a command to disable the setting.

- **check-retransmission**—Prevent inconsistent TCP retransmission. This command is disabled by default.
- **checksum-verification**—Verify the TCP checksum, dropping packets that fail verification. This command is disabled by default.
- **exceed-mss** {**allow** | **drop**}—Allow or drop packets whose data length exceeds the TCP maximum segment size. The default is to allow the packets.
- **invalid-ack** {**allow** | **drop**}—Allow or drop packets with an invalid ACK. The default is to drop the packet, with the exception of WAAS connections, where they are allowed. You might see invalid ACKs in the following instances:

  - In the TCP connection SYN-ACK-received status, if the ACK number of a received TCP packet is not exactly the same as the sequence number of the next TCP packet sending out, it is an invalid ACK.
  - Whenever the ACK number of a received TCP packet is greater than the sequence number of the next TCP packet sending out, it is an invalid ACK.

- **queue-limit** *pkt_num* [**timeout** *seconds*]—Set the maximum number of out-of-order packets that can be buffered and put in order for a TCP connection, between 1 and 250 packets. The default is 0, which means this setting is disabled and the default system queue limit is used depending on the type of traffic:
  - Connections for application inspection (the **inspect** command), and TCP check-retransmission (the TCP map **check-retransmission** command) have a queue limit of 3 packets. If the ASA receives a TCP packet with a different window size, then the queue limit is dynamically changed to match the advertised setting.
  - For other TCP connections, out-of-order packets are passed through untouched.

  If you set the **queue-limit** command to be 1 or above, then the number of out-of-order packets allowed for all TCP traffic matches this setting. For example, for application inspection and TCP check-retransmission traffic, any advertised settings from TCP packets are ignored in favor of the **queue-limit** setting. For other TCP traffic, out-of-order packets are now buffered and put in order instead of passed through untouched.

  The **timeout** *seconds* argument sets the maximum amount of time that out-of-order packets can remain in the buffer, between 1 and 20 seconds; if they are not put in order and passed on within the timeout period, then they are dropped. The default is 4 seconds. You cannot change the timeout for any traffic if the *pkt_num* argument is set to 0; you need to set the limit to be 1 or above for the **timeout** keyword to take effect.

- **reserved-bits** {**allow** | **clear** | **drop**}—Set the action for reserved bits in the TCP header. You can **allow** the packet (without changing the bits), **clear** the bits and allow the packet, or **drop** the packet.
- **seq-past-window** {**allow** | **drop**}—Set the action for packets that have past-window sequence numbers, namely the sequence number of a received TCP packet is greater than the right edge of the TCP receiving window. You can **allow** the packets only if the **queue-limit** command is set to 0 (disabled). The default is to drop the packets.
- **synack-data** {**allow** | **drop**}—Allow or drop TCP SYNACK packets that contain data. The default is to drop the packet.
- **syn-data** {**allow** | **drop**}—Allow or drop SYN packets with data. The default is to allow the packet.
- **tcp-options** {**md5** | **mss** | **selective-ack** | **timestamp** | **window-scale** | **range** *lower upper*} *action*—Set the action for packets with TCP options. These options are named: **md5**, **mss**, **selective-ack** (selective acknowledgment mechanism), **timestamp**, and **window-scale** (window scale mechanism). For other options, you specify them by number on the **range** keyword, where the range limits are 6-7, 9-18, and 20-255. To target a single option by number, enter the same number for the lower and upper range. You can enter the command multiple times in a map to define your complete policy. Note that if a TCP connection is inspected, all options are cleared except the MSS and selective-acknowledgment (SACK) options, regardless of your configuration. Following are the possible actions:
  - **allow** [**multiple**]—Allow packets that contain a single option of this type. This is the default for all of the named options. If you want to allow packets even if they contain more than one instance of the option, add the **multiple** keyword. (The **multiple** keyword is not available with **range**.)

- **maximum** *limit*—For **mss** only. Set the maximum segment size to the indicated limit, from 68-65535. The default TCP MSS is defined on the **sysopt connection tcpmss** command.
- **clear**—Remove the options of this type from the header and allow the packet. This is the default for all of the numbered options. Note that clearing the timestamp option disables PAWS and RTT.
- **drop**—Drop packets that contain this option. This action is available for **md5** and **range** only.

The TCP normalizer maintains the timestamp database if timestamp options are present during the initial TCP handshake. The timestamp values depend on the exchange of timestamp options between client and server. If there is a communication gap or if the server suddenly starts sending timestamp options, PAWS (Protect Against Wrapped Sequence numbers) check failures can occur. When a PAWS check fails, the ASA drops the packet and sends an ACK with the most recent timestamp values to the other side.

- **ttl-evasion-protection**—Have the maximum TTL for a connection be determined by the TTL in the initial packet. The TTL for subsequent packets can decrease, but it cannot increase. The system will reset the TTL to the lowest previously-seen TTL for that connection. This protects against TTL evasion attacks. TTL evasion protection is enabled by default, so you would only need to enter the **no** form of this command.

  For example, an attacker can send a packet that passes policy with a very short TTL. When the TTL goes to zero, a router between the ASA and the endpoint drops the packet. It is at this point that the attacker can send a malicious packet with a long TTL that appears to the ASA to be a retransmission and is passed. To the endpoint host, however, it is the first packet that has been received by the attacker. In this case, an attacker is able to succeed without security preventing the attack.

- **urgent-flag** {**allow** | **clear**}—Set the action for packets with the URG flag. You can **allow** the packet, or **clear** the flag and allow the packet. The default is to clear the flag.

  The URG flag is used to indicate that the packet contains information that is of higher priority than other data within the stream. The TCP RFC is vague about the exact interpretation of the URG flag, therefore end systems handle urgent offsets in different ways, which may make the end system vulnerable to attacks.

- **window-variation** {**allow** | **drop**}—Allow or drop a connection that has changed its window size unexpectedly. The default is to allow the connection.

  The window size mechanism allows TCP to advertise a large window and to subsequently advertise a much smaller window without having accepted too much data. From the TCP specification, “shrinking the window” is strongly discouraged. When this condition is detected, the connection can be dropped.

Step 3 Apply the TCP map to a traffic class using a service policy.

a) Define the traffic class with an L3/L4 class map and add the map to a policy map.

```
class-map name
match parameter
policy-map name
class name
```

Example:

```
hostname(config)# class-map normalization
hostname(config-cmap)# match any
hostname(config)# policy-map global_policy
```

```
hostname(config-pmap)# class normalization
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name. For information on matching statements for class maps, see Create a Layer 3/4 Class Map for Through Traffic, on page 234.

b) Apply the TCP map: **set connection advanced-options** *tcp-map-name*

Example:

```
hostname(config-pmap-c)# set connection advanced-options tcp_map1
```

c) If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy global_policy global
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Examples

For example, to allow urgent flag and urgent offset packets for all traffic sent to the range of TCP ports between the well known FTP data port and the Telnet port, enter the following commands:

```
hostname(config)# tcp-map tmap
hostname(config-tcp-map)# urgent-flag allow
hostname(config-tcp-map)# class-map urg-class
hostname(config-cmap)# match port tcp range ftp-data telnet
hostname(config-cmap)# policy-map pmap
hostname(config-pmap)# class urg-class
hostname(config-pmap-c)# set connection advanced-options tmap
hostname(config-pmap-c)# service-policy pmap global
```

Bypass TCP state checks for asymmetrical routing (TCP state bypass)

TCP state bypass allows traffic to bypass stateful inspection checks on ASA devices.

Implementation considerations

If you have an asymmetrical routing environment in your network, where the outbound and inbound flow for a given connection can go through two different ASA devices, you need to implement TCP state bypass on the affected traffic.

However, TCP state bypass weakens the security of your network, so you should apply bypass on very specific, limited traffic classes.

The asymmetrical routing problem

The asymmetrical routing problem occurs when return traffic for a connection goes through a different firewall than the outbound traffic. This causes packets to be dropped because they do not match existing connection entries in the fast path table.

Traffic inspection and fast path operation

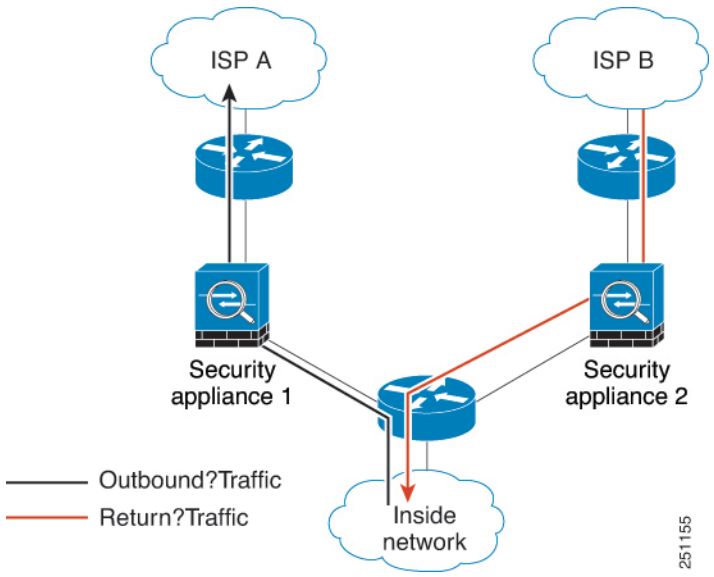
By default, the ASA inspects all traffic using the Adaptive Security Algorithm. The ASA allows or drops traffic based on the security policy. The ASA maximizes the firewall performance by checking the state of each packet (new connection or established connection). It assigns new connection SYN packets to the session management path. It assigns established connections to the fast path. It assigns packets requiring advanced inspection to the control plane path.

The ASA allows TCP packets that match existing connections to pass through via the fast path without rechecking every aspect of the security policy. This feature improves performance. However, the fast path establishment method can interfere with asymmetric routing solutions. The method uses SYN packets and performs checks such as TCP sequence number validation. Both outbound and inbound flows of a connection must pass through the same ASA device.

Asymmetric routing scenario

For example, a new connection goes to Security Appliance 1. The SYN packet goes through the session management path, and an entry for the connection is added to the fast path table. If subsequent packets of this connection go through Security Appliance 1, then the packets match the entry in the fast path, and are passed through. But if subsequent packets go to Security Appliance 2, where no SYN packet went through the session management path, then no entry exists in the fast path for the connection, and Security Appliance 2 drops the packets. The following figure shows an asymmetric routing example where the outbound traffic goes through a different ASA than the inbound traffic:

Figure 44: Asymmetric routing



If you have asymmetric routing configured on upstream routers and traffic alternates between two ASA devices, then you can configure TCP state bypass for specific traffic. TCP state bypass alters the way sessions are established in the fast path and disables the fast path checks. This feature treats TCP traffic similarly to a UDP connection: when a non-SYN packet matching the specified networks enters the ASA device, and there is not a fast path entry, then the packet goes through the session management path to establish the connection in the fast path. Once in the fast path, the traffic bypasses the fast path checks.

Guidelines and limitations for TCP state bypass

TCP state bypass unsupported features

The system does not support these features when you use TCP state bypass:

- Application inspection—Inspection requires both inbound and outbound traffic to go through the same ASA, so inspection is not applied to TCP state bypass traffic.
- AAA authenticated sessions—When a user authenticates with one ASA, traffic returning via the other ASA will be denied because the user did not authenticate with that ASA.
- TCP Intercept, maximum embryonic connection limit, TCP sequence number randomization—The ASA does not keep track of the state of the connection, so these features are not applied.
- TCP normalization—The TCP normalizer is disabled.

TCP state bypass NAT guidelines

Because the translation session is established separately for each device, be sure to configure static NAT on both devices for TCP state bypass traffic. If you use dynamic NAT, the address chosen for the session on Device 1 will differ from the address chosen for the session on Device 2.

Configure TCP State Bypass

To bypass TCP state checking in asymetrical routing environments, carefully define a traffic class that applies to the affected hosts or networks only, then enable TCP State Bypass on the traffic class using a service policy. Because bypass reduces the security of the network, limit its application as much as possible.

Before you begin

If there is no traffic on a given connection for 2 minutes, the connection times out. You can override this default using the **set connection timeout idle** command for the TCP state bypass traffic class. Normal TCP connections timeout by default after 60 minutes.

Procedure

Step 1 Create an L3/L4 class map to identify the hosts that require TCP State Bypass. Use an access-list match to identify the source and destination hosts.

```
class-map name
match parameter
```

Example:

```
hostname(config)# access-list bypass extended permit tcp host 10.1.1.1 host 10.2.2.2
hostname(config)# class-map bypass-class
hostname(config-cmap)# match access-list bypass
```

Step 2 Add or edit a policy map that sets the actions to take with the class map traffic, and identify the class map.

```
policy-map name
class name
```

Example:

```
hostname(config)# policy-map global_policy
hostname(config-pmap)# class bypass-class
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name. For the class map, specify the class you created earlier in this procedure.

Step 3 Enable TCP State Bypass on the class: **set connection advanced-options tcp-state-bypass**

Step 4 If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy global_policy global
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Example

The following is a sample configuration for TCP state bypass:

```
hostname(config)# access-list tcp_bypass extended permit tcp 10.1.1.0 255.255.255.224 any

hostname(config)# class-map tcp_bypass
hostname(config-cmap)# description "TCP traffic that bypasses stateful firewall"
hostname(config-cmap)# match access-list tcp_bypass

hostname(config-cmap)# policy-map tcp_bypass_policy
hostname(config-pmap)# class tcp_bypass
hostname(config-pmap-c)# set connection advanced-options tcp-state-bypass

hostname(config-pmap-c)# service-policy tcp_bypass_policy interface outside
```

Disable TCP Sequence Randomization

Each TCP connection has two ISNs: one generated by the client and one generated by the server. The ASA randomizes the ISN of the TCP SYN passing in both the inbound and outbound directions.

Randomizing the ISN of the protected host prevents an attacker from predicting the next ISN for a new connection and potentially hijacking the new session. However, TCP sequence randomization effectively breaks TCP SACK (Selective Acknowledgement), as the sequence numbers the client sees are different from what the server sees.

You can disable TCP initial sequence number randomization if necessary, for example, because data is getting scrambled. For example:

- If another in-line firewall is also randomizing the initial sequence numbers, there is no need for both firewalls to be performing this action, even though this action does not affect the traffic.
- If you use eBGP multi-hop through the ASA, and the eBGP peers are using MD5. Randomization breaks the MD5 checksum.
- You use a WAAS device that requires the ASA not to randomize the sequence numbers of connections.
- You enable hardware bypass for the ISA 3000, and TCP connections are dropped when the ISA 3000 is no longer part of the data path.

Note We do not recommend disabling TCP sequence randomization when using clustering. There is a small chance that some TCP sessions won't be established, because the SYN/ACK packet might be dropped.

Procedure

Step 1 Create an L3/L4 class map to identify the traffic whose TCP sequence numbers should not be randomized. The class match should be for TCP traffic; you can identify specific hosts (with an ACL), do a TCP port match, or simply match any traffic.

```
class-map name
match parameter
```

Example:

```
hostname(config)# access-list preserve-sq-no extended permit tcp any host 10.2.2.2
hostname(config)# class-map no-tcp-random
hostname(config-cmap)# match access-list preserve-sq-no
```

Step 2 Add or edit a policy map that sets the actions to take with the class map traffic, and identify the class map.

```
policy-map name
class name
```

Example:

```
hostname(config)# policy-map global_policy
hostname(config-pmap)# class no-tcp-random
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name. For the class map, specify the class you created earlier in this procedure.

Step 3 Disable TCP sequence number randomization on the class:

set connection random-sequence-number disable

If you later decide to turn it back on, replace “disable “with **enable**.

Step 4 If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy global_policy global
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Offload Large Flows

If you deploy the ASA on supported devices in a data center, you can identify select traffic to be offloaded to a super fast path, where traffic is switched in the NIC itself. Offloading can help you improve performance for data-intensive applications such as large file transfers.

- High Performance Computing (HPC) Research sites, where the ASA is deployed between storage and high compute stations. When one research site backs up using FTP file transfer or file sync over NFS, the large amount of data traffic affects all contexts on the ASA. Offloading FTP file transfer and file sync over NFS reduces the impact on other traffic.
- High Frequency Trading (HFT), where the ASA is deployed between workstations and the Exchange, mainly for compliance purposes. Security is usually not a concern, but latency is a major concern.

Before being offloaded, the ASA first applies normal security processing, such as access rules and inspection, during connection establishment. The ASA also does session tear-down. But once a connection is established, if it is eligible to be offloaded, further processing happens in the NIC rather than the ASA.

Offloaded flows continue to receive limited stateful inspection, such as basic TCP flag and option checking, and checksum verification if you configure it. The system can selectively escalate packets to the firewall system for further processing if necessary.

To identify flows that can be offloaded, you create a service policy rule that applies the flow offloading service. A matching flow is then offloaded if it meets the following conditions:

- IPv4 addresses only.

- TCP, UDP, GRE only.
- Standard or 802.1Q tagged Ethernet frames only.
- (Transparent mode only.) Multicast flows for bridge groups that contain two and only two interfaces.

Reverse flows for offloaded flows are also offloaded.

Restrictions for flow offload implementation

Device limitations

The feature is supported on the following devices:

- Secure Firewall 3100 (static only)
- Secure Firewall 4200 (static only)
- Firepower 4100/9300

Flows that cannot be offloaded

Do not attempt to offload these flow types:

- Any flows that do not use IPv4 addressing, such as IPv6 addressing.
- Flows for any protocol other than TCP, UDP, and GRE.



Note PPTP GRE connections cannot be offloaded.

- Flows that require inspection. In some cases, such as FTP, the secondary data channel can be offloaded although the control channel cannot be offloaded.
- IPsec and TLS/DTLS VPN connections that terminate on the device.
- Multicast flows in routed mode.
- Multicast flows in transparent mode for bridge groups that have three or more interfaces.
- TCP Intercept flows.
- TCP state bypass flows. You cannot configure flow offload and TCP state bypass on the same traffic.
- AAA cut-through proxy flows.
- Vpath, VXLAN related flows.
- Flows tagged with security groups.
- Except for the Secure Firewall 4200: Reverse flows that are forwarded from a different cluster node, in the case of asymmetric flows in a cluster.
- Centralized flows in a cluster, if the flow owner is not the control unit.

Additional limitations

Consider these operational restrictions when implementing flow offload:

- Flow offload and Dead Connection Detection (DCD) are not compatible. Do not configure DCD on connections that can be offloaded.
- If more than one flow that matches flow offload conditions are queued to be offloaded at the same time to the same location on the hardware, only the first flow is offloaded. The other flows are processed normally. This is called a *collision*. Use the **show flow-offload flow** command in the CLI to display statistics for this situation.
- Although offloaded flows pass through FXOS interfaces, statistics for these flows do not appear on the logical device interface. Thus, logical device interface counters and packet rates do not reflect offloaded flows.

Cluster redirection

When traffic for an existing flow is sent to a different node, then that traffic is redirected to the owner node over the cluster control link. Because asymmetric flows can create a lot of traffic on the cluster control link, offloading these flows can improve performance, for example, for distribited site-to-site VPN mode. This feature is supported for these environments:

- Secure Firewall 4200
- Routed and transparent mode
- Spanned EtherChannel mode
- Multiple context mode

Conditions for reversing offload

After a flow is offloaded, packets within the flow are returned to the firewall operating system for further processing if they meet these conditions:

- They include TCP options other than Timestamp.
- They are fragmented.
- They are subject to Equal-Cost Multi-Path (ECMP) routing, and ingress packets move from one interface to another.
- A change to the routing table applies to the flow. A packet is returned to the device to determine the new route.

Configure Flow Offload

To configure flow offload, you must enable the service and then create service policies to identify the traffic that is eligible for offloading. For the Firepower 4100/9300: When you initially enable the service, you must reboot. Flow offload is enabled by default for the Secure Firewall 3100/4200.

Procedure

Step 1 Enable the flow offload service.

flow-offload enable

Flow offload is enabled by default for the Secure Firewall 3100/4200.

For the Firepower 4100/9300: When you initially enable the service, you must reboot.

If a reload is required, there are special considerations for clusters or failover pairs if you want a hitless change:

- Clustering—First enter the command on the control node, but do not reboot the control node immediately. Instead, reboot each node of the cluster first, then return to the control node and reboot it. You can then configure the offloading service policy on the control node.
- Failover—First enter the command on the active unit, but do not reboot it immediately. Instead, reboot the standby unit, then reboot the active unit. You can then configure the offloading service policy on the active unit.

In multiple-context mode, enabling or disabling flow offload enables or disables it for all contexts. You cannot have different settings per context.

Example:

```
ciscoasa(config)# flow-offload enable

WARNING: This command will take effect after the running-config is
saved and the system has been rebooted.

ciscoasa(config)# write memory
ciscoasa(config)# reload
```

Step 2 (Secure Firewall 4200) Configure cluster redirect.

flow-offload cluster-redirect

Cluster redirect is enabled by default. This feature requires the **flow-offload enable** command, which is also enabled by default.

For asymmetric flows, cluster redirect lets the forwarding node offload flows to hardware. When traffic for an existing flow is sent to a different node, then that traffic is redirected to the owner node over the cluster control link. Because asymmetric flows can create a lot of traffic on the cluster control link, letting the forwarder offload these flows can improve performance.

Step 3 Create the service policy rule that identifies traffic that is eligible for offload.

a) Create an L3/L4 class map to identify the traffic that is eligible for flow offload. Matching by access-list or port would be the most typical options.

```
class-map name
match parameter
```

Example:

```
hostname(config)# access-list offload permit tcp 10.1.1.0 255.255.255.224 any
hostname(config)# class-map flow_offload
hostname(config-cmap)# match access-list offload
```

b) Add or edit a policy map that sets the actions to take with the class map traffic, and identify the class map.

policy-map *name*
class *name*

Example:

```
hostname(config)# policy-map offload_policy
hostname(config-pmap)# class flow_offload
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name. For the class map, specify the class you created earlier in this procedure.

c) Enable flow offload on the class: **set connection advanced-options flow-offload**

d) If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy offload_policy interface outside
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Example

The following example classifies all TCP traffic from the 10.1.1.0 255.255.255.224 subnet as eligible for offload and attaches the policy to the outside interface.

```
hostname(config)# access-list offload permit tcp 10.1.1.0 255.255.255.224 any
hostname(config)# class-map flow_offload
hostname(config-cmap)# match access-list offload
hostname(config)# policy-map offload_policy
hostname(config-pmap)# class flow_offload
hostname(config-pmap-c)# set connection advanced-options flow-offload
hostname(config)# service-policy offload_policy interface outside
```

IPsec flow offload

IPsec flow offload is a performance optimization feature that

- offloads IPsec connections to field-programmable gate array (FPGA) or specialized hardware components after initial setup
- improves device performance by handling pre-decryption, decryption, pre-encryption, and encryption processing in hardware, and

- is enabled by default on supporting device models while allowing system software to handle inner flow security policies.

IPsec flow offload characteristics

After the initial setup of an IPsec site-to-site VPN or remote access VPN security association (SA), IPsec connections are offloaded to the field-programmable gate array (FPGA) in the device. This process improves device performance.In the Secure Firewall 1200 series, IPsec connections are offloaded to the Marvell Cryptographic Accelerator (CPT) to improve device performance.

Offloaded operations include pre-decryption and decryption processing on ingress and egress. The system software applies your security policies for traffic within the inner flow.

IPsec flow offload applies to these device types:

- Secure Firewall 1200
- Secure Firewall 3100
- Secure Firewall 4200

IPsec flow offload is also used when the device's VTI loopback interface is enabled.

For asymmetric flows in cluster distributed site-to-site VPN mode, IPsec flow offload lets the flow owner decrypt IPsec traffic in hardware that was forwarded over the cluster control link. This feature is not configurable and is always available with IPsec flow offload.

Limitations for IPsec flow offload

IPsec flows that are not offloaded include:

- IKEv1 tunnels. Only IKEv2 tunnels will be offloaded. IKEv2 supports stronger ciphers.
- Flows that have volume-based rekeying configured.
- Flows that have compression configured.
- Flows that have post-fragmentation configured.
- Flows that have anti-replay window size other than 64 bits and anti-replay is not disabled.
- AH format. Only ESP or NAT-T format will be supported.
- Flows that have firewall filter enabled.
- Modes:
  - Transport mode flows. Only tunnel mode flows will be offloaded.
  - Multiple context mode.

Configure IPsec Flow Offload

IPsec flow offload is enabled by default on hardware platforms that support the feature. However, egress optimization is not enabled by default, so you need to configure it if you want the feature.

Before you begin

IPsec flow offload is configured globally. You cannot configure it for selected traffic flows.

Use the **no** form of these commands to disable the features.

To see the current configuration state, use the **show flow-offload ipsec info** command.

Procedure

Step 1 Enable IPsec flow offload.

flow-offload-ipsec

Step 2 Enable egress optimization to optimize the data path to enhance performance for single tunnel flows.

flow-offload-ipsec egress-optimization

The configuration for egress optimization is separate from flow offload. However, even if enabled, it is effective only if you also enable IPsec flow offload. Egress optimization is not enabled by default.

DTLS crypto acceleration

ASA supports the Datagram Transport Layer Security (*DTLS*) cryptographic acceleration for specific firewall models. This feature improves the throughput of the DTLS-encrypted and DTLS-decrypted traffic for both IPv4 and IPv6 traffic. The ASA also performs optimization of the egress-encrypted packets to improve latency. The data path is optimized to enhance performance for single tunnel flows. Both features are enabled by default and work only for DTLS 1.2.

Supported models

- Secure Firewall 3100
- Secure Firewall 4200

Crypto accelerator details

In Secure Firewall 3100 and 4200, FPGA and the Nitrox V crypto accelerator support DTLS cryptographic acceleration

Configure DTLS Crypto Acceleration

By default, DTLS crypto acceleration is enabled. You can disable it if desired.

The ASA will not perform DTLS crypto acceleration under the following conditions:

- The flows use DTLS 1.0 or packet compression.
- The DTLS keys are rekeyed.
- Clustering or multiple context mode.

Procedure

Step 1 Disable DTLS crypto acceleration on the device.

no flow-offload-dtls

Example:

```
ciscoasa(config)# no flow-offload-dtls
```

To re-enable it, use the **flow-offload-dtls** command.

Step 2 Disable optimization of egress-encrypted packets and improve latency.

no flow-offload-dtls egress-optimization

Example:

```
ciscoasa(config)# flow-offload-dtls egress-optimization
```

To re-enable it, use the **flow-offload-dtls egress-optimization** command.

Monitor DTLS crypto acceleration

Use CLI commands on the threat defense device to verify and monitor DTLS crypto acceleration and optimization of egress-encrypted packets.

- To verify the status of DTLS crypto acceleration and optimization of egress-encrypted packets, use this command:

```
ciscoasa# show flow-offload-dtls info
  DTLS offload : Enabled
  Egress Optimization: Enabled
```

- To view the DTLS crypto acceleration statistics, use this command:

```
ciscoasa# show flow-offload-dtls statistics
  Packet stats of Pipe 0
  ----------------------
  Rx Packet count : 975638666
  Tx Packet count : 975638666
  Error Packet count : 0
  Drop Packet count : 0

  CAM stats of Pipe 0
  ----------------------
  Option ID Table CAM Hit Count : 1145314723
  Option ID Table CAM Miss Count : 0
  Tunnel Table CAM Hit Count : 0
  Tunnel Table CAM Miss Count : 0
  6-Tuple CAM Hit Count : 975638666
  6-Tuple CAM Miss Count : 169676057
   NOTE: The counters displayed are cumulative counters
```

```
   for all offload applications and indicates the total packets
  offloaded
```

- To view the device's Nitrox V crypto accelerator statistics for Secure Firewall 3100 and 4200, use this command:

```
ciscoasa# show crypto accelerator statistics

Crypto Accelerator Status
-------------------------
<snip>
[Offloaded SSL Input statistics, Pipe 0]
      Input packets: 290593023
      Input bytes: 147049729714
      Decrypted packets: 290593023
      Decrypted bytes: 147049729714
   [Offloaded SSL Output statistics, Pipe 0]
      Output packets: 254271808
      Output bytes: 136352952720
      Encrypted packets: 254271808
      Encrypted bytes: 136352952720
.
.
.
```

Configure Connection Settings for Specific Traffic Classes (All Services)

You can configure different connection settings for specific traffic classes using service policies. Use service policies to:

- Customize connection limits and timeouts used to protect against DoS and SYN-flooding attacks.
- Implement Dead Connection Detection so that valid but idle connections remain alive.
- Disable TCP sequence number randomization in cases where you do not need it.
- Customize how the TCP Normalizer protects against abnormal TCP packets.
- Implement TCP State Bypass for traffic subject to asymetrical routing. Bypass traffic is not subject to inspection.
- Implement Stream Control Transmission Protocol (SCTP) State Bypass to turn off SCTP stateful inspection.
- Implement flow offload to improve performance on supported hardware platforms.
- Decrement time-to-live (TTL) on packets so that the ASA will show up on trace route output.

Note If you decrement time to live, packets with a TTL of 1 will be dropped, but a connection will be opened for the session on the assumption that the connection might contain packets with a greater TTL. Note that some packets, such as OSPF hello packets, are sent with TTL = 1, so decrementing time to live can have unexpected consequences for transparent mode ASA devices. The decrement time-to-live settings does not impact the OSPF process when ASA is operating in a routed mode.

You can configure any combination of these settings for a given traffic class, except for TCP State Bypass and TCP Normalizer customization, which are mutually exclusive.

Tip This procedure shows a service policy for traffic that goes through the ASA. You can also configure the connection maximum and embryonic connection maximum for management (to the box) traffic.

Before you begin

If you want to customize the TCP Normalizer, create the required TCP Map before proceeding.

The **set connection** command (for connection limits and sequence randomization) and **set connection timeout** commands are described here separately for each parameter. However, you can enter the commands on one line, and if you enter them separately, they are shown in the configuration as one command.

Procedure

Step 1 Create an L3/L4 class map to identify the traffic for which you want to customize connection settings.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map CONNS
hostname(config-cmap)# match any
```

For information on matching statements, see Create a Layer 3/4 Class Map for Through Traffic, on page 234.

Step 2 Add or edit a policy map that sets the actions to take with the class map traffic, and identify the class map.

```
policy-map name
class name
```

Example:

```
hostname(config)# policy-map global_policy
```

```
hostname(config-pmap)# class CONNS
```

In the default configuration, the global_policy policy map is assigned globally to all interfaces. If you want to edit the global_policy, enter global_policy as the policy name. For the class map, specify the class you created earlier in this procedure.

Step 3 Set connection limits and TCP sequence number randomization. (TCP Intercept.)

By default, there are no connection limits. If you implement limits, the system must start tracking them, which can increase CPU and memory usage and result in operational problems for systems under heavy load, especially in a cluster.

- **set connection conn-max** *n*—(TCP, UDP, SCTP.) The maximum number of simultaneous connections that are allowed, between 0 and 2000000, for the entire class. The default is 0, which allows unlimited connections. For TCP connections, this applies to established connections only.
  - If two servers are configured to allow simultaneous connections, the connection limit is applied to each configured server separately.
  - Because the limit is applied to a class, one attack host can consume all the connections and leave none for the rest of the hosts that are matched to the class.
- **set connection per-client-max** *n*—(TCP, UDP, SCTP.) The maximum number of simultaneous connections allowed per client, between 0 and 2000000. The default is 0, which allows unlimited connections. This argument restricts the maximum number of simultaneous connections that are allowed for each host that is matched to the class. For TCP connections, this includes established, half-open, and half-closed connections.
- **set connection embryonic-conn-max** *n*—The maximum number of simultaneous embryonic TCP connections allowed, between 0 and 2000000. The default is 0, which allows unlimited connections. By setting a non-zero limit, you enable TCP Intercept, which protects inside systems from a DoS attack perpetrated by flooding an interface with TCP SYN packets. Also set the per-client options to protect against SYN flooding.
- **set connection per-client-embryonic-max** *n*—The maximum number of simultaneous embryonic TCP connections allowed per client, between 0 and 2000000. The default is 0, which allows unlimited connections.
- **set connection syn-cookie-mss** *n*—The server maximum segment size (MSS) for SYN-cookie generation for embryonic connections upon reaching the embryonic connections limit, from 48 to 65535 . The default is 1380. This setting is meaningful only if you configure **set connection embryonic-conn-max** or **per-client-embryonic-max**.
- **set connection random-sequence-number** {**enable** | **disable**}—Whether to enable or disable TCP sequence number randomization. Randomization is enabled by default.

Example:

```
hostname(config-pmap-c)# set connection conn-max 256 random-sequence-number disable
```

Step 4 Set connection timeouts and Dead Connection Detection (DCD).

The defaults described below assume you have not changed the global defaults for these behaviors using the **timeout** command; the global defaults override the ones described here. Enter **0** to disable the timer, so that a connection never times out.

- **set connection timeout embryonic** *hh*:*mm*:*ss*—The timeout period until a TCP embryonic (half-open) connection is closed, between 0:0:5 and 1193:00:00. The default is 0:0:30.
- **set connection timeout idle** *hh*:*mm*:*ss* [**reset**]—The idle timeout period after which an established connection of any protocol closes, between 0:0:1 and 1193:0:0. The default is 1:0:0. For TCP traffic, the **reset** keyword sends a reset to TCP endpoints when the connection times out.

  The default **udp** idle timeout is 2 minutes. The default **icmp** idle timeout is 2 seconds. The default **esp** and **ha** idle timeout is 30 seconds. For all other protocols, the default idle timeout is 2 minutes.
- **set connection timeout half-closed** *hh*:*mm*:*ss*—The idle timeout period until a half-closed connection is closed, between 0:5:0 (for 9.1(1) and earlier) or 0:0:30 (for 9.1(2) and later) and 1193:0:0. The default is 0:10:0. Half-closed connections are not affected by DCD. Also, the ASA does not send a reset when taking down half-closed connections.
- **set connection timeout dcd** [*retry-interval* [*max_retries*]]—Enable Dead Connection Detection (DCD). Before expiring an idle connection, the ASA probes the end hosts to determine if the connection is valid. If both hosts respond, the connection is preserved, otherwise the connection is freed. When operating in transparent firewall mode, you must configure static routes for the endpoints. You cannot configure DCD on connections that are also offloaded, so ensure DCD and flow offload traffic classes do not overlap. Use the **show conn detail** command to track how many DCD probes have been sent by the initiator and responder.

  The retry-interval sets the time duration in *hh***:***mm***:***ss* format to wait after each unresponsive DCD probe before sending another probe, between 0:0:1 and 24:0:0. The default is 0:0:15. The max-retries sets the number of consecutive failed retries for DCD before declaring the connection as dead. The minimum value is 1 and the maximum value is 255. The default is 5.

  For systems that are operating in a cluster or high-availability configuration, we recommend that you do not set the interval to less than one minute (0:1:0). If the connection needs to be moved between systems, the changes required take longer than 30 seconds, and the connection might be deleted before the change is accomplished.

Example:

```
hostname(config-pmap-c)# set connection timeout idle 2:0:0 embryonic 0:40:0
half-closed 0:20:0 dcd
```

Step 5 Decrement time-to-live (TTL) on packets that match the class: **set connection decrement-ttl**

This command, along with the **icmp unreachable** command, is required to allow a traceroute through the ASA that shows the ASA as one of the hops.

Example:

```
hostname(config)# class-map global-policy
hostname(config-cmap)# match any
hostname(config-cmap)# exit
hostname(config)# policy-map global_policy
hostname(config-pmap)# class global-policy
hostname(config-pmap-c)# set connection decrement-ttl
hostname(config-pmap-c)# exit
hostname(config)# icmp unreachable rate-limit 50 burst-size 6
```

Step 6 Set advanced connection options.

Advanced options are special purpose configurations that are not needed under normal circumstances. You configure them with the **set connection advanced-options** command.

- **set connection advanced-options** *tcp_map_name*—Customize TCP Normalizer behavior by applying a TCP map. For detailed information, see Customize Abnormal TCP Packet Handling (TCP Maps, TCP Normalizer), on page 384.
- **set connection advanced-options tcp-state-bypass**—Implement TCP State Bypass. For detailed information, see Bypass TCP state checks for asymmetrical routing (TCP state bypass), on page 387.
- **set connection advanced-options sctp-state-bypass**—Implement SCTP State Bypass to turn off SCTP stateful inspection. For more information, see SCTP Stateful Inspection, on page 331.
- **set connection advanced-options flow-offload**—(ASA on the Firepower 4100/9300 chassis, FXOS 1.1.3 or later, only.) Implement flow offloading. Eligible traffic is offloaded to a super fast path, where the flows are switched in the NIC itself. You must also enter the **flow-offload enable** command, which is not part of the service policy.

Example:

```
hostname(config-pmap-c)# set connection advanced-options tcp_map1
```

Step 7 If you are editing an existing service policy (such as the default global policy called global_policy), you are done. Otherwise, activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy global_policy global
```

The **global** keyword applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Example

The following example sets the connection limits and timeouts for all traffic:

```
hostname(config)# class-map CONNS
hostname(config-cmap)# match any
hostname(config-cmap)# policy-map CONNS
hostname(config-pmap)# class CONNS
hostname(config-pmap-c)# set connection conn-max 1000 embryonic-conn-max 3000
hostname(config-pmap-c)# set connection timeout idle 2:0:0 embryonic 0:40:0
half-closed 0:20:0 dcd
hostname(config-pmap-c)# service-policy CONNS interface outside
```

You can enter **set connection** commands with multiple parameters or you can enter each parameter as a separate command. The ASA combines the commands into one line in the running configuration. For example, if you entered the following two commands in class configuration mode:

```
hostname(config-pmap-c)# set connection conn-max 600
hostname(config-pmap-c)# set connection embryonic-conn-max 50
```

The output of the **show running-config policy-map** command would display the result of the two commands in a single, combined command:

```
set connection conn-max 600 embryonic-conn-max 50
```

Configure TCP Options

You can configure options to control some aspects of TCP behavior. The defaults for these settings are appropriate for most networks.

Procedure

Step 1 (CLI). Configure TCP reset behavior.

service { **resetinbound** [**interface** *interface_name*] | **resetoutbound** [**interface** *interface_name*] | **resetoutside** }

- **resetinbound**. Sends TCP resets for all inbound TCP sessions that attempt to transit the ASA and are denied by the ASA based on access lists or AAA settings. The ASA also sends resets for packets that are allowed by an access list or AAA, but do not belong to an existing connection and are denied by the stateful firewall. Traffic between same security level interfaces is also affected. When this option is not enabled, the ASA silently discards denied packets. If you do not specify an interface, then this setting applies to all interfaces.
- **resetoutbound**. Sends TCP resets for all outbound TCP sessions that attempt to transit the ASA and are denied by the ASA based on access lists or AAA settings. The ASA also sends resets for packets that are allowed by an access list or AAA, but do not belong to an existing connection and are denied by the stateful firewall. Traffic between same security level interfaces is also affected. When this option is not enabled, the ASA silently discards denied packets. This option is enabled by default. You might want to disable outbound resets to reduce the CPU load during traffic storms, for example.
- **resetoutside**. Enables resets for TCP packets that terminate at the least secure interface and are denied by the ASA based on access lists or AAA settings. The ASA also sends resets for packets that are allowed by an access list or AAA, but do not belong to an existing connection and are denied by the stateful firewall. When this option is not enabled, the ASA silently discards the packets of denied packets.

  We recommend that you use the this option with interface PAT. This option allows the ASA to terminate the IDENT from an external SMTP or FTP server. Actively resetting these connections avoids the 30-second timeout delay.

Step 2 Set TCP MSS to ensure that the maximum TCP segment size for through traffic does not exceed the value you set and that the maximum is not less than a specified size.

sysopt connection tcpmss [**minimum**] *bytes*

Without **minimum** keyword. Sets the maximum TCP segment size in bytes, between 48 and any maximum number. The default value is 1380 bytes. You can disable this feature by setting bytes to 0.

minimum. Overrides the maximum segment size to be no less than the specified bytes, between 48 and 65535 bytes. This feature is disabled by default (set to 0).

Step 3 Set TCP connection time wait.

sysopt connection timewait

Use this command to force each TCP connection to linger in a shortened TIME_WAIT state of at least 15 seconds after the final normal TCP close-down sequence. You might want to use this feature if an end host application default TCP terminating sequence is a simultaneous close.

Step 4 Set the maximum number of TCP unprocessed segments.

sysopt connection tcp-max-unprocessed-seg *segments*

Sets the maximum number of TCP unprocessed segments, from 6 to 24. The default is 6. If you find that SIP phones are not connecting to the call manager, you can try increasing the maximum number of unprocessed TCP segments.

Monitoring Connections

You can use the following commands to monitor connections:

- **show conn** [**detail**]

  Shows connection information. Detailed information uses flags to indicate special connection characteristics. For example, the "b" flag indicates traffic subject to TCP State Bypass.

  When you use the **detail** keyword, you can see information about Dead Connection Detection (DCD) probing, which shows how often the connection was probed by the initiator and responder. For example, the connection details for a DCD-enabled connection would look like the following:

```
TCP dmz: 10.5.4.11/5555 inside: 10.5.4.10/40299,
    flags UO , idle 1s, uptime 32m10s, timeout 1m0s, bytes 11828,
cluster sent/rcvd bytes 0/0, owners (0,255)
  Traffic received at interface dmz
        Locally received: 0 (0 byte/s)
  Traffic received at interface inside
        Locally received: 11828 (6 byte/s)
  Initiator: 10.5.4.10, Responder: 10.5.4.11
  DCD probes sent: Initiator 5, Responder 5
```

- **show flow-offload** {**info** [**detail**] | **cpu** | **flow** [**count** | **detail**] | **statistics**}

  Shows information about the flow offloading, including general status information, CPU usage for offloading, offloaded flow counts and details, and offloaded flow statistics.

- **show service-policy**

  Shows service policy statistics, including Dead Connection Detection (DCD) statistics.

- **show threat-detection statistics top tcp-intercept** [**all** | **detail**]

  View the top 10 protected servers under attack. The **all** keyword shows the history data of all the traced servers. The **detail** keyword shows history sampling data. The ASA samples the number of attacks 30 times during the rate interval, so for the default 30 minute period, statistics are collected every 60 seconds.

Note In the ASA configuration, embryonic connections—connection requests that have not yet completed the three-way handshake process—are closed quickly and not synchronized between the active and standby devices. This design ensures HA system efficiency and security. For this reason, there might be a difference in the number of connections on both ASAs, which is to be expected.

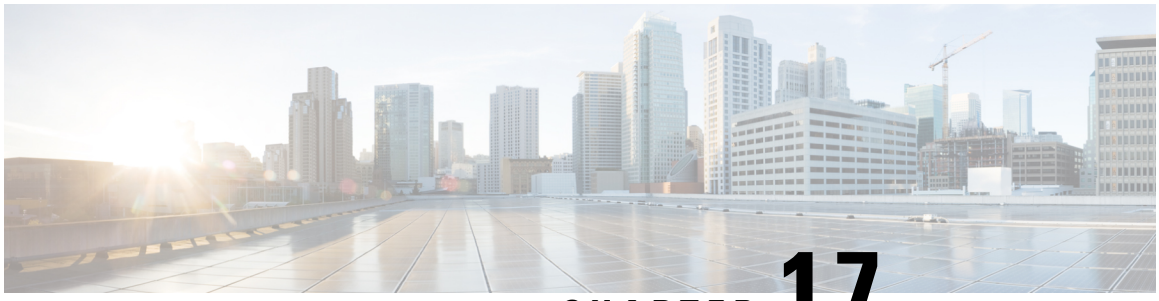
History for Connection Settings

Feature Name	Platform Releases	Description
TCP state bypass	8.2(1)	This feature was introduced. The following command was introduced: **set connection advanced-options tcp-state-bypass**.
Connection timeout for all protocols	8.2(2)	The idle timeout was changed to apply to all protocols, not just TCP. The following command was modified: **set connection timeout**
Timeout for connections using a backup static route	8.2(5)/8.4(2)	When multiple static routes exist to a network with different metrics, the ASA uses the one with the best metric at the time of connection creation. If a better route becomes available, then this timeout lets connections be closed so a connection can be reestablished to use the better route. The default is 0 (the connection never times out). To take advantage of this feature, change the timeout to a new value. We modified the following command: **timeout floating-conn**.
Configurable timeout for PAT xlate	8.4(3)	When a PAT xlate times out (by default after 30 seconds), and the ASA reuses the port for a new translation, some upstream routers might reject the new connection because the previous connection might still be open on the upstream device. The PAT xlate timeout is now configurable, to a value between 30 seconds and 5 minutes. We introduced the following command: **timeout pat-xlate**. *This feature is not available in 8.5(1) or 8.6(1).*
Increased maximum connection limits for service policy rules	9.0(1)	The maximum number of connections for service policy rules was increased from 65535 to 2000000. We modified the following commands: **set connection conn-max**, **set connection embryonic-conn-max**, **set connection per-client-embryonic-max**, **set connection per-client-max**.
Decreased the half-closed timeout minimum value to 30 seconds	9.1(2)	The half-closed timeout minimum value for both the global timeout and connection timeout was lowered from 5 minutes to 30 seconds to provide better DoS protection. We modified the following commands: **set connection timeout half-closed**, **timeout half-closed**.

Feature Name	Platform Releases	Description
Connection holddown timeout for route convergence.	9.4(3) 9.6(2)	You can now configure how long the system should maintain a connection when the route used by the connection no longer exists or is inactive. If the route does not become active within this holddown period, the connection is freed. You can reduce the holddown timer to make route convergence happen more quickly. However, the 15 second default is appropriate for most networks to prevent route flapping. We added the following command: **timeout conn-holddown**.
SCTP idle timeout and SCTP state bypass	9.5(2)	You can set an idle timeout for SCTP connections. You can also enable SCTP state bypass to turn off SCTP stateful inspection on a class of traffic. We added or modified the following commands: **timeout sctp**, **set connection advanced-options sctp-state-bypass**.
Flow offload for the ASA on the Firepower 9300.	9.5(2.1)	You can identify flows that should be offloaded from the ASA and switched directly in the NIC (on the Firepower 9300). This provides improved performance for large data flows in data centers. This feature requires FXOS 1.1.3. We added or modified the following commands: **clear flow-offload**, **flow-offload enable**, **set-connection advanced-options flow-offload**, **show conn detail**, **show flow-offload**.
Flow offload support for the ASA on the Firepower 4100 series.	9.6(1)	You can identify flows that should be offloaded from the ASA and switched directly in the NIC for the Firepower 4100 series. This feature requires FXOS 1.1.4. There are no new commands or ASDM screens for this feature.
Flow offload support for multicast connections in transparent mode.	9.6(2)	You can now offload multicast connections to be switched directly in the NIC on transparent mode Firepower 4100 and 9300 series devices. Multicast offload is available for bridge groups that contain two and only two interfaces. There are no new commands or ASDM screens for this feature.

Feature Name	Platform Releases	Description
Changes in TCP option handling.	9.6(2)	You can now specify actions for the TCP MSS and MD5 options in a packet's TCP header when configuring a TCP map. In addition, the default handling of the MSS, timestamp, window-size, and selective-ack options has changed. Previously, these options were allowed, even if there were more than one option of a given type in the header. Now, packets are dropped by default if they contain more than one option of a given type. For example, previously a packet with 2 timestamp options would be allowed, now it will be dropped. You can configure a TCP map to allow multiple options of the same type for MD5, MSS, selective-ack, timestamp, and window-size. For the MD5 option, the previous default was to clear the option, whereas the default now is to allow it. You can also drop packets that contain the MD5 option. For the MSS option, you can set the maximum segment size in the TCP map (per traffic class). The default for all other TCP options remains the same: they are cleared. We modified the following command: **timeout igp stale-route**.
Stale route timeout for interior gateway protocols	9.7(1)	You can now configure the timeout for removing stale routes for interior gateway protocols such as OSPF. We added the following command: **timeout igp stale-route**.
Global timeout for ICMP errors	9.8(1)	You can now set the idle time before the ASA removes an ICMP connection after receiving an ICMP echo-reply packet. When this timeout is disabled (the default), and you enable ICMP inspection, then the ASA removes the ICMP connection as soon as an echo-reply is received; thus any ICMP errors that are generated for the (now closed) connection are dropped. This timeout delays the removal of ICMP connections so you can receive important ICMP errors. We added the following command: **timeout icmp-error**
Default idle timeout for TCP state bypass	9.10(1)	The default idle timeout for TCP state bypass connections is now 2 minutes instead of 1 hour.
Initiator and responder information for Dead Connection Detection (DCD), and DCD support in a cluster.	9.13(1)	If you enable Dead Connection Detection (DCD), you can use the **show conn detail** command to get information about the initiator and responder. Dead Connection Detection allows you to maintain an inactive connection, and the **show conn** output tells you how often the endpoints have been probed. In addition, DCD is now supported in a cluster. New/Modified commands: **show conn** (output only).
Configure the maximum segment size (MSS) for embryonic connections.	9.16(1)	You can configure a service policy to set the server maximum segment size (MSS) for SYN-cookie generation for embryonic connections upon reaching the embryonic connections limit. This is meaningful for service policies where you are also setting embryonic connection maximums. New or changed commands: **set connection syn-cookie-mss**.

Feature Name	Platform Releases	Description
Configure the SIP Call Init idle global timeout	9.16(x)	You can configure the idle time after which a SIP session in Call_Init call state (after the initial invite message is received) will be deleted, i.e. when no response is received, between 0:0:1 and 1193:0:0. The default is 32 seconds (0:00:32).
IPsec flow offload.	9.18(1)	On the Secure Firewall 3100, IPsec flows are offloaded by default. After the initial setup of an IPsec site-to-site VPN or remote access VPN security association (SA), IPsec connections are offloaded to the field-programmable gate array (FPGA) in the device, which should improve device performance. We added the following commands: **clear flow-offload-ipsec**, **flow-offload-ipsec**, **show flow-offload-ipsec**
DTLS Crypto Acceleration	9.22(1)	Cisco Secure Firewall 4200 and 3100 series support DTLS cryptographic acceleration. The hardware performs DTLS encryption and decryption, and improves the throughput of the DTLS-encrypted and DTLS-decrypted traffic. The hardware also performs optimization of the egress-encrypted packets to improve latency. New/Modified commands: **flow-offload-dtls**, **flow-offload-dtls egress-optimization**
Flow offload is enabled by default for the Secure Firewall 3100/4200	9.23(1)	Flow offload is now enabled by default. Added/modified commands: **flow-offload enable**.
Cluster redirect: flow offload support for the Secure Firewall 4200 asymmetric cluster traffic	9.23(1)	For asymmetric flows, cluster redirect lets the forwarding node offload flows to hardware. This feature is enabled by default. When traffic for an existing flow is sent to a different node, then that traffic is redirected to the owner node over the cluster control link. Because asymmetric flows can create a lot of traffic on the cluster control link, letting the forwarder offload these flows can improve performance. Added/modified commands: **flow-offload cluster-redirect**, **show conn**, **show flow-offload flow**, **show flow-offload flow protocol**,**show flow-offload info**.
IPsec flow offload for traffic on the cluster control link on the Secure Firewall 4200 in distributed site-to-site VPN mode	9.23(1)	For asymmetric flows in distributed site-to-site VPN mode, IPsec flow offload now lets the flow owner decrypt IPsec traffic in hardware that was forwarded over the cluster control link. This feature is not configurable and is always available when you enable IPsec flow offload. Added/modified commands: **flow-offload-ipsec**, **show crypto ipsec sa detail**.



CHAPTER 17

Quality of Service

Have you ever participated in a long-distance phone call that involved a satellite connection? The conversation might be interrupted with brief, but perceptible, gaps at odd intervals. Those gaps are the time, called the latency, between the arrival of packets being transmitted over the network. Some network traffic, such as voice and video, cannot tolerate long latency times. Quality of service (QoS) is a feature that lets you give priority to critical traffic, prevent bandwidth hogging, and manage network bottlenecks to prevent packet drops.

The following topics describe how to apply QoS policies.

- About QoS, on page 411
- Guidelines for QoS, on page 413
- Configure QoS, on page 413
- Monitor QoS, on page 419
- Configuration Examples for Priority Queuing and Policing, on page 420
- History for QoS, on page 422

About QoS

You should consider that in an ever-changing network environment, QoS is not a one-time deployment, but an ongoing, essential part of network design.

This section describes the QoS features available on the ASA.

Supported QoS Features

The ASA supports the following QoS features:

- Policing—To prevent classified traffic from hogging the network bandwidth, you can limit the maximum bandwidth used per class. See Policing, on page 412 for more information.
- Priority queuing—For critical traffic that cannot tolerate latency, such as Voice over IP (VoIP), you can identify traffic for Low Latency Queuing (LLQ) so that it is always transmitted ahead of other traffic. See Priority Queuing, on page 412.

What is a Token Bucket?

A token bucket is used to manage a device that regulates the data in a flow, for example, a traffic policer. A token bucket itself has no discard or priority policy. Rather, a token bucket discards tokens and leaves to the flow the problem of managing its transmission queue if the flow overdrives the regulator.

A token bucket is a formal definition of a rate of transfer. It has three components: a burst size, an average rate, and a time interval. Although the average rate is generally represented as bits per second, any two values may be derived from the third by the relation shown as follows:

average rate = burst size / time interval

Here are some definitions of these terms:

- Average rate—Also called the committed information rate (CIR), it specifies how much data can be sent or forwarded per unit time on average.
- Burst size—Also called the Committed Burst (Bc) size, it specifies in bytes per burst how much traffic can be sent within a given unit of time to not create scheduling concerns.
- Time interval—Also called the measurement interval, it specifies the time quantum in seconds per burst.

In the token bucket metaphor, tokens are put into the bucket at a certain rate. The bucket itself has a specified capacity. If the bucket fills to capacity, newly arriving tokens are discarded. Each token is permission for the source to send a certain number of bits into the network. To send a packet, the regulator must remove from the bucket a number of tokens equal in representation to the packet size.

If not enough tokens are in the bucket to send a packet, the packet waits until the packet is discarded or marked down. If the bucket is already full of tokens, incoming tokens overflow and are not available to future packets. Thus, at any time, the largest burst a source can send into the network is roughly proportional to the size of the bucket.

Policing

Policing is a way of ensuring that no traffic exceeds the maximum rate (in bits/second) that you configure, thus ensuring that no one traffic class can take over the entire resource. When traffic exceeds the maximum rate, the ASA drops the excess traffic. Policing also sets the largest single burst of traffic allowed.

Priority Queuing

LLQ priority queuing lets you prioritize certain traffic flows (such as latency-sensitive traffic like voice and video) ahead of other traffic. Priority queuing uses an LLQ priority queue on an interface (see Configure the Priority Queue for an Interface, on page 415), while all other traffic goes into the “best effort” queue. Because queues are not of infinite size, they can fill and overflow. When a queue is full, any additional packets cannot get into the queue and are dropped. This is called *tail drop*. To avoid having the queue fill up, you can increase the queue buffer size. You can also fine-tune the maximum number of packets allowed into the transmit queue. These options let you control the latency and robustness of the priority queuing. Packets in the LLQ queue are always transmitted before packets in the best effort queue.

How QoS Features Interact

You can configure each of the QoS features alone if desired for the ASA. Often, though, you configure multiple QoS features on the ASA so you can prioritize some traffic, for example, and prevent other traffic from causing bandwidth problems. You can configure:

Priority queuing (for specific traffic) + Policing (for the rest of the traffic).

You cannot configure priority queuing and policing for the same set of traffic.

DSCP (DiffServ) Preservation

DSCP (DiffServ) markings are preserved on all traffic passing through the ASA. The ASA does not locally mark/remark any classified traffic. For example, you could key off the Expedited Forwarding (EF) DSCP bits of every packet to determine if it requires “priority” handling and have the ASA direct those packets to the LLQ.

Guidelines for QoS

Context Mode Guidelines

Supported in single context mode only. Does not support multiple context mode.

Firewall Mode Guidelines

Supported in routed firewall mode only. Does not support transparent firewall mode.

IPv6 Guidelines

Does not support IPv6.

Additional Guidelines and Limitations

- QoS is applied unidirectionally; only traffic that enters (or exits, depending on the QoS feature) the interface to which you apply the policy map is affected.
- For priority traffic, you cannot use the **class-default** class map.
- For priority queuing, the priority queue must be configured for a physical interface.
- For policing, to-the-box traffic is not supported.
- For policing, traffic to and from a VPN tunnel bypasses interface policing.
- For policing, when you match a tunnel group class map, only outbound policing is supported.

Configure QoS

Use the following sequence to implement QoS on the ASA.

Procedure

Step 1 Determine the Queue and TX Ring Limits for a Priority Queue, on page 414.
Step 2 Configure the Priority Queue for an Interface, on page 415.
Step 3 Configure a Service Rule for Priority Queuing and Policing, on page 416.

Determine the Queue and TX Ring Limits for a Priority Queue

Use the following worksheets to determine the priority queue and TX ring limits.

Queue Limit Worksheet

The following worksheet shows how to calculate the priority queue size. Because queues are not of infinite size, they can fill and overflow. When a queue is full, any additional packets cannot get into the queue and are dropped (called *tail drop*). To avoid having the queue fill up, you can adjust the queue buffer size according to Configure the Priority Queue for an Interface, on page 415.

Tips on the worksheet:

- Outbound bandwidth—For example, DSL might have an uplink speed of 768 Kbps. Check with your provider.
- Average packet size—Determine this value from a codec or sampling size. For example, for VoIP over VPN, you might use 160 bytes. We recommend 256 bytes if you do not know what size to use.
- Delay—The delay depends on your application. For example, the recommended maximum delay for VoIP is 200 ms. We recommend 500 ms if you do not know what delay to use.

Table 13: Queue Limit Worksheet

1	_____ *Outbound bandwidth (Mbps or Kbps)*	Mbps	x	**125**	=	_____ *# of bytes/ms*		
		Kbps	x	**.125**	=	_____ *# of bytes/ms*		
2	_____ *# of bytes/ms from Step 1*		÷	_____ *Average packet size (bytes)*	x	_____ *Delay (ms)*	=	_____ *Queue limit (# of packets)*

TX Ring Limit Worksheet

The following worksheet shows how to calculate the TX ring limit. This limit determines the maximum number of packets allowed into the Ethernet transmit driver before the driver pushes back to the queues on the interface to let them buffer packets until the congestion clears. This setting guarantees that the hardware-based transmit ring imposes a limited amount of extra latency for a high-priority packet.

Tips on the worksheet:

- Outbound bandwidth—For example, DSL might have an uplink speed of 768 Kbps. Check with your provider.
- Maximum packet size—Typically, the maximum size is 1538 bytes, or 1542 bytes for tagged Ethernet. If you allow jumbo frames (if supported for your platform), then the packet size might be larger.
- Delay—The delay depends on your application. For example, to control jitter for VoIP, you should use 20 ms.

Table 14: TX Ring Limit Worksheet

1	_____ *Outbound bandwidth (Mbps or Kbps)*	Mbps	x	**125**	=	_____ *# of bytes/ms*		
		Kbps	x	**0.125**	=	_____ *# of bytes/ms*		
2	_____ *# of bytes/ms from Step 1*		÷	_____ *Maximum packet size (bytes)*	x	_____ *Delay (ms)*	=	_____ *TX ring limit (# of packets)*

Configure the Priority Queue for an Interface

If you enable priority queuing for traffic on a physical interface, then you need to also create the priority queue on each interface. Each physical interface uses two queues: one for priority traffic, and the other for all other traffic. For the other traffic, you can optionally configure policing.

Procedure

Step 1 Create the priority queue for the interface.

priority-queue *interface_name*

Example:

```
hostname(config)# priority-queue inside
```

The *interface_name* argument specifies the physical interface name on which you want to enable the priority queue.

Step 2 Change the size of the priority queues.

queue-limit *number_of_packets*

The default queue limit is 1024 packets. Because queues are not of infinite size, they can fill and overflow. When a queue is full, any additional packets cannot get into the queue and are dropped (called *tail drop*). To avoid having the queue fill up, you can use the **queue-limit** command to increase the queue buffer size.

The upper limit of the range of values for the **queue-limit** command is determined dynamically at run time. To view this limit, enter **queue-limit ?** on the command line. The key determinants are the memory needed to support the queues and the memory available on the device.

The **queue-limit** that you specify affects both the higher priority low-latency queue and the best effort queue.

Example:

```
hostname(config-priority-queue)# queue-limit 260
```

Step 3 Specify the depth of the priority queues.

tx-ring-limit *number_of_packets*

The default tx-ring-limit is 511 packets. This command sets the maximum number of low-latency or normal priority packets allowed into the Ethernet transmit driver before the driver pushes back to the queues on the interface to let them buffer packets until the congestion clears. This setting guarantees that the hardware-based transmit ring imposes a limited amount of extra latency for a high-priority packet.

The upper limit of the range of values for the **tx-ring-limit** command is determined dynamically at run time. To view this limit, enter **tx-ring-limit ?** on the command line. The key determinants are the memory needed to support the queues and the memory available on the device.

The **tx-ring-limit** that you specify affects both the higher priority low-latency queue and the best-effort queue.

Example:

```
hostname(config-priority-queue)# tx-ring-limit 3
```

Examples

The following example establishes a priority queue on interface “outside” (the GigabitEthernet0/1 interface), with the default queue-limit and tx-ring-limit:

```
hostname(config)# priority-queue outside
```

The following example establishes a priority queue on the interface “outside” (the GigabitEthernet0/1 interface), sets the queue-limit to 260 packets, and sets the tx-ring-limit to 3:

```
hostname(config)# priority-queue outside
hostname(config-priority-queue)# queue-limit 260
hostname(config-priority-queue)# tx-ring-limit 3
```

Configure a Service Rule for Priority Queuing and Policing

You can configure priority queuing and policing for different class maps within the same policy map. See How QoS Features Interact, on page 413 for information about valid QoS configurations.

Before you begin

- You cannot use the **class-default** class map for priority traffic.
- For policing, to-the-box traffic is not supported.
- For policing, traffic to and from a VPN tunnel bypasses interface policing.
- For policing, when you match a tunnel group class map, only outbound policing is supported.
- For priority traffic, identify only latency-sensitive traffic.
- For policing traffic, you can choose to police all other traffic, or you can limit the traffic to certain types.

Procedure

Step 1 Create an L3/L4 class map to identify the traffic for which you want to perform priority queuing.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map priority_traffic
hostname(config-cmap)# match access-list priority
```

See Create a Layer 3/4 Class Map for Through Traffic, on page 234 for more information.

Step 2 Create an L3/L4 class map to identify the traffic for which you want to perform priority policing.

```
class-map name
match parameter
```

Example:

```
hostname(config)# class-map policing_traffic
hostname(config-cmap)# match access-list policing
```

Tip
If you use an ACL for traffic matching, policing is applied in the direction specified in the ACL only. That is, traffic going from the source to the destination is policed, but not the reverse.

Step 3 Add or edit a policy map: **policy-map** *name*

Example:

```
hostname(config)# policy-map QoS_policy
```

Step 4 Identify the class map you created for prioritized traffic and configure priority queuing for the class.

```
class priority_map_name
```

priority

Example:

```
hostname(config-pmap)# class priority_class
hostname(config-pmap-c)# priority
```

Step 5 Identify the class map you created for policed traffic: **class** *name*

Example:

```
hostname(config-pmap)# class policing_class
```

Step 6 Configure policing for the class.

police {**output** | **input**} *conform-rate* [*conform-burst*] [**conform-action** [**drop** | **transmit**]] [**exceed-action** [**drop** | **transmit**]]

The options are:

- **output**—Enables policing of traffic flowing in the output direction.
- **input**—Enables policing of traffic flowing in the input direction.
- *conform-rate*—Sets the rate limit for this traffic class, from 8000 and 2000000000 bits per second. For example, to limit traffic to 5Mbps, enter 5000000.
- *conform-burst*—(Optional.) Specifies the maximum number of instantaneous bytes allowed in a sustained burst before throttling to the conforming rate value, between 1000 and 512000000 bytes. If you omit the variable, the burst size is calculated as 1/32 of the conform-rate in bytes. For example, the burst size for a 5Mbps rate would be 156250.
- **conform-action**—(Optional.) Sets the action to take when the traffic is below the policing rate and burst size. You can drop or transmit the traffic. The default is to transmit the traffic.
- **exceed-action**—(Optional.) Sets the action to take when traffic exceeds the policing rate and burst size. You can drop or transmit packets that exceed the policing rate and burst size. The default is to drop excess packets.

Example:

```
hostname(config-pmap-c)# police output 56000 10500
```

Step 7 Activate the policy map on one or more interfaces.

service-policy *policymap_name* {**global** | **interface** *interface_name*}

Example:

```
hostname(config)# service-policy QoS_policy interface inside
```

The **global** option applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. You can override the global policy on an interface by applying a service policy to that interface. You can only apply one policy map to each interface.

Monitor QoS

The following topics explain how to monitor QoS.

QoS Police Statistics

To view the QoS statistics for traffic policing, use the **show service-policy police** command.

```
hostname# show service-policy police

Global policy:
 Service-policy: global_fw_policy

Interface outside:
 Service-policy: qos
  Class-map: browse
   police Interface outside:
    cir 56000 bps, bc 10500 bytes
    conformed 10065 packets, 12621510 bytes; actions: transmit
    exceeded 499 packets, 625146 bytes; actions: drop
    conformed 5600 bps, exceed 5016 bps
  Class-map: cmap2
   police Interface outside:
    cir 200000 bps, bc 37500 bytes
    conformed 17179 packets, 20614800 bytes; actions: transmit
    exceeded 617 packets, 770718 bytes; actions: drop
    conformed 198785 bps, exceed 2303 bps
```

QoS Priority Statistics

To view statistics for service policies implementing the **priority** command, use the **show service-policy priority** command.

```
hostname# show service-policy priority
Global policy:
 Service-policy: global_fw_policy
Interface outside:
 Service-policy: qos
  Class-map: TG1-voice
   Priority:
    Interface outside: aggregate drop 0, aggregate transmit 9383
```

“Aggregate drop” denotes the aggregated drop in this interface; “aggregate transmit” denotes the aggregated number of transmitted packets in this interface.

QoS Priority Queue Statistics

To display the priority-queue statistics for an interface, use the **show priority-queue statistics** command. The results show the statistics for both the best-effort (BE) queue and the low-latency queue (LLQ). The following example shows the use of the **show priority-queue statistics** command for the interface named test.

```
hostname# show priority-queue statistics test

Priority-Queue Statistics interface test

Queue Type         = BE
Packets Dropped    = 0
Packets Transmit   = 0
Packets Enqueued   = 0
Current Q Length   = 0
Max Q Length       = 0

Queue Type         = LLQ
Packets Dropped    = 0
Packets Transmit   = 0
Packets Enqueued   = 0
Current Q Length   = 0
Max Q Length       = 0
hostname#
```

In this statistical report:

- “Packets Dropped” denotes the overall number of packets that have been dropped in this queue.
- “Packets Transmit” denotes the overall number of packets that have been transmitted in this queue.
- “Packets Enqueued” denotes the overall number of packets that have been queued in this queue.
- “Current Q Length” denotes the current depth of this queue.
- “Max Q Length” denotes the maximum depth that ever occurred in this queue.

Configuration Examples for Priority Queuing and Policing

The following sections provide examples of configuring priority queuing and policing.

Class Map Examples for VPN Traffic

In the following example, the **class-map** command classifies all non-tunneled TCP traffic, using an ACL named tcp_traffic:

```
hostname(config)# access-list tcp_traffic permit tcp any any
hostname(config)# class-map tcp_traffic
hostname(config-cmap)# match access-list tcp_traffic
```

In the following example, other, more specific match criteria are used for classifying traffic for specific, security-related tunnel groups. These specific match criteria stipulate that a match on tunnel-group (in this

case, the previously-defined Tunnel-Group-1) is required as the first match characteristic to classify traffic for a specific tunnel, and it allows for an additional match line to classify the traffic (IP differential services code point, expedited forwarding).

```
hostname(config)# class-map TG1-voice
hostname(config-cmap)# match tunnel-group tunnel-grp1
hostname(config-cmap)# match dscp ef
```

In the following example, the **class-map** command classifies both tunneled and non-tunneled traffic according to the traffic type:

```
hostname(config)# access-list tunneled extended permit ip 10.10.34.0 255.255.255.0
192.168.10.0 255.255.255.0
hostname(config)# access-list non-tunneled extended permit tcp any any
hostname(config)# tunnel-group tunnel-grp1 type IPsec_L2L

hostname(config)# class-map browse
hostname(config-cmap)# description "This class-map matches all non-tunneled tcp traffic."
hostname(config-cmap)# match access-list non-tunneled

hostname(config-cmap)# class-map TG1-voice
hostname(config-cmap)# description "This class-map matches all dscp ef traffic for
tunnel-grp 1."
hostname(config-cmap)# match dscp ef
hostname(config-cmap)# match tunnel-group tunnel-grp1

hostname(config-cmap)# class-map TG1-BestEffort
hostname(config-cmap)# description "This class-map matches all best-effort traffic for
tunnel-grp1."
hostname(config-cmap)# match tunnel-group tunnel-grp1
hostname(config-cmap)# match flow ip destination-address
```

The following example shows a way of policing traffic within a tunnel, provided the classed traffic is not specified as a tunnel, but does go *through* the tunnel. In this example, 192.168.10.10 is the address of the host machine on the private side of the remote tunnel, and the ACL is named “host-over-l2l”. By creating a class-map (named “host-specific”), you can then police the “host-specific” class before the LAN-to-LAN connection polices the tunnel. In this example, the “host-specific” traffic is rate-limited before the tunnel, then the tunnel is rate-limited:

```
hostname(config)# access-list host-over-l2l extended permit ip any host 192.168.10.10
hostname(config)# class-map host-specific
hostname(config-cmap)# match access-list host-over-l2l
```

Priority and Policing Example

The following example builds on the configuration developed in the previous section. As in the previous example, there are two named class-maps: tcp_traffic and TG1-voice.

```
hostname(config)# class-map TG1-best-effort
hostname(config-cmap)# match tunnel-group Tunnel-Group-1
hostname(config-cmap)# match flow ip destination-address
```

Adding a third class map provides a basis for defining a tunneled and non-tunneled QoS policy, as follows, which creates a simple QoS policy for tunneled and non-tunneled traffic, assigning packets of the class TG1-voice to the low latency queue and setting rate limits on the tcp_traffic and TG1-best-effort traffic flows.

In this example, the maximum rate for traffic of the tcp_traffic class is 56,000 bits/second and a maximum burst size of 10,500 bytes per second. For the TC1-BestEffort class, the maximum rate is 200,000 bits/second, with a maximum burst of 37,500 bytes/second. Traffic in the TC1-voice class has no policed maximum speed or burst rate because it belongs to a priority class.

```
hostname(config)# access-list tcp_traffic permit tcp any any
hostname(config)# class-map tcp_traffic
hostname(config-cmap)# match access-list tcp_traffic

hostname(config)# class-map TG1-voice
hostname(config-cmap)# match tunnel-group tunnel-grp1
hostname(config-cmap)# match dscp ef

hostname(config-cmap)# class-map TG1-BestEffort
hostname(config-cmap)# match tunnel-group tunnel-grp1
hostname(config-cmap)# match flow ip destination-address

hostname(config)# policy-map qos
hostname(config-pmap)# class tcp_traffic
hostname(config-pmap-c)# police output 56000 10500

hostname(config-pmap-c)# class TG1-voice
hostname(config-pmap-c)# priority

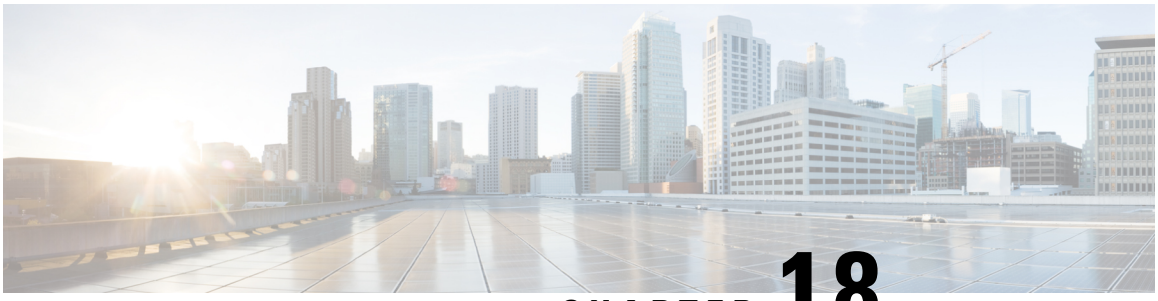
hostname(config-pmap-c)# class TG1-best-effort
hostname(config-pmap-c)# police output 200000 37500

hostname(config-pmap-c)# class class-default
hostname(config-pmap-c)# police output 1000000 37500

hostname(config-pmap-c)# service-policy qos global
```

History for QoS

Feature Name	Platform Releases	Description
Priority queuing and policing	7.0(1)	We introduced QoS priority queuing and policing. We introduced the following commands: **priority-queue**, **queue-limit**, **tx-ring-limit**, **priority, police, show priority-queue statistics, show service-policy police, show service-policy priority, show running-config priority-queue, clear configure priority-queue** .
Shaping and hierarchical priority queuing	7.2(4)/8.0(4)	We introduced QoS shaping and hierarchical priority queuing. We introduced the following commands: **shape**, **show service-policy shape**.
Ten Gigabit Ethernet support for a standard priority queue on the ASA 5585-X	8.2(3)/8.4(1)	We added support for a standard priority queue on Ten Gigabit Ethernet interfaces for the ASA 5585-X.



CHAPTER 18

Threat Detection

The following topics describe how to configure threat detection statistics and scanning threat detection.

- Detecting Threats, on page 423
- Guidelines for Threat Detection, on page 426
- Defaults for Threat Detection, on page 426
- Configure Threat Detection, on page 427
- Monitoring Threat Detection, on page 433
- Examples for Threat Detection, on page 441
- History for Threat Detection, on page 442

Detecting Threats

Threat detection on the ASA provides a front-line defense against attacks. Threat detection works at Layer 3 and 4 to develop a baseline for traffic on the device, analyzing packet drop statistics and accumulating "top" reports based on traffic patterns. In comparison, a module that provides IPS or Next Generation IPS services identifies and mitigates attack vectors up to Layer 7 on traffic the ASA permitted, and cannot see the traffic dropped already by the ASA. Thus, threat detection and IPS can work together to provide a more comprehensive threat defense.

Threat detection consists of the following elements:

- Different levels of statistics gathering for various threats.

  Threat detection statistics can help you manage threats to your ASA; for example, if you enable scanning threat detection, then viewing statistics can help you analyze the threat. You can configure two types of threat detection statistics:

  - Basic threat detection statistics—Includes information about attack activity for the system as a whole. Basic threat detection statistics are enabled by default and have no performance impact. See Basic Threat Detection Statistics, on page 424.
  - Advanced threat detection statistics—Tracks activity at an object level, so the ASA can report activity for individual hosts, ports, protocols, or ACLs. Advanced threat detection statistics can have a major performance impact, depending on the statistics gathered, so only the ACL statistics are enabled by default. See Advanced Threat Detection Statistics, on page 425.

- Scanning threat detection, which determines when a host is performing a scan. You can optionally shun any hosts determined to be a scanning threat. See Scanning Threat Detection, on page 425.

- Threat Detection for VPN Services, which you can use to protect against the following types of VPN attack from IPv4 addresses:
  - Excessive failed authentication attempts to a remote access VPN, for example brute-force username/password scanning.
  - Client initiation attacks, where the attacker starts but does not complete repeated connection attempts to a remote access VPN head-end from a single host.
  - Access attempts to invalid VPN services, that is, services that are for internal use only.

  These attacks, even when unsuccessful in their attempt to gain access, can consume computational resources and in some cases result in Denial of Service. See Configure Threat Detection for VPN Services, on page 431.

Basic Threat Detection Statistics

Using basic threat detection statistics, the ASA monitors the rate of dropped packets and security events due to the following reasons:

- Denial by ACLs.
- Bad packet format (such as invalid-ip-header or invalid-tcp-hdr-length).
- Connection limits exceeded (both system-wide resource limits, and limits set in the configuration).
- DoS attack detected (such as an invalid SPI, Stateful Firewall check failure).
- Basic firewall checks failed. This option is a combined rate that includes all firewall-related packet drops in this list. It does not include non-firewall-related drops such as interface overload, packets failed at application inspection, and scanning attack detected.
- Suspicious ICMP packets detected.
- Packets failed application inspection.
- Interface overload.
- Scanning attack detected. This option monitors scanning attacks; for example, the first TCP packet is not a SYN packet, or the TCP connection failed the 3-way handshake. Full scanning threat detection takes this scanning attack rate information and acts on it by classifying hosts as attackers and automatically shunning them, for example.
- Incomplete session detection such as TCP SYN attack detected or UDP session with no return data attack detected.

When the ASA detects a threat, it immediately sends a system log message (733100). The ASA tracks two types of rates: the average event rate over an interval, and the burst event rate over a shorter burst interval. The burst rate interval is 1/30th of the average rate interval or 10 seconds, whichever is higher. For each received event, the ASA checks the average and burst rate limits; if both rates are exceeded, then the ASA sends two separate system messages, with a maximum of one message for each rate type per burst period.

Basic threat detection affects performance only when there are drops or potential threats; even in this scenario, the performance impact is insignificant.

Advanced Threat Detection Statistics

Advanced threat detection statistics show both allowed and dropped traffic rates for individual objects such as hosts, ports, protocols, or ACLs.

Caution Enabling advanced statistics can affect the ASA performance, depending on the type of statistics enabled. Enabling host statistics affects performance in a significant way; if you have a high traffic load, you might consider enabling this type of statistics temporarily. Port statistics, however, has modest impact.

Scanning Threat Detection

A typical scanning attack consists of a host that tests the accessibility of every IP address in a subnet (by scanning through many hosts in the subnet or sweeping through many ports in a host or subnet). The scanning threat detection feature determines when a host is performing a scan. Unlike IPS scan detection that is based on traffic signatures, ASA threat detection scanning maintains an extensive database that contains host statistics that can be analyzed for scanning activity.

The host database tracks suspicious activity such as connections with no return activity, access of closed service ports, vulnerable TCP behaviors such as non-random IPID, and many more behaviors.

If the scanning threat rate is exceeded, then the ASA sends a syslog message (733101), and optionally shuns the attacker. The ASA tracks two types of rates: the average event rate over an interval, and the burst event rate over a shorter burst interval. The burst event rate is 1/30th of the average rate interval or 10 seconds, whichever is higher. For each event detected that is considered to be part of a scanning attack, the ASA checks the average and burst rate limits. If either rate is exceeded for traffic sent from a host, then that host is considered to be an attacker. If either rate is exceeded for traffic received by a host, then that host is considered to be a target.

The following table lists the default rate limits for scanning threat detection.

Table 15: Default Rate Limits for Scanning Threat Detection

Average Rate	Burst Rate
5 drops/sec over the last 600 seconds.	10 drops/sec over the last 20 second period.
5 drops/sec over the last 3600 seconds.	10 drops/sec over the last 120 second period.

Caution The scanning threat detection feature can affect the ASA performance and memory significantly while it creates and gathers host- and subnet-based data structure and information.

Guidelines for Threat Detection

Security Context Guidelines

Except for advanced threat statistics and VPN services, threat detection is supported in single mode only. In Multiple mode, TCP Intercept statistics are the only statistic supported.

Types of Traffic Monitored

- For statistics, only through-the-box traffic is monitored; to-the-box traffic is not monitored.
- Traffic that is denied by an ACL does not trigger scanning threat detection; only traffic that is allowed through the ASA and that creates a flow is affected by scanning threat detection.
- For VPN services, only to-the-box traffic from IPv4 addresses is monitored.

Defaults for Threat Detection

Basic threat detection statistics are enabled by default.

The following table lists the default settings. You can view all these default settings using the **show running-config all threat-detection** command.

For advanced statistics, by default, statistics for ACLs are enabled.

For VPN service threat detection, all services are disabled by default.

Table 16: Basic Threat Detection Default Settings

	Trigger Settings	
Packet Drop Reason	**Average Rate**	**Burst Rate**
• DoS attack detected • Bad packet format • Connection limits exceeded • Suspicious ICMP packets detected	100 drops/sec over the last 600 seconds.	400 drops/sec over the last 20 second period.
	80 drops/sec over the last 3600 seconds.	320 drops/sec over the last 120 second period.
Scanning attack detected	5 drops/sec over the last 600 seconds.	10 drops/sec over the last 20 second period.
	4 drops/sec over the last 3600 seconds.	8 drops/sec over the last 120 second period.

Packet Drop Reason	Trigger Settings	
	Average Rate	Burst Rate
Incomplete session detected such as TCP SYN attack detected or UDP session with no return data attack detected (combined)	100 drops/sec over the last 600 seconds.	200 drops/sec over the last 20 second period.
	80 drops/sec over the last 3600 seconds.	160 drops/sec over the last 120 second period.
Denial by ACLs	400 drops/sec over the last 600 seconds.	800 drops/sec over the last 20 second period.
	320 drops/sec over the last 3600 seconds.	640 drops/sec over the last 120 second period.
• Basic firewall checks failed • Packets failed application inspection	400 drops/sec over the last 600 seconds.	1600 drops/sec over the last 20 second period.
	320 drops/sec over the last 3600 seconds.	1280 drops/sec over the last 120 second period.
Interface overload	2000 drops/sec over the last 600 seconds.	8000 drops/sec over the last 20 second period.
	1600 drops/sec over the last 3600 seconds.	6400 drops/sec over the last 120 second period.

Configure Threat Detection

Basic threat detection statistics are enabled by default, and might be the only threat detection service that you need. Use the following procedure if you want to implement additional threat detection services.

Procedure

Step 1 Configure Basic Threat Detection Statistics, on page 427.

Basic threat detection statistics include activity that might be related to an attack, such as a DoS attack.

Step 2 Configure Advanced Threat Detection Statistics, on page 428.

Step 3 Configure Scanning Threat Detection, on page 430.

Step 4 Configure Threat Detection for VPN Services, on page 431.

Configure Basic Threat Detection Statistics

Basic threat detection statistics is enabled by default. You can disabled it, or turn it on again if you disable it.

Procedure

Step 1 Enable basic threat detection statistics (if you previously disabled it).

threat-detection basic-threat

Example:

```
hostname(config)# threat-detection basic-threat
```

Basic threat detection is enabled by default. Use **no threat-detection basic-threat** to disable it.

Step 2 (Optional) Change the default settings for one or more type of event.

threat-detection rate {**acl-drop** | **bad-packet-drop** | **conn-limit-drop** | **dos-drop** | **fw-drop** | **icmp-drop** | **inspect-drop** | **interface-drop** | **scanning-threat** | **syn-attack**} **rate-interval** *rate_interval* **average-rate** *av_rate* **burst-rate** *burst_rate*

For a description of each event type, see Basic Threat Detection Statistics, page 8-1.

When you use this command with the **scanning-threat** keyword, it is also used in the scanning threat detection. If you do not configure basic threat detection, you can still use this command with the **scanning-threat** keyword to configure the rate limits for scanning threat detection.

You can configure up to three different rate intervals for each event type.

Example:

```
hostname(config)# threat-detection rate dos-drop rate-interval 600 average-rate 60 burst-rate
 100
```

Configure Advanced Threat Detection Statistics

You can configure the ASA to collect extensive statistics. By default, statistics for ACLs are enabled. To enable other statistics, perform the following steps.

Procedure

Step 1 (Optional) Enable *all* statistics.

threat-detection statistics

To enable only certain statistics, enter this command for each statistic type (shown later in this procedure), and do not also enter the command without any options. You can enter **threat-detection statistics** (without any options) and then customize certain statistics by entering the command with statistics-specific options (for example, **threat-detection statistics host number-of-rate 2**). If you enter **threat-detection statistics** (without any options) and then enter a command for specific statistics, but without any statistic-specific options, then that command has no effect because it is already enabled.

If you enter the **no** form of this command, it removes all **threat-detection statistics** commands, including the **threat-detection statistics access-list** command, which is enabled by default.

Example:

```
hostname(config)# threat-detection statistics
```

Step 2 (Optional) Enable statistics for ACLs (if they were disabled previously).

threat-detection statistics access-list

Statistics for ACLs are enabled by default. ACL statistics are only displayed using the **show threat-detection top access-list** command.

Example:

```
hostname(config)# threat-detection statistics access-list
```

Step 3 (Optional) Configure statistics for hosts (**host** keyword), TCP and UDP ports (**port** keyword), or non-TCP/UDP IP protocols (**protocol** keyword).

threat-detection statistics {**host** | **port** | **protocol**} [**number-of-rate** {**1** | **2** | **3**}]

The **number-of-rate** keyword sets the number of rate intervals maintained for statistics. The default number of rate intervals is **1**, which keeps the memory usage low. To view more rate intervals, set the value to **2** or **3**. For example, if you set the value to **3**, then you view data for the last 1 hour, 8 hours, and 24 hours. If you set this keyword to **1** (the default), then only the shortest rate interval statistics are maintained. If you set the value to **2**, then the two shortest intervals are maintained.

The host statistics accumulate for as long as the host is active and in the scanning threat host database. The host is deleted from the database (and the statistics cleared) after 10 minutes of inactivity.

Example:

```
hostname(config)# threat-detection statistics host number-of-rate 2
hostname(config)# threat-detection statistics port number-of-rate 2
hostname(config)# threat-detection statistics protocol number-of-rate 3
```

Step 4 (Optional) Configure statistics for attacks intercepted by TCP Intercept.

threat-detection statistics tcp-intercept [**rate-interval** *minutes*] [**burst-rate** *attacks_per_sec*] [**average-rate** *attacks_per_sec*]

Where:

- **rate-interval** sets the size of the history monitoring window, between 1 and 1440 minutes. The default is 30 minutes. During this interval, the ASA samples the number of attacks 30 times.
- **burst-rate** sets the threshold for syslog message generation, between 25 and 2147483647. The default is 400 per second. When the burst rate is exceeded, syslog message 733104 is generated.
- **average-rate** sets the average rate threshold for syslog message generation, between 25 and 2147483647. The default is 200 per second. When the average rate is exceeded, syslog message 733105 is generated.

To enable TCP Intercept, see Protect Servers from a SYN Flood DoS Attack (TCP Intercept), on page 381.

Note
This command is available in multiple context mode, unlike the other threat-detection commands.

Example:

```
hostname(config)# threat-detection statistics tcp-intercept rate-interval 60
burst-rate 800 average-rate 600
```

Configure Scanning Threat Detection

You can configure scanning threat detection to identify attackers and optionally shun them.

You can configure the ASA to send system log messages about an attacker or you can automatically shun the host. By default, the system log message 730101 is generated when a host is identified as an attacker. Be sure to except addresses from shunning when you expect a lot of messages from the host. For example, if you have enabled PIM multicast, exempt the PIM routers or PIM messages will be dropped.

Procedure

Step 1 Enable scanning threat detection.

threat-detection scanning-threat [**shun** [**except** {**ip-address** *ip_address mask* | **object-group** *network_object_group_id*}]]

By default, the system log message 733101 is generated when a host is identified as an attacker. Enter this command multiple times to identify multiple IP addresses or network object groups to exempt from shunning.

Example:

```
hostname(config)# threat-detection scanning-threat shun except
ip-address 10.1.1.0 255.255.255.0
```

Step 2 (Optional) Set the duration of the shun for attacking hosts.

threat-detection scanning-threat shun duration *seconds*

Example:

```
hostname(config)# threat-detection scanning-threat shun duration 2000
```

Step 3 (Optional) Change the default event limit for when the ASA identifies a host as an attacker or as a target.

threat-detection rate scanning-threat rate-interval *rate_interval* **average-rate** *av_rate* **burst-rate** *burst_rate*

If you already configured this command as part of the basic threat detection configuration, then those settings are shared with the scanning threat detection feature; you cannot configure separate rates for basic and scanning threat detection. If you do not set the rates using this command, the default values are used for both the scanning threat detection feature and the basic threat detection feature. You can configure up to three different rate intervals by entering separate commands.

Example:

```
hostname(config)# threat-detection rate scanning-threat rate-interval 1200
average-rate 10 burst-rate 20

hostname(config)# threat-detection rate scanning-threat rate-interval 2400
average-rate 10 burst-rate 20
```

Configure Threat Detection for VPN Services

You can enable threat detection for VPN services to help prevent denial of service (DoS) attacks from IPv4 addresses. There are separate services available for the following types of attack:

- Remote access VPN login authentication. By repeatedly starting login attempts in a password-spray attack, the attacker can consume resources used for authentication attempts, thus preventing real users from logging into the VPN.
- Client initiation attacks, where the attacker starts but does not complete repeated connection attempts to a remote access VPN head-end from a single host. Like the password-spray attack, this attack can consume resources and prevent valid users from connecting to the VPN.
- Attempts to connect to an invalid VPN service, that is, services that are for internal use only. An IP address that attempts this connection is immediately shunned.

When you enable these services, the system automatically shuns hosts that exceed thresholds to prevent further attempts. You can manually remove the shun using the **no shun** command for the address.

To manually reset the counters for the services to 0, use the **clear threat-detection service** command.

Before you begin

When deciding on appropriate hold-down and threshold values, consider the use of NAT in your environment. If you use PAT, so that many requests can come from the same IP address, then you should consider higher values for the authentication failure and client initiation services, to ensure valid users have enough time to complete their connections. For example, a hotel, where many customers might try connecting within very short time periods.

Procedure

Step 1 Enable threat detection for remote access VPN authentication failures.

threat-detection service remote-access-authentication hold-down *minutes* **threshold** *count*

Where:

- **hold-down** *minutes* defines the hold-down period from the last failure. The threshold count of consecutive failures must be met within the hold-down period of the previous failure to trigger a shun for the attacker's IPv4 address. For example, if the hold-down period is 10 minutes and the threshold is 20, and if there are 20 consecutive authentication failures from a single IPv4 address, and if the timespan between any

two consecutive failures does not exceed 10 minutes, then the source IPv4 address will be shunned. You can specify a time between 1 and 1440 minutes.

- **threshold** *count* defines the number of failed attempts that must occur within the hold-down period to trigger the shun. You can specify a threshold between 1 and 100.

To disable the service, use the following command:

no threat-detection service remote-access-authentication

Example:

The following example sets a metric of 10 failures within 20 minutes.

```
ciscoasa(config)# threat-detection service remote-access-authentication
hold-down 10 threshold 20
```

Step 2 Enable threat detection for remote access VPN client initiations.

threat-detection service remote-access-client-initiations hold-down *minutes* **threshold** *count*

Where:

- **hold-down** *minutes* defines the hold-down period from the last initiation. The threshold count of consecutive initiations must be met within the hold-down period of the previous initiation to trigger a shun for the client's IPv4 address. For example, if the hold-down period is 10 minutes and the threshold is 20, and if there are 20 consecutive initiations from a single IPv4 address, and if the timespan between any two consecutive initiations does not exceed 10 minutes, then the source IPv4 address will be shunned. You can specify a time between 1 and 1440 minutes.
- **threshold** *count* defines the number of initiations that must occur within the hold-down period to trigger the shun. You can specify a threshold between 5 and 100.

To disable the service, use the following command:

no threat-detection service remote-access-client-initiations

Example:

The following example sets a metric of 10 initiations within 20 minutes.

```
ciscoasa(config)# threat-detection service remote-access-client-initiations
hold-down 10 threshold 20
```

Step 3 Enable threat detection for attempts to connect to invalid VPN services.

threat-detection service invalid-vpn-access

To disable the service, use the following command:

no threat-detection service invalid-vpn-access

Example:

The following example enables the Invalid VPN Access service.

```
ciscoasa(config)# threat-detection service invalid-vpn-access
```

Monitoring Threat Detection

The following topics explain how to monitor threat detection and view traffic statistics.

Monitoring Basic Threat Detection Statistics

To display basic threat detection statistics, use the following command:

show threat-detection rate [**min-display-rate** *min_display_rate*] [**acl-drop** | **bad-packet-drop** | **conn-limit-drop** | **dos-drop** | **fw-drop** | **icmp-drop** | **inspect-drop** | **interface-drop** | **scanning-threat** | **syn-attack**]

The **min-display-rate** *min_display_rate* argument limits the display to statistics that exceed the minimum display rate in events per second. You can set the *min_display_rate* between 0 and 2147483647.

The other arguments let you limit the display to specific categories. For a description of each event type, see Basic Threat Detection Statistics, on page 424.

The output shows the average rate in events/sec over two fixed time periods: the last 10 minutes and the last 1 hour. It also shows: the current burst rate in events/sec over the last completed burst interval, which is 1/30th of the average rate interval or 10 seconds, whichever is larger; the number of times the rates were exceeded (triggered); and the total number of events over the time periods.

The ASA stores the count at the end of each burst period, for a total of 30 completed burst intervals. The unfinished burst interval presently occurring is not included in the average rate. For example, if the average rate interval is 20 minutes, then the burst interval is 20 seconds. If the last burst interval was from 3:00:00 to 3:00:20, and you use the **show** command at 3:00:25, then the last 5 seconds are not included in the output.

The only exception to this rule is if the number of events in the unfinished burst interval already exceeds the number of events in the oldest burst interval (#1 of 30) when calculating the total events. In that case, the ASA calculates the total events as the last 29 complete intervals, plus the events so far in the unfinished burst interval. This exception lets you monitor a large increase in events in real time.

You can clear statistics using the **clear threat-detection rate** command.

The following is sample output from the **show threat-detection rate** command:

```
hostname# show threat-detection rate

                        Average(eps)    Current(eps) Trigger         Total events
  10-min ACL  drop:                0              0       0                   16
  1-hour ACL  drop:                0              0       0                  112
  1-hour SYN attck:                5              0       2                21438
  10-min  Scanning:                0              0      29                  193
  1-hour  Scanning:              106              0      10               384776
  1-hour Bad  pkts:               76              0       2               274690
  10-min  Firewall:                0              0       3                   22
  1-hour  Firewall:               76              0       2               274844
  10-min DoS attck:                0              0       0                    6
  1-hour DoS attck:                0              0       0                   42
  10-min Interface:                0              0       0                  204
```

```
1-hour Interface:                88              0       0               318225
```

Monitoring Advanced Threat Detection Statistics

To monitor advanced threat detection statistics, use the commands shown in the following table. The display output shows the following:

- The average rate in events/sec over fixed time periods.
- The current burst rate in events/sec over the last completed burst interval, which is 1/30th of the average rate interval or 10 seconds, whichever is larger
- The number of times the rates were exceeded (for dropped traffic statistics only)
- The total number of events over the fixed time periods.

The ASA stores the count at the end of each burst period, for a total of 30 completed burst intervals. The unfinished burst interval presently occurring is not included in the average rate. For example, if the average rate interval is 20 minutes, then the burst interval is 20 seconds. If the last burst interval was from 3:00:00 to 3:00:20, and you use the **show** command at 3:00:25, then the last 5 seconds are not included in the output.

The only exception to this rule is if the number of events in the unfinished burst interval already exceeds the number of events in the oldest burst interval (#1 of 30) when calculating the total events. In that case, the ASA calculates the total events as the last 29 complete intervals, plus the events so far in the unfinished burst interval. This exception lets you monitor a large increase in events in real time.

Command	Purpose
show threat-detection statistics [**min-display-rate** *min_display_rate*] **top** [[**access-list** | **host** | **port-protocol**] [**rate-1** | **rate-2** | **rate-3**] | **tcp-intercept** [**all**] **detail**]]	Displays the top 10 statistics. If you do not enter any options, the top 10 statistics are shown for all categories. The **min-display-rate** *min_display_rate* argument limits the display to statistics that exceed the minimum display rate in events per second. You can set the *min_display_rate* between 0 and 2147483647. Following rows explain optional keywords.
show threat-detection statistics [**min-display-rate** *min_display_rate*] **top access-list** [**rate-1** | **rate-2** | **rate-3**]	To view the top 10 ACEs that match packets, including both permit and deny ACEs, use the **access-list** keyword. Permitted and denied traffic are not differentiated in this display. If you enable basic threat detection using the **threat-detection basic-threat** command, you can track ACL denies using the **show threat-detection rate acl-drop** command. The **rate-1** keyword shows the statistics for the smallest fixed rate intervals available in the display; **rate-2** shows the next largest rate interval; and **rate-3**, if you have three intervals defined, shows the largest rate interval. For example, the display shows statistics for the last 1 hour, 8 hours, and 24 hours. If you set the **rate-1** keyword, the ASA shows only the 1 hour time interval.
show threat-detection statistics [**min-display-rate** *min_display_rate*] **top host** [**rate-1** | **rate-2** | **rate-3**]	To view only host statistics, use the **host** keyword. **Note**: Due to the threat detection algorithm, an interface used as a combination failover and state link could appear in the top 10 hosts; this is expected behavior, and you can ignore this IP address in the display.

Command	Purpose
show threat-detection statistics [**min-display-rate** *min_display_rate*] **top port-protocol** [**rate-1** | **rate-2** | **rate-3**]	To view statistics for ports and protocols, use the **port-protocol** keyword. The **port-protocol** keyword shows statistics for both ports and protocols (both must be enabled for the display), and shows the combined statistics of TCP/UDP port and IP protocol types. TCP (protocol 6) and UDP (protocol 17) are not included in the display for IP protocols; TCP and UDP ports are, however, included in the display for ports. If you only enable statistics for one of these types, port or protocol, then you will only view the enabled statistics.
show threat-detection statistics [**min-display-rate** *min_display_rate*] **top tcp-intercept** [**all**] **detail**]]	To view TCP Intercept statistics, use the **tcp-intercept** keyword. The display includes the top 10 protected servers under attack. The **all** keyword shows the history data of all the traced servers. The **detail** keyword shows history sampling data. The ASA samples the number of attacks 30 times during the rate interval, so for the default 30 minute period, statistics are collected every 60 seconds.
show threat-detection statistics [**min-display-rate** *min_display_rate*] **host** [*ip_address* [*mask*]]	Displays statistics for all hosts or for a specific host or subnet.
show threat-detection statistics [**min-display-rate** *min_display_rate*] **port** [*start_port*[**-***end_port*]]	Displays statistics for all ports or for a specific port or range of ports.
show threat-detection statistics [**min-display-rate** *min_display_rate*] **protocol** [*protocol_number* | *protocol*]	Displays statistics for all IP protocols or for a specific protocol. The *protocol_number* argument is an integer between 0 and 255. The *protocol* argument can be one of **ah**, **eigrp**, **esp**, **gre**, **icmp**, **icmp6**, **igmp**, **igrp**, **ip**, **ipinip**, **ipsec**, **nos**, **ospf**, **pcp**, **pim**, **pptp**, **snp**, **tcp**, **udp**.

Evaluating Host Threat Detection Statistics

The following is sample output from the **show threat-detection statistics host** command:

```
hostname# show threat-detection statistics host

                        Average(eps)    Current(eps) Trigger         Total events
Host:10.0.0.1: tot-ses:289235 act-ses:22571 fw-drop:0 insp-drop:0 null-ses:21438 bad-acc:0
  1-hour Sent byte:               2938               0       0             10580308
  8-hour Sent byte:                367               0       0             10580308
 24-hour Sent byte:                122               0       0             10580308
  1-hour Sent pkts:                 28               0       0               104043
  8-hour Sent pkts:                  3               0       0               104043
 24-hour Sent pkts:                  1               0       0               104043
  20-min Sent drop:                  9               0       1                10851
  1-hour Sent drop:                  3               0       1                10851
  1-hour Recv byte:               2697               0       0              9712670
  8-hour Recv byte:                337               0       0              9712670
 24-hour Recv byte:                112               0       0              9712670
  1-hour Recv pkts:                 29               0       0               104846
  8-hour Recv pkts:                  3               0       0               104846
 24-hour Recv pkts:                  1               0       0               104846
  20-min Recv drop:                 42               0       3                50567
  1-hour Recv drop:                 14               0       1                50567
Host:10.0.0.0: tot-ses:1 act-ses:0 fw-drop:0 insp-drop:0 null-ses:0 bad-acc:0
  1-hour Sent byte:                  0               0       0                  614
```

```
  8-hour Sent byte:                0                0        0              614
 24-hour Sent byte:                0                0        0              614
  1-hour Sent pkts:                0                0        0                6
  8-hour Sent pkts:                0                0        0                6
 24-hour Sent pkts:                0                0        0                6
  20-min Sent drop:                0                0        0                4
  1-hour Sent drop:                0                0        0                4
  1-hour Recv byte:                0                0        0              706
  8-hour Recv byte:                0                0        0              706
 24-hour Recv byte:                0                0        0              706
  1-hour Recv pkts:                0                0        0                7
```

The following table explains the output.

Table 17: show threat-detection statistics host

Field	Description
Host	The host IP address.
tot-ses	The total number of sessions for this host since it was added to the database.
act-ses	The total number of active sessions that the host is currently involved in.
fw-drop	The number of firewall drops. Firewall drops is a combined rate that includes all firewall-related packet drops tracked in basic threat detection, including ACL denials, bad packets, exceeded connection limits, DoS attack packets, suspicious ICMP packets, TCP SYN attack packets, and UDP session with no return data attack packets. It does not include non-firewall-related drops such as interface overload, packets failed at application inspection, and scanning attack detected.
insp-drop	The number of packets dropped because they failed application inspection.
null-ses	The number of null sessions, which are TCP SYN sessions that did not complete within the 3-second timeout, and UDP sessions that did not have any data sent by its server 3 seconds after the session starts.
bad-acc	The number of bad access attempts to host ports that are in a closed state. When a port is determined to be in a null session (see the null-ses field description), the port state of the host is set to HOST_PORT_CLOSE. Any client accessing the port of the host is immediately classified as a bad access without the need to wait for a timeout.

Field	Description
Average(eps)	The average rate in events/sec over each time period. The ASA stores the count at the end of each burst period, for a total of 30 completed burst intervals. The unfinished burst interval presently occurring is not included in the average rate. For example, if the average rate interval is 20 minutes, then the burst interval is 20 seconds. If the last burst interval was from 3:00:00 to 3:00:20, and you use the **show** command at 3:00:25, then the last 5 seconds are not included in the output. The only exception to this rule is if the number of events in the unfinished burst interval already exceeds the number of events in the oldest burst interval (#1 of 30) when calculating the total events. In that case, the ASA calculates the total events as the last 29 complete intervals, plus the events so far in the unfinished burst interval. This exception lets you monitor a large increase in events in real time.
Current(eps)	The current burst rate in events/sec over the last completed burst interval, which is 1/30th of the average rate interval or 10 seconds, whichever is larger. For the example specified in the Average(eps) description, the current rate is the rate from 3:19:30 to 3:20:00
Trigger	The number of times the dropped packet rate limits were exceeded. For valid traffic identified in the sent and received bytes and packets rows, this value is always 0, because there are no rate limits to trigger for valid traffic.
Total events	The total number of events over each rate interval. The unfinished burst interval presently occurring is not included in the total events. The only exception to this rule is if the number of events in the unfinished burst interval already exceeds the number of events in the oldest burst interval (#1 of 30) when calculating the total events. In that case, the ASA calculates the total events as the last 29 complete intervals, plus the events so far in the unfinished burst interval. This exception lets you monitor a large increase in events in real time.
20-min, 1-hour, 8-hour, and 24-hour	Statistics for these fixed rate intervals. For each interval: • Sent byte—The number of successful bytes sent from the host. • Sent pkts—The number of successful packets sent from the host. • Sent drop—The number of packets sent from the host that were dropped because they were part of a scanning attack. • Recv byte—The number of successful bytes received by the host. • Recv pkts—The number of successful packets received by the host. • Recv drop—the number of packets received by the host that were dropped because they were part of a scanning attack.

Monitoring Scanning Threat Detection Shunned Hosts, Attackers, and Targets

To monitor and manage shunned hosts and attackers and targets for scanning threat detection, use the following commands. These commands are for scanning threat detection only and do not apply to other services.

- **show threat-detection shun**

  Displays the hosts that are currently shunned. For example:

```
hostname# show threat-detection shun

Shunned Host List:
(outside) src-ip=10.0.0.13 255.255.255.255
(inside) src-ip=10.0.0.13 255.255.255.255
```

- **clear threat-detection shun** [*ip_address* [*mask*]]

  Releases a host from being shunned. If you do not specify an IP address, all hosts are cleared from the shun list.

  For example, to release the host at 10.1.1.6, enter the following command:

```
hostname# clear threat-detection shun 10.1.1.6
```

- **show threat-detection scanning-threat** [**attacker** | **target**]

  Displays hosts that the ASA decides are attackers (including hosts on the shun list), and displays the hosts that are the target of an attack. If you do not enter an option, both attackers and target hosts are displayed. For example:

```
hostname# show threat-detection scanning-threat
Latest Target Host & Subnet List:
    192.168.1.0 (l2l)
    192.168.1.249 (l2l)
Latest Attacker Host & Subnet List:
    192.168.10.234 (outside)
    192.168.10.0 (outside)
    192.168.10.2 (outside)
    192.168.10.3 (outside)
    192.168.10.4 (outside)
    192.168.10.5 (outside)
    192.168.10.6 (outside)
    192.168.10.7 (outside)
    192.168.10.8 (outside)
    192.168.10.9 (outside)
```

Monitoring Threat Detection for VPN Services

You can monitor threat detection for VPN services using syslog and show commands, as explained in the following topics.

Syslog Monitoring for Threat Detection VPN Services

You might see the following syslog messages related to these services:

- %ASA-6-733200: Threat-detection Info: *message*

  This message reports general informational events for threat detection.

- %ASA-4-733201: Threat-detection: Service[*service*] Peer[*peer*]: threshold of *threshold-value* was exceeded. Adding shun to interface *interface*. *Additional_message*

  This message shows that the threat detection service shunned an IP address due to suspicious activity for the specified service. The message might contain additional information. For example, for RA VPN client initiation attempts, the additional information might be "SSL (or IKEv2): RA excessive client initiation requests."

  You can see the list of shunned hosts using the **show shun** command. If you know the IP address is not an attacker, you can remove the shun using the **no shun** command.

Show Command Monitoring for Threat Detection for VPN Services

To display statistics for threat detection VPN services, use the following command:

show threat-detection service [*service*] [**entries** | **details**]

You can optionally limit the view to a particular service (**remote-access-authentication**, **remote-access-client-initiations**, or **invalid-vpn-access**). You can limit the view further by adding these parameters:

- **entries**—Display only the entries being tracked. For example, the IP addresses that have had failed authentication attempts.
- **details**—Display both service details and service entries.

Based on selected options, the display output shows the following:

- The name of the service
- The state of the service: enabled or disabled
- The service hold-down setting
- The service threshold setting
- Service action statistics
  - Failed—A failure occurrence when processing the reported occurrence.
  - Blocking—The reported occurrence is within the hold-down period and the threshold was met or exceeded. As a result, the service automatically installed a shun to block the mischievous peer.
  - Recording—The reported occurrence is outside of the hold-down period, or the threshold was met or exceeded. As a result, the service will record the occurrence.
  - Unsupported—The reported occurrence does not currently support automatic shunning.
  - Disabled—An occurrence was reported; but the service has been disabled.

Examples

The following example shows that all services are enabled, and potential attackers are being tracked for the remote-access-authentication service.

```
ciscoasa# show threat-detection service
Service: invalid-vpn-access
    State     : Enabled
    Hold-down : 1 minutes
    Threshold : 1
    Stats:
        failed      :          0
        blocking    :          0
        recording   :          0
        unsupported :          0
        disabled    :          0
    Total entries: 0
Service: remote-access-authentication
    State     : Enabled
    Hold-down : 10 minutes
    Threshold : 20
    Stats:
        failed      :          0
        blocking    :          1
        recording   :          4
        unsupported :          0
        disabled    :          0
    Total entries: 3
Name: remote-access-client-initiations
    State     : Enabled
    Hold-down : 10 minutes
    Threshold : 20
    Stats:
        failed      :          0
        blocking    :          0
        recording   :          0
        unsupported :          0
        disabled    :          0
    Total entries: 0
```

The following is an example of the **show threat-detection service entries** command.

```
ciscoasa# show threat-detection service remote-access-authentication entries
Service: remote-access-authentication
    Total entries: 2

Idx Source              Interface            Count          Age         Hold-down
--- ------------------- -------------------- -------------- ----------- ---------
  1 192.168.100.101/ 32              outside              1         721           0
  2 192.168.100.102/ 32              outside              2         486         114
Total number of IPv4 entries: 2

NOTE: Age is in seconds since last reported. Hold-down is in seconds remaining.
```

The following is an example of the **show threat-detection service details** command.

```
ciscoasa# show threat-detection service remote-access-authentication details
Service: remote-access-authentication
    State     : Enabled
    Hold-down : 10 minutes
    Threshold : 20
    Stats:
        failed      :          0
        blocking    :          1
        recording   :          4
        unsupported :          0
        disabled    :          0
```

```
     Total entries: 2

Idx Source              Interface            Count          Age         Hold-down
--- ------------------- -------------------- -------------- ----------- ---------
  1 192.168.100.101/ 32              outside              1         721           0
  2 192.168.100.102/ 32              outside              2         486         114
Total number of IPv4 entries: 2

NOTE: Age is in seconds since last reported. Hold-down is in seconds remaining.
```

Removing Shuns Applied for VPN Service Violations

You can monitor shuns applied for VPN services, and remove shuns, using the following commands. Note that shuns applied by threat detection for VPN services do not appear in the **show threat-detection shun** command, which applies to scanning threat detection only.

- **show shun** [*ip_address*]

  Shows shunned hosts, including those shunned automatically by threat detection for VPN services, or manually using the **shun** command. You can optionally limit the view to a specified IP address.

- **no shun** *ip_address* [**interface** *if_name*]

  Removes the shun from the specified IP address only. You can optionally specify the interface name for the shun, if the address is shunned on more than one interface and you want to leave the shun in place on some interfaces.

- **clear shun**

  Removes the shun from all IP addresses.

Examples for Threat Detection

The following example configures basic threat detection statistics, and changes the DoS attack rate settings. All advanced threat detection statistics are enabled, with the host statistics number of rate intervals lowered to 2. The TCP Intercept rate interval is also customized. Scanning threat detection is enabled with automatic shunning for all addresses except 10.1.1.0/24. The scanning threat rate intervals are customized.

```
threat-detection basic-threat
threat-detection rate dos-drop rate-interval 600 average-rate 60 burst-rate 100
threat-detection statistics
threat-detection statistics host number-of-rate 2
threat-detection statistics tcp-intercept rate-interval 60 burst-rate 800 average-rate 600
threat-detection scanning-threat shun except ip-address 10.1.1.0 255.255.255.0
threat-detection rate scanning-threat rate-interval 1200 average-rate 10 burst-rate 20
threat-detection rate scanning-threat rate-interval 2400 average-rate 10 burst-rate 20
```

History for Threat Detection

Feature Name	Platform Releases	Description
Basic and advanced threat detection statistics, scanning threat detection	8.0(2)	Basic and advanced threat detection statistics, scanning threat detection was introduced. The following commands were introduced: **threat-detection basic-threat**, **threat-detection rate**, **show threat-detection rate**, **clear threat-detection rate, threat-detection statistics**, **show threat-detection statistics**, **threat-detection scanning-threat**, **threat-detection rate scanning-threat**, **show threat-detection scanning-threat**, **show threat-detection shun**, **clear threat-detection shun**.
Shun duration	8.0(4)/8.1(2)	You can now set the shun duration, The following command was introduced: **threat-detection scanning-threat shun duration**.
TCP Intercept statistics	8.0(4)/8.1(2)	TCP Intercept statistics were introduced. The following commands were modified or introduced: **threat-detection statistics tcp-intercept**, **show threat-detection statistics top tcp-intercept**, **clear threat-detection statistics**.
Customize host statistics rate intervals	8.1(2)	You can now customize the number of rate intervals for which statistics are collected. The default number of rates was changed from 3 to 1. The following command was modified: **threat-detection statistics host number-of-rates**.
Burst rate interval changed to 1/30th of the average rate.	8.2(1)	In earlier releases, the burst rate interval was 1/60th of the average rate. To maximize memory usage, the sampling interval was reduced to 30 times during the average rate.
Customize port and protocol statistics rate intervals	8.3(1)	You can now customize the number of rate intervals for which statistics are collected. The default number of rates was changed from 3 to 1. The following commands were modified: **threat-detection statistics port number-of-rates**, **threat-detection statistics protocol number-of-rates**.
Improved memory usage	8.3(1)	The memory usage for threat detection was improved. The following command was introduced: **show threat-detection memory**.

Feature Name	Platform Releases	Description
Threat Detection for VPN services	9.20(3)	You can configure threat detection for VPN services to protect against the following types of VPN attack from IPv4 addresses: • Excessive failed authentication attempts to a remote access VPN, for example brute-force username/password scanning. • Client initiation attacks, where the attacker starts but does not complete repeated connection attempts to a remote access VPN head-end from a single host. • Access attempts to invalid VPN services, that is, services that are for internal use only. These attacks, even when unsuccessful in their attempt to gain access, can consume computational resources and in some cases result in Denial of Service. The following commands were introduced or changed: **clear threat-detection service**, **show threat-detection service**, **shun**, **threat-detection service**.