



Wide-Area Networking Configuration Guide: Wide-Area Application Services, Cisco IOS Release 15M&T

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 NON-INFRINGEMENT 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.

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/go/trademarks>. 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)

© 2018 Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1

Wide-Area Networking Overview 1

- Finding Feature Information 1
- Frame Relay 1
 - Frame Relay-ATM Internetworking 4
- Switched Multimegabit Data Service 4
- Link Access Procedure - Balanced and X.25 5
- Layer 2 VPN 6
 - Layer 2 Tunneling Protocol Version 3 7
 - L2VPN Pseudowire Redundancy 7
 - L2VPN Interworking 7
 - Layer 2 Local Switching 7
- WAAS Express 7

CHAPTER 2

Configuring WAAS Express 9

- Finding Feature Information 9
- Prerequisites for WAAS Express 9
- Restrictions for WAAS Express 10
- Information About WAAS Express 11
 - WAAS Express Overview 11
 - WAAS Express-Enabled Traffic Optimization Process 13
- Key Services of WAAS Express 14
 - Transport Flow Optimization 14
 - Compression 15
 - Autodiscovery of WAAS Express Devices 16
- Application Acceleration 16
 - CIFS-Express Accelerator 17

HTTP-Express Accelerator	18
SSL-Express Accelerator	20
WAAS Express Application Policies	22
Multiple WAN Links	32
SNMP Support for WAAS Express	32
Troubleshooting Tips	32
How to Configure WAAS Express	33
Configuring WAN Optimization Parameters	33
Defining WAAS Express Policies	34
Defining Class Maps	34
Associating Class Maps with Policy Maps	36
Troubleshooting Tips	38
Enabling WAAS Express	38
Troubleshooting Tips	39
Configuring Upload DRE	39
Configuring CIFS-Express Accelerator	40
Configuring HTTP-Express Accelerator	42
Configuring HTTP Metadata Caching	43
Configuring SSL-Express Accelerator	46
Configuring SNMP Traps for WAAS Express	48
Configuration Examples for WAAS Express	49
Example: Associating Class Maps with Policy Maps	49
Example: Configuring WAAS Express	50
Example: Configuring CIFS-Express Accelerator	51
Example: Configuring HTTP-Express Accelerator	51
Example: Configuring SSL-Express Accelerator	51
Example: Configuring SNMP Traps for WAAS Express	52
Additional References	52
Feature Information for WAAS Express	53

CHAPTER 3

Configuring Cisco IOS NAM PA for WAAS Express	55
Finding Feature Information	55
Restrictions for Cisco IOS NAM PA for WAAS Express	56
Information About Cisco IOS NAM PA for WAAS Express	56

NetFlow Overview	56
MACE Metrics	56
MACE Configuration Plane	56
WAAS Express	57
ART Engine	57
MACE Exporter	58
How to Configure Cisco IOS NAM PA for WAAS Express	59
Enabling MACE on an Interface	59
Enabling MACE on WAAS	62
Configuration Examples for Cisco IOS NAM PA for WAAS Express	64
Example: Enabling MACE on an Interface	64
Example: Enabling MACE on WAAS	65
Additional References	65
Feature Information for Cisco IOS NAM PA for WAAS Express	66

CHAPTER 4

Configuring AVC to Monitor MACE Metrics	67
Finding Feature Information	67
Restrictions for Configuring AVC to Monitor MACE Metrics	67
Information about Configuring AVC to Monitor MACE Metrics	68
New Functionality for MACE Phase 2	68
NetFlow Overview	70
MACE Metrics	70
MACE Configuration Plane	70
WAAS Express	71
ART Engine	71
MACE Exporter	72
How to Configure AVC to Monitor MACE Metrics	72
Configuring MACE for WAAS	72
Configuring MACE for an Interface	76
Additional References	82
Feature Information for Configuring AVC to Monitor MACE Metrics	82



CHAPTER 1

Wide-Area Networking Overview

Cisco IOS software provides a range of wide-area networking capabilities to fit almost every network environment need. Cisco offers cell relay via the Switched Multimegabit Data Service (SMDS), circuit switching via ISDN, packet switching via Frame Relay, and the benefits of both circuit and packet switching via Asynchronous Transfer Mode (ATM). LAN emulation (LANE) provides connectivity between ATM and other LAN types. Cisco's Wide-Area Application Services (WAAS) Express software offers bandwidth optimization and application acceleration capabilities. The *Cisco IOS Wide-Area Networking Configuration Guide* presents a set of general guidelines for configuring the following software components:

This module gives a high-level description of each technology. For specific configuration information, see the appropriate module.

- [Finding Feature Information, on page 1](#)
- [Frame Relay, on page 1](#)
- [Switched Multimegabit Data Service, on page 4](#)
- [Link Access Procedure - Balanced and X.25, on page 5](#)
- [Layer 2 VPN, on page 6](#)
- [WAAS Express, on page 7](#)

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see [Bug Search Tool](#) and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Frame Relay

The Cisco Frame Relay implementation currently supports routing on IP, DECnet, AppleTalk, XNS, Novell IPX, CLNS, Banyan VINES, and transparent bridging.

Although Frame Relay access was originally restricted to leased lines, dialup access is now supported. For more information, for dialer profiles or for legacy dial-on-demand routing (DDR) see the see the module Dial-on-Demand Routing Configuration.

To install software on a new router or access server by downloading software from a central server over an interface that supports Frame Relay, see the module Loading and Maintaining System Images.

To configure access between Systems Network Architecture (SNA) devices over a Frame Relay network, see the module Configuring SNA Frame Relay Access Support.

The Frame Relay software provides the following capabilities:

- Support for the three generally implemented specifications of Frame Relay Local Management Interfaces (LMIs):
 - The Frame Relay Interface joint specification produced by Northern Telecom, Digital Equipment Corporation, StrataCom, and Cisco Systems
 - The ANSI-adopted Frame Relay signal specification, T1.617 Annex D
 - The ITU-T-adopted Frame Relay signal specification, Q.933 Annex A
- Conformity to ITU-T I-series (ISDN) recommendation as I122, "Framework for Additional Packet Mode Bearer Services":
 - The ANSI-adopted Frame Relay encapsulation specification, T1.618
 - The ITU-T-adopted Frame Relay encapsulation specification, Q.922 Annex A
- Conformity to Internet Engineering Task Force (IETF) encapsulation in accordance with RFC 2427, except bridging.
- Support for a keepalive mechanism, a multicast group, and a status message, as follows:
 - The keepalive mechanism provides an exchange of information between the network server and the switch to verify that data is flowing.
 - The multicast mechanism provides the network server with a local data-link connection identifier (DLCI) and a multicast DLCI. This feature is specific to our implementation of the Frame Relay joint specification.
 - The status mechanism provides an ongoing status report on the DLCIs known by the switch.
- Support for both PVCs and SVCs in the same sites and routers.

SVCs allow access through a Frame Relay network by setting up a path to the destination endpoints only when the need arises and tearing down the path when it is no longer needed.

- Support for Frame Relay Traffic Shaping beginning with Cisco IOS Release 11.2. Traffic shaping provides the following:
 - Rate enforcement for individual circuits--The peak rate for outbound traffic can be set to the committed information rate (CIR) or some other user-configurable rate.
 - Dynamic traffic throttling on a per-virtual-circuit basis--When backward explicit congestion notification (BECN) packets indicate congestion on the network, the outbound traffic rate is automatically stepped down; when congestion eases, the outbound traffic rate is stepped up again.
 - Enhanced queueing support on a per-virtual circuit basis--Custom queueing, priority queueing, and weighted fair queueing can be configured for individual virtual circuits.
- Transmission of congestion information from Frame Relay to DECnet Phase IV and CLNS. This mechanism promotes forward explicit congestion notification (FECN) bits from the Frame Relay layer to upper-layer protocols after checking for the FECN bit on the incoming DLCI. Use this Frame Relay congestion information to adjust the sending rates of end hosts. FECN-bit promotion is enabled by default on any interface using Frame Relay encapsulation. No configuration is required.

- Support for Frame Relay Inverse ARP as described in RFC 1293 for the AppleTalk, Banyan VINES, DECnet, IP, and IPX protocols, and for native hello packets for DECnet, CLNP, and Banyan VINES. It allows a router running Frame Relay to discover the protocol address of a device associated with the virtual circuit.
- Support for Frame Relay switching, whereby packets are switched based on the DLCI—a Frame Relay equivalent of a Media Access Control (MAC)-level address. Routers are configured as a hybrid DTE switch or pure Frame Relay DCE access node in the Frame Relay network.

Frame Relay switching is used when all traffic arriving on one DLCI can be sent out on another DLCI to the same next-hop address. In such cases, the Cisco IOS software need not examine the frames individually to discover the destination address, and, as a result, the processing load on the router decreases.

The Cisco implementation of Frame Relay switching provides the following functionality:

- Switching over an IP tunnel
 - Switching over Network-to-Network Interfaces (NNI) to other Frame Relay switches
 - Local serial-to-serial switching
 - Switching over ISDN B channels
 - Traffic shaping on switched PVCs
 - Congestion management on switched PVCs
 - Traffic policing on User-Network Interface (UNI) DCE
 - FRF.12 fragmentation on switched PVCs
- Support for *subinterfaces* associated with a physical interface. The software groups one or more PVCs under separate subinterfaces, which in turn are located under a single physical interface. See the Configuring Frame Relay module.
 - Support for fast-path transparent bridging, as described in RFC 1490, for Frame Relay encapsulated serial and High-Speed Serial Interfaces (HSSIs) on all platforms.
 - Support of the Frame Relay DTE MIB specified in RFC 1315. However, the error table is not implemented. To use the Frame Relay MIB, refer to your MIB publications.
 - Support for Frame Relay fragmentation. Cisco has developed the following three types of Frame Relay fragmentation:
 - End-to-End FRF.12 Fragmentation

FRF.12 fragmentation is defined by the FRF.12 Implementation Agreement. This standard was developed to allow long data frames to be fragmented into smaller pieces (fragments) and interleaved with real-time frames. End-to-end FRF.12 fragmentation is recommended for use on PVCs that share links with other PVCs that are transporting voice and on PVCs transporting Voice over IP (VoIP).

- Frame Relay Fragmentation Using FRF.11 Annex C

When VoFR (FRF.11) and fragmentation are both configured on a PVC, the Frame Relay fragments are sent in the FRF.11 Annex C format. This fragmentation is used when FRF.11 voice traffic is sent on the PVC, and it uses the FRF.11 Annex C format for data.

See the module Configuring Voice over Frame Relay in the *Cisco IOS Voice, Video, and Fax Configuration Guide* for configuration tasks and examples for Frame Relay fragmentation using FRF.11 Annex C.

- Cisco Proprietary Fragmentation

Cisco proprietary fragmentation is used on data packets on a PVC that is also used for voice traffic.

See the module Configuring Voice over Frame Relay in the *Cisco IOS Voice, Video, and Fax Configuration Guide* for configuration tasks and examples for Cisco proprietary fragmentation.

Frame Relay-ATM Internetworking

Cisco IOS software supports the Frame Relay Forum implementation agreements for Frame Relay-ATM Interworking. Frame Relay-ATM Interworking enables Frame Relay and ATM networks to exchange data, despite differing network protocols. There are two types of Frame Relay-ATM Interworking:

FRF.5 Frame Relay-ATM Network Interworking

FRF.5 provides network interworking functionality that allows Frame Relay end users to communicate over an intermediate ATM network that supports FRF.5. Multiprotocol encapsulation and other higher-layer procedures are transported transparently, just as they would be over leased lines.

FRF.5 describes network interworking requirements between Frame Relay Bearer Services and Broadband ISDN (BISDN) permanent virtual circuit (PVC) services.

The FRF.5 standard is defined by the Frame Relay Forum Document Number FRF.5: *Frame Relay/ATM PVC Network Interworking Implementation Agreement*. For information about which sections of this implementation agreement are supported by Cisco IOS software, see Frame Relay-ATM Interworking Supported Standards.

FRF.8 Frame Relay-ATM Service Interworking

FRF.8 provides service interworking functionality that allows a Frame Relay end user to communicate with an ATM end user. Traffic is translated by a protocol converter that provides communication among dissimilar Frame Relay and ATM equipment.

FRF.8 describes a one-to-one mapping between a Frame Relay PVC and an ATM PVC.

The FRF.8 standard is defined by the Frame Relay Forum Document Number FRF.8: *Frame Relay/ATM PVC Network Service Interworking Implementation Agreement*. For information about which sections of this implementation agreement are supported by Cisco IOS software, see Frame Relay-ATM Interworking Supported Standards.

Switched Multimegabit Data Service

The Cisco implementation of the SMDS protocol is based on cell relay technology as defined in the Bellcore Technical advisories, which are based on the IEEE 802.6 standard. We provide an interface to an SMDS network using DS1 or DS3 high-speed transmission facilities. Connection to the network is made through a device called an SDSU--an SMDS digital service unit (DSU). The SDSU attaches to a router or access server through a serial port. On the other side, the SDSU terminates the line.

The implementation of SMDS supports the IP, DECnet, AppleTalk, XNS, Novell IPX, Banyan VINES, and OSI internetworking protocols, and transparent bridging.

The implementation of SMDS also supports SMDS encapsulation over an ATM interface. For more information and for configuration tasks, see Configuring ATM.

Routing of AppleTalk, DECnet, IP, IPX, and ISO CLNS is fully dynamic; that is, the routing tables are determined and updated dynamically. Routing of the other supported protocols requires that you establish a

static routing table of SMDS neighbors in a user group. Once this table is set up, all interconnected routers and access servers provide dynamic routing.



Note When configuring IP routing over SMDS, you may need to make adjustments to accommodate split horizon effects. Refer to the Configuring EIGRP module for information about how Cisco software handles possible split horizon conflicts. By default, split horizon is *disabled* for SMDS networks.

The SMDS implementation includes multiple logical IP subnetworks support as defined by RFC 1209. This RFC describes routing IP over an SMDS cloud in which each connection is considered a host on one specific private network, and points to cases where traffic must transit from network to network.

The implementation of SMDS also provides the Data Exchange Interface (DXI) Version 3.2 with *heartbeat*. The heartbeat mechanism periodically generates a heartbeat poll frame.

When a multicast address is not available to a destination, pseudobroadcasting can be enabled to broadcast packets to those destinations using a unicast address.

Link Access Procedure - Balanced and X.25

X.25 is one of a group of specifications published by the ITU-T. These specifications are international standards that are formally called *Recommendations*. The ITU-T *Recommendation X.25* defines how connections between DTE and DCE are maintained for remote terminal access and computer communications. The X.25 specification defines protocols for two layers of the Open Systems Interconnection (OSI) reference model. The data link layer protocol defined is LAPB. The network layer is sometimes called the packet level protocol (PLP), but is commonly (although less correctly) referred to as the X.25 protocol.

The ITU-T updates its *Recommendations* periodically. The specifications dated 1980 and 1984 are the most common versions currently in use. Additionally, the International Standards Organization (ISO) has published ISO 7776:1986 as an equivalent to the LAPB standard, and ISO 8208:1989 as an equivalent to the ITU-T 1984 *Recommendation X.25* packet layer. The Cisco X.25 software follows the ITU-T 1984 *Recommendation X.25*, except for its Defense Data Network (DDN) and Blacker Front End (BFE) operation, which follow the ITU-T 1980 *Recommendation X.25*.



Note The ITU-T carries out the functions of the former CCITT. The 1988 X.25 standard was the last published as a CCITT *Recommendation*. The first ITU-T *Recommendation* is the 1993 revision.

In addition to providing remote terminal access, The Cisco X.25 software provides transport for LAN protocols--IP, DECnet, XNS, ISO CLNS, AppleTalk, Novell IPX, Banyan VINES, and Apollo Domain--and bridging.

Cisco IOS X.25 software provides the following capabilities:

- LAPB datagram transport--LAPB is a protocol that operates at Level 2 (the data link layer) of the OSI reference model. It offers a reliable connection service for exchanging data (in units called *frames*) with one other host. The LAPB connection is configured to carry a single protocol or multiple protocols. Protocol datagrams (IP, DECnet, AppleTalk, and so forth) are carried over a reliable LAPB connection, or datagrams of several of these protocols are encapsulated in a proprietary protocol and carried over a LAPB connection. Cisco also implements transparent bridging over multiprotocol LAPB encapsulations on serial interfaces.

- X.25 datagram transport-- X.25 can establish connections with multiple hosts; these connections are called virtual circuits. Protocol datagrams (IP, DECnet, AppleTalk, and so forth) are encapsulated inside packets on an X.25 virtual circuit. Mappings between the X.25 address of a host and its datagram protocol addresses enable these datagrams to be routed through an X.25 network, thereby permitting an X.25 PDN to transport LAN protocols.
- X.25 switch--X.25 calls can be routed based on their X.25 addresses either between serial interfaces on the same router (local switching) or across an IP network to another router, using X.25 over TCP (XOT). XOT encapsulates the X.25 packet level inside a TCP connection, allowing X.25 equipment to be connected via a TCP/IP-based network. The Cisco X.25 switching features provide a convenient way to connect X.25 equipment, but do not provide the specialized features and capabilities of an X.25 PDN.
- ISDN D channel--X.25 traffic over the D channel, using up to 9.6 kbps bandwidth, can be used to support many applications. For example, it may be required as a primary interface where low volume sporadic interactive traffic is the normal mode of operation. For information on how to configure X.25 on ISDN, refer to the modules *Configuring X.25 on ISDN* and *Configuring X.25 on ISDN Using AO/DI*.
- PAD--User sessions can be carried across an X.25 network using the packet assembler/disassembler (PAD) protocols defined by the ITU-T Recommendations X.3 and X.29.
- QLLC--The Cisco IOS software can use the Qualified Logical Link Control (QLLC) protocol to carry SNA traffic through an X.25 network.
- Connection-Mode Network Service (CMNS)--CMNS is a mechanism that uses OSI-based network service access point (NSAP) addresses to extend local X.25 switching to nonserial media (for example, Ethernet, FDDI, and Token Ring). This implementation provides the X.25 PLP over Logical Link Control, type 2 (LLC2) to allow connections over nonserial interfaces. The Cisco CMNS implementation supports services defined in ISO Standards 8208 (packet level) and 8802-2 (frame level).
- DDN and BFE X.25--The DDN-specified Standard Service is supported. The DDN X.25 Standard Service is the required protocol for use with DDN Packet-Switched Nodes (PSNs). The Defense Communications Agency (DCA) has certified the Cisco DDN X.25 Standard Service implementation for attachment to the DDN. The Cisco DDN implementation also includes Blacker Front End operation.
- X.25 MIB--Subsets of the specifications in *SNMP MIB Extension for X.25 LAPB* (RFC 1381) and *SNMP MIB Extension for the X.25 Packet Layer* (RFC 1382) are supported. The LAPB XID Table, X.25 Cleared Circuit Table, and X.25 Call Parameter Table are not implemented. All values are read-only. To use the X.25 MIB, refer to the RFCs.
- Closed User Groups (CUGs)--A CUG is a collection of DTE devices for which the network controls access between two members and between a member and a nonmember. An X.25 network can support up to 10,000 CUGs. CUGs allow various network subscribers (DTE devices) to be segregated into private subnetworks that have limited incoming or outgoing access.

The Cisco X.25 implementation does not support fast switching.

Layer 2 VPN

Layer 2 VPN (L2VPN) services are point-to-point. They provide Layer 2 point-to-point connectivity over either an Multiprotocol Label Switching (MPLS) or a pure IP (Layer 2 Tunneling Protocol Version 3) core.

Layer 2 Tunneling Protocol Version 3

The Layer 2 Tunneling Protocol Version 3 feature expands Cisco's support of Layer 2 VPNs. Layer 2 Tunneling Protocol Version 3 (L2TPv3) is an IETF I2tpevt working group draft that provides several enhancements to L2TP to tunnel any Layer 2 payload over L2TP. Specifically, L2TPv3 defines the L2TP protocol for tunneling Layer 2 payloads over an IP core network by using Layer 2 VPNs.

L2VPN Pseudowire Redundancy

L2VPNs can provide pseudowire resiliency through their routing protocols. When connectivity between end-to-end PE routers fails, an alternative path to the directed LDP session and the user data can take over. However, there are some parts of the network where this rerouting mechanism does not protect against interruptions in service. The L2VPN Pseudowire Redundancy feature provides the ability to ensure that the CE2 router in can always maintain network connectivity, even if one or all the failures in the figure occur. The L2VPN Pseudowire Redundancy feature enables you to set up backup pseudowires. You can configure the network with redundant pseudowires (PWs) and redundant network elements.

L2VPN Interworking

Layer 2 transport over MPLS and IP already exists for like-to-like attachment circuits, such as Ethernet-to-Ethernet or PPP-to-PPP. L2VPN Interworking builds on this functionality by allowing disparate attachment circuits to be connected. An interworking function facilitates the translation between the different Layer 2 encapsulations. The L2VPN Interworking feature supports Ethernet, 802.1Q (VLAN), Frame Relay, ATM AAL5, and PPP attachment circuits over MPLS and L2TPv3.

Layer 2 Local Switching

Local switching allows you to switch Layer 2 data between two interfaces of the same type (for example, ATM to ATM, or Frame Relay to Frame Relay) or between interfaces of different types (for example, Frame Relay to ATM) on the same router. The interfaces can be on the same line card or on two different cards. During these kinds of switching, the Layer 2 address is used, not any Layer 3 address. Same-port local switching allows you to switch Layer 2 data between two circuits on the same interface.

WAAS Express

Cisco's WAAS Express software interoperates with WAN optimization headend applications from Cisco. Cisco WAAS Express improves WAN access and use by optimizing applications, such as backup (is backup an application or a mechanism?), that require high bandwidth or are bound to a LAN.

WAAS Express helps enterprises meet the following objectives:

- Complement the Cisco WAN optimization system by adding the capability to branch routers.
- Provide branch office employees with LAN-like access to information and applications across a geographically distributed network.
- Minimize unnecessary WAN bandwidth consumption through the use of advanced compression algorithms.
- Virtualize print and other local services to branch office users.

- Improve application performance over WAN by addressing the following common issues:
 - Low data rates (constrained bandwidth)
 - Slow delivery of frames (high network latency)
 - Higher rates of packet loss (low reliability)

The Network Analysis Module (NAM) Performance Agent (PA) for WAAS Express analyzes and measures network traffic. The PA enables baselining, monitoring, and troubleshooting of application performance. The analysis and measurement of network traffic is done by the Measurement, Aggregation, and Correlation Engine (MACE). MACE performs the required measurements on a subset of traffic and exports the necessary metrics to a target.



CHAPTER 2

Configuring WAAS Express

Cisco's Wide-Area Application Services (WAAS) Express feature is a key component of the Cisco WAAS product portfolio. WAAS Express is a cost-effective, IOS-based, WAN optimization solution that increases the amount of available bandwidth for small-to-mid-size branch offices and remote locations, while accelerating TCP-based applications that operate in a WAN environment.

WAAS Express uses the capabilities of IOS software and provides a small-footprint, cost-effective solution that transparently integrates into the Integrated Services Routers (ISRs) Generation 2 (G2). WAAS Express extends the WAN optimization solution to the entire ISR G2. WAAS Express is fully interoperable with WAAS on Service Module-Service Ready Engine (SM-SRE) modules and with WAAS appliances and can be managed by a common WAAS Central Manager (WCM).

- [Finding Feature Information, on page 9](#)
- [Prerequisites for WAAS Express, on page 9](#)
- [Restrictions for WAAS Express, on page 10](#)
- [Information About WAAS Express, on page 11](#)
- [How to Configure WAAS Express, on page 33](#)
- [Configuration Examples for WAAS Express, on page 49](#)
- [Additional References, on page 52](#)
- [Feature Information for WAAS Express, on page 53](#)

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see [Bug Search Tool](#) and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for WAAS Express

WAAS Express requires a Wide-Area Application Virtualization Engine (WAVE) as a peer device, running WAAS Version 4.4.3c.9 or higher, at the data center. WAAS Express requires no additional hardware, assuming that the maximum memory is installed on the device. WAAS Express uses the device's CPU and memory for

optimization. The data center component consists of a Cisco WAAS data center appliance and a WAAS Central Manager to manage WAAS Express-enabled devices.

You must have a valid license for WAAS Express. WAAS Express includes an evaluation license for 60 days. WAAS Express switches to one of the following modes based on the available memory on the device:

- **WAAS_Standard**—If maximum memory (the amount of memory that can be upgraded on the device) is available on the device
- **WAAS_Trial_Limited**—If default memory (the amount of memory with which the device is shipped) is available on the device
- **WAAS_Disabled**—If less than default memory is available on the device



Note The maximum and default memory depend on the device.

As part of the new Appxk9 license support for WAAS Express in Cisco IOS Release IOS 15.3(3)M, if you are upgrading the WAAS Express devices to Cisco IOS Release 15.3(3)M and later releases, you need to upgrade the WAAS Central Manager to 5.3.1 or later, else the devices go offline.

The number of connections that can be optimized on a platform running WAAS Express depend on the mode enabled. For example, on a particular platform, WAAS Express can optimize 200 connections in the **WAAS_Standard** mode, but can optimize only up to 50 connections in the **WAAS_Trial_Limited** mode; if there are more than 50 concurrent connections in **WAAS_Trial_Limited** mode, they will be passed through and not optimized.

The amount of Data Redundancy Elimination (DRE) cache also depends on the mode being operated.

Restrictions for WAAS Express

- Wide-Area Application Services (WAAS) Express is not supported on Cisco Integrated Services Routers (ISRs) Generation 1 (G1) (Cisco 1800, Cisco 2800, and Cisco 3800 Series Routers). In Cisco 1905 and 1921 routers, Data Redundancy Elimination (DRE) is disabled because the maximum memory in these devices is 512 MB. For more information about DRE, see the “Compression” section.
- WAAS Express does not support the definition of a user-defined policy map of the type `waas`.
- WAAS Express is not supported on Layer 2 Tunneling Protocol version 2 (L2TPv2) and Layer 2 Tunneling Protocol version 3 (L2TPv3) interfaces.
- SSL 3.0 is deprecated in the following releases:
 - Cisco IOS Release 15.3(3)M5 and later maintenance releases
 - Cisco IOS Release 15.4(3)M2 and later releases

Therefore, WAAS Express does not support SSL3.0, but supports Transport Layer Security version 1.0(TLSv1.0) only. However, SSL 3.0 is supported in Cisco IOS Release 15.4(3)M.

- The maximum number of concurrent connections optimized by WAAS Express depends on the platform.
- As WAAS Express uses the device CPU to optimize the traffic, there might be a performance impact on other CPU-heavy features as well as on the performance of WAAS Express.

- You can configure Flexible NetFlow to collect WAAS Express flow information. If WAAS Express, cryptomaps, and Flexible NetFlow are configured on the same WAN interface, the ingress Flexible NetFlow record contains only a small number of optimized packets.
- The following restrictions apply to WAAS Express licensing:
 - Low memory is not available in permanent licenses.
 - In the WAAS_Trial_Limited mode, a limited number of flows are optimized, the available DRE memory is limited, and I/O memory-pool resizing is not performed.
- WAAS Express should be enabled only on an interface that is designated as the WAN interface.
- WAAS Express does not support the following configurations, where:
 - Packets from a connection are routed between two WAN interfaces, that is, packets are received on a WAN interface and routed to another WAN interface. As a result, packets neither originate from nor are destined to the branch. WAAS Express is enabled on both WAN interfaces.
 - Packets from a connection are routed to the same WAN interface, that is, packets are received on a WAN interface and routed to the same WAN interface. As a result, packets neither originate from nor are destined to the branch. WAAS Express is enabled on the WAN interface.
 - Packets from a connection are asymmetrically routed between two or more edge devices. At least one of these devices must have WAAS Express enabled on the WAN link.
 - Traffic is load balanced (in per flow/destination manner) across multiple WAN links, but WAAS Express is enabled only on a subset of these interfaces.

Information About WAAS Express

WAAS Express Overview

The Wide-Area Application Services (WAAS) Express software optimizes TCP traffic flows across a WAN. WAAS Express integrates with native services such as security, NetFlow, and quality of service (QoS). WAAS Express provides the following benefits:

- Bandwidth optimization
- Application acceleration
- Increase in remote user productivity
- Interoperation with existing Cisco WAAS infrastructure
- Network transparency
- Deployment flexibility with on-demand service enablement

WAAS Express is supported on Cisco ISR G2 devices (Cisco 800, Cisco 890, Cisco 1905, Cisco 1921, Cisco 1941, Cisco 2901, Cisco 2911, Cisco 2921, Cisco 2951, Cisco 3925, and Cisco 3945 Series routers) only. These devices must have the maximum Dynamic random-access memory (DRAM) installed because WAAS

Express stores Data Redundancy Elimination (DRE) cache in DRAM. For more information about DRE, see the “Compression” section. WAAS Express supports WAN speeds in the range from 1.5 to 10 Mbps.

WAAS Express switches to one of the following modes based on the available memory on the device: WAAS_Standard, WAAS_Trial_Limited, or WAAS_Disabled. The table below lists the default memory and maximum memory that can be installed on each of the devices that supports WAAS Express.

Device Platform	Default Memory	Maximum Memory
88x	512 MB	768 MB
89x	512 MB	768 MB
1905/1921	512 MB	512 MB
1941	512 MB	2.5 GB
29xx	512 MB	2.5 GB
2951	1 GB	4 GB
3925/3945	1 GB	4 GB

The Cisco WAN optimization system consists of ISR G2 devices running the WAAS Express software feature and WAVEs that work together to optimize TCP traffic in your network. When client and server applications attempt to communicate with each other, the network intercepts the traffic and acts on behalf of the client application and the destination server. WAAS Express and Wide-Area Application Virtualization Engines (WAVE) examine the traffic and use built-in application policies to determine whether the traffic in the network can be optimized.

Any application that uses TCP as its underlying transport can benefit from WAAS Express. However, a lot depends on the redundant elements in the application’s transactions and the WAN bandwidth condition. The more the redundancy, the more the application is benefitted by WAAS Express. Typical applications that benefit from WAAS Express include HTTP, FTP, and mail applications.

WAAS Express uses the following TCP optimization techniques to overcome the most common challenges associated with transporting traffic over a WAN:

- **Constrained bandwidth:** Data caching and data compression reduce the amount of data sent over the WAN, which increases data transfer rates. These solutions improve the total transaction time on congested links by reducing the amount of data sent across the WAN.
- **Packet loss:** The optimized TCP stack in WAAS overcomes issues associated with high packet loss and protects the communicating endpoints from the state of the WAN.

Two devices using WAAS Express can optimize network traffic between themselves, but without any DRE.

When a WAAS Express device is moved from one device group to another device group on the WAAS Central Manager (WCM), the policy definitions under the new device group do not function. When a device is unassigned from a device group, the device gets the policies from the backup policy set from the previously assigned group.

Perform the following steps on WCM when you move a device between device groups. You can either perform only the first step or perform from step 2 through step 4 to move a device between device groups.

1. Go to the “Policy Definitions” page of the device that you want to move; select the new device group, and click “Submit”.

2. Go to “Assign Devices” page of device group-1, and unassign the device from the device group.
3. Go to “Assign Devices” page of device group-2, and assign the device to the device group.
4. Go to “Policy Definitions” page of device group-2, and click “Force DG settings”.

WAAS Express offers the following benefits:

- Ease of deployment: WAAS Express deployment is easy through simple software activation on any Cisco ISR G2 device running Cisco software.
- Integration with device services: WAAS Express integrates with security, QoS, and other services native to Cisco software.
- Application acceleration: WAAS Express mitigates the effects of WAN latency while allowing data to be transferred faster.
- Investment protection: WAAS Express enables you to deploy your current WAAS deployments to smaller branches and devices without having to invest heavily in WAN optimization overlay technologies.

The following is a list of some of the features with which WAAS Express interoperates:

- Dynamic Multipoint VPN (DMVPN)
- Flexible NetFlow
- IPsec
- Network Address Translation (NAT)
- Quality of service (QoS)
- Virtual Tunnel Interfaces (VTIs)
- Zone-based Firewall

WAAS Express-Enabled Traffic Optimization Process

The following steps describe how a Wide-Area Application Services (WAAS) Express-enabled network optimizes connections between a branch office client and the destination server:

1. A branch office client attempts to connect to the destination server over the native application port.
2. The branch client intercepts the traffic.
3. The branch client performs the following actions:
 1. Examines parameters in the TCP headers of the traffic and then refers to the application policies to determine whether the intercepted traffic should be optimized. Information such as the source and destination IP addresses in the TCP header allows the branch client to match the traffic to an application policy. For a list of the default policies, see the “WAAS Application Policies” section.
 2. Negotiates with the data-center Wide-Area Application Virtualization Engine (WAVE) about whether the traffic must be optimized.
 3. Based on the negotiation, if the branch client determines that the traffic should be optimized, it adds information to the TCP header that informs the next client in the network path to optimize the traffic.
4. The branch client passes the client request through the network to its original destination server.

5. The data-center WAVE performs the following actions:
 1. Intercepts traffic going to the destination server.
 2. Establishes an optimized connection with the branch client. If optimization is disabled on the data-center WAVE, then the connection established between the data-center WAVE and the branch client is not optimized, and the traffic passes over the network unoptimized.
6. WAAS optimizes subsequent traffic between the branch client and data-center WAVE depending on the connection type.

WAAS Express does not optimize traffic in the following scenarios:

- The number of concurrent connections exceed the maximum number of concurrent connections supported on a WAAS Express device.
- WAAS Express and WAVE intercept non-TCP traffic such as Internet Control Message Protocol (ICMP).
- WAVE is overloaded and does not have resources to optimize traffic.
- The intercepted traffic matches an application policy that specifies to pass the traffic unoptimized.



Note If unoptimized traffic reaches a WAVE, the WAVE forwards the traffic in pass-through mode without affecting the performance of the application by using the passed-through connection.

Key Services of WAAS Express

Transport Flow Optimization

WAAS Express uses the transport flow optimization (TFO) features described in the following sections to optimize the traffic intercepted by WAAS devices. TFO protects the communicating clients and servers from negative WAN conditions such as bandwidth constraints, packet loss, congestion, and retransmission.

Windows Scaling

Windows scaling allows the receiver of a TCP packet to advertise that its TCP receive window can exceed 64 KB. The receive window size determines the amount of space available on the receiver for unacknowledged data. Windows scaling allows TCP endpoints to take advantage of the bandwidth available in your network and not be limited to the default window size specified in the TCP header.



Note WAAS Express limits the maximum receive window size to 64 KB on the WAN side and to 32 KB on the LAN side.

RFC 1323 describes TCP extensions for high performance.

Selective Acknowledgement

Selective Acknowledgement (SACK) is an efficient packet loss recovery and retransmission feature that allows clients to recover from packet losses more quickly than the default recovery mechanism used by TCP.

By default, TCP uses a cumulative acknowledgment scheme that forces the sender either to wait for a round trip to learn if any packets were not received by the recipient or to unnecessarily retransmit segments that may have been correctly received.

SACK allows the receiver to inform the sender about all segments received, so the sender needs to retransmit only the segments that are lost.

RFC 2018 describes TCP SACK options.

Binary Increase Congestion TCP

Binary Increase Congestion (BIC) TCP is a congestion management protocol that enables a network to recover quickly from packet loss events.

When a network loses a packet, BIC TCP reduces the receiver's window size and sets that reduced size as the new value for the minimum window size value. BIC TCP then sets the maximum window size value to the size of the window that existed just before the packet loss occurred. Because packet loss occurred at the maximum window size, the network can transfer traffic without dropping packets with sizes between the minimum and maximum window size values.

If BIC TCP does not register packet loss at the updated maximum window size, then this window size becomes the new minimum. If packet loss does occur, the updated window size becomes the new maximum. This process continues until BIC TCP determines the new optimum minimum and maximum window size values.

Compression

WAAS Express uses two compression technologies to help reduce the size of data transmitted over a WAN: DRE and Lempel-Ziv (LZ).

These compression technologies reduce the size of transmitted data by removing redundant information before sending the shortened data stream over WAN. By reducing the amount of transferred data, WAAS compression can reduce network utilization and application response times.

When WAAS Express uses compression to optimize TCP traffic, it replaces repeated data in the stream with a much shorter reference and then sends the shortened data stream across the WAN. The receiving WAAS Express device uses its local redundancy library to reconstruct the data stream before passing it to the destination client or server.

LZ

Lempel-Ziv (LZ) is a standards-based compression that can minimize the amount of bandwidth consumed by a TCP flow. LZ compression operates on smaller data streams and keeps limited compression history.

Traffic between two WAAS Express devices is optimized using TFO and LZ.

LZ compression can be used in conjunction with DRE or independently.

DRE

The Data Redundancy Elimination (DRE) compression scheme is based on a shared cache architecture, where each device involved in compression and decompression shares the same redundancy library.

DRE operates on significantly large data streams (typically tens to hundreds of bytes or more) and maintains a much larger compression history. Large chunks of redundant data are common in file system operations when files are incrementally changed from one version to another or when certain elements such as file headers and logs are common to many files.

In WAAS Express, DRE is performed completely in device memory; thus maximum DRAM is required in every platform.



Note DRE optimization is not supported for connections between WAAS Express devices. In such cases, traffic is optimized using TFO and LZ.

WAAS Express compresses upload traffic (with some limitations) by using the DRE algorithm. WAAS Express decompresses the download traffic that is compressed using DRE.

With WAAS Express Phase 2, DRE compression is also supported in the upload direction. You can enable or disable the upload DRE operations for troubleshooting purposes.

Upload DRE is useful in the download-edit-upload scenario, where a user in a branch office downloads a file from the data center, modifies the file, and uploads the modified document back to the data center. If the modifications are small and localized, the upload of the modified file can benefit from the unmodified contents stored in the DRE cache. The compression in the upload direction is performed based on the cache entries that were added in the download direction.

Autodiscovery of WAAS Express Devices

The autodiscovery feature of WAAS Express enables WAVEs and WAAS Express devices to automatically locate peer WAVEs on a network by adding TCP options on control packets. After discovering a peer device automatically, the WAVEs terminate and separate the LAN-to-WAN TCP connections and add a buffering layer to resolve the differing speeds or WAAS Express proxies the connection on the device in different segments to achieve optimization benefits. After a WAVE establishes a connection with a peer WAVE, the two devices can establish an optimized link for TCP traffic or pass the non-TCP traffic as unoptimized.

The autodiscovery of peer WAAS devices is achieved using TCP options. These TCP options are recognized and understood only by WAAS devices and are ignored by non-WAAS devices.

A server is blacklisted by WAAS Express if the server is not able to receive TCP packets with options because these TCP packets are being blocked by network devices such as firewalls. WAAS Express learns not to send TCP packets with options to these blacklisted servers.

Application Acceleration

In addition to the TCP optimization features that enhance the flow of traffic over a WAN, WAAS Express provides selected application acceleration benefits. WAAS Express reduces the response time of remote applications. Even though TFO optimizes traffic over a WAN, protocol messages between branch office clients and remote servers can cause slow application response time. To resolve this issue, WAAS Express helps to respond to messages locally so that the client need not wait for a response from the remote server.

WAAS Express supports the following application accelerators:

- CIFS-Express—Optimizes Common Internet File System (CIFS) traffic exchanged with a remote file server.
- HTTP-Express—Optimizes HTTP traffic.
- SSL-Express—Optimizes encrypted Secure Sockets Layer (SSL) and Transport Layer Security (TLS) traffic. SSL-Express accelerator provides traffic encryption and decryption within a WAAS Express-enabled network to allow end-to-end traffic optimization.

For an accelerator to operate, you must enable the accelerator on both the WAAS Express device and the peer WAVE at either end of a WAN link. CIFS-Express is a single-sided accelerator; it works when it is enabled only on the WAAS Express device. Each WAAS Express-enabled device uses application policies to match specific types of traffic to an application and to determine whether that application traffic should be optimized and accelerated. For more information, see the “WAAS Application Policies” section.

All WAAS Express accelerators, except SSL-Express accelerator, are mutually exclusive. This means that only one accelerator is applied to a specific flow. The only two accelerators that can act in conjunction are SSL-Express and HTTP-Express accelerators. TFO, along with DRE and LZ, forms a transport optimization (TO) module.

For a particular flow, the following optimizations are possible:

- Application accelerator-only (CIFS-Express accelerator or HTTP-Express accelerator)
- TFO-only (no accelerator)
- TFO (TFO+LZ or TFO+DRE or TFO+LZ+DRE)
- HTTP-Express accelerator + TFO
- CIFS-Express accelerator + TFO
- SSL-Express accelerator + TFO
- SSL-Express accelerator + HTTP-Express accelerator + TFO

CIFS-Express Accelerator

CIFS-Express accelerator allows a WAAS Express-enabled device to reply to client requests by using locally cached data instead of retrieving this data from remote file and application servers.

In a typical Common Internet File System (CIFS) application accelerator scenario, a client sends a large number of synchronous requests that require the client to wait for a response before sending the next request. Compressing data over the WAN is not sufficient for an acceptable response time. For example, when you open a 5 MB Word document, about 700 CIFS requests (550 read requests plus 150 other requests) are produced. If all these requests are sent over a 100 ms round trip WAN, the response time is at least 70 seconds (700 x 0.1 seconds). WAAS CIFS-Express accelerator minimizes the synchronous effect of the CIFS protocol, which reduces application response time.

CIFS-Express accelerator requires about 10 MB of memory as a global entity (across all platforms) and about 130 KB for each supported flow. Other accelerators, such as HTTP-Express accelerator, that are mutually exclusive with CIFS-Express accelerator use the same 130 KB per flow.

CIFS-Express accelerator supports the following:

- Optimization for file sharing services. However, CIFS-Express accelerator does not support print services and Remote Procedure Call (RPC) optimization.
- Optimization for operating systems such as Windows 2000, Windows XP, Windows Vista, Windows 7, Linux, Mac, and NetApp.
- Optimization for Server Message Block Version 1 (SMBv1).

CIFS-Express accelerator operates only on flows where WAAS Express is configured on an edge device; CIFS-Express accelerator is not applied to a connection if the WAAS Express device acts as a core device for a CIFS-Express session.

You can enable CIFS-Express accelerator by using the **enable** command in WAAS CIFS configuration mode and disable it by using the **no** form of the command. Enter the WAAS CIFS configuration mode by using the **accelerator cifs-express** command in parameter map configuration mode. In addition to the global CIFS-Express accelerator configuration, you can configure traffic streams, classified by the Cisco Common Classification Policy Language policy map, that will go through CIFS-Express acceleration. This can be achieved by updating the class map configuration in WAAS Express by using the **optimize tfo** command.

CIFS works on TCP ports 139 and 445. Both the ports are subject to optimization when synchronization messages (SYNs), with one of these values on the destination port, arrive on the LAN side.

The subsequent sections describe CIFS-Express acceleration that require minimum memory and CPU resources.

Read Ahead

The read ahead feature of the CIFS-Express accelerator allows WAAS Express to read a file ahead of user requests when an application is conducting a sequential file read. The maximum read request in CIFS is limited to 64 KB. Most applications, including Microsoft Office and Windows Explorer, do not pipeline read requests.

Async Write

The async write feature of CIFS-Express accelerator facilitates efficient write operations. The maximum write request in CIFS is 64 KB.

The mechanism followed by CIFS-Express accelerator to write asynchronously is described below.

Whenever a write request arrives, the accelerator forwards the request to the server and provides a local success reply to the client.

- If an error response is received from the server, a local error response is sent to one of the subsequent write requests.
- If there are no subsequent write requests before the file is closed, a local error reply is sent to the close request.

Because the primary reason for write request failures is a lack of disk space or quota, async write optimization is disabled when the available space drops below 20 MB. This value can be configured.

Negative Caching

The negative caching feature of CIFS-Express accelerator allows WAAS Express to store information about missing files to reduce round-trips across the WAN.

While browsing directories (remote or local) in a Windows operating system before Windows 7, Windows Explorer continuously asks for mostly nonexistent files and metadata, such as icons, thumbnails, and author information. This metadata is stored in “alternate data streams,” and therefore, the negative caching is also referred to as the alternate data streaming negative caching.

The accelerator caches negative response for 3 seconds (this can be configured) and provides local responses if possible. The reason for the relatively short, default 3 seconds interval is to decrease the probability of issues that might be caused when alternate data streams are created by another desktop during the same period.

HTTP-Express Accelerator

WAAS Express optimizes web-based applications by using the HTTP-Express accelerator. This accelerator express enables a fast connection setup and eliminates round trips associated with the connection setup. HTTP-Express accelerator also helps to minimize the impact of latency or limited bandwidth.

HTTP-Express accelerator operates only on flows where WAAS Express is configured on an edge device. WAAS Express applies HTTP-Express acceleration only if the connections originate from the branch; the connections originating from the data center are not optimized.

You can enable HTTP-Express accelerator by using the **enable** command in WAAS HTTP configuration mode and disable the accelerator by using the **no** form of the command. Enter the WAAS HTTP configuration mode by using the **accelerator http-express** command in parameter map configuration mode. In addition to the global HTTP-Express accelerator configuration, you can configure traffic streams, classified by the Cisco Common Classification Policy Language policy map, that will go through HTTP-Express acceleration. This can be achieved by updating the class map configuration in WAAS Express using the **optimize tfo** command.

By default, when the HTTP-Express accelerator is enabled, it is applied for ports 80, 8080, 8000, 8001, and 3128, which are classified as HTTP under the default application policy configuration.

HTTP Metadata Caching

HTTP metadata caching enables the client side WAVE to respond locally to certain types of HTTP requests by caching the metadata information found in the HTTP response headers. A metadata cache is used to store attributes of an object, but not the actual object. Expiration time (time beyond which an object cannot be cached), file name, and file size are some examples of metadata.

Metadata cache is used to eliminate unnecessary round trips to the server over the WAN by locally servicing conditional requests at the branch WAAS Express when possible (thereby reducing the latency experienced by the client).

Metadata cache serves the following types of HTTP responses from the local cache of the edge WAAS Express device:

- Redirect response
- Conditional response
- Authorization-required response

HTTP metadata cache is used to cache all metadata entries necessary for the above listed responses. The table below lists the header information stored in metadata cache:

Table 1: HTTP Information Stored in Metadata Cache

Header	Description
Etag	The entity tag associated with this entity. It is represented as a string.
Expires	The date and time at which this entity will no longer be valid and will need to be fetched again from the original source. It is expressed in absolute time and is used for HTTP1.0+ compatibility. It is represented as a string.
Max-age	The amount of time for which the entry will still be valid. It is expressed in seconds, starting at the time the server created it. This is the HTTP1.1 accepted expiration time.
Last-Modified	The last date and time when this entity changed. It is expressed in absolute time as a string.

If the responses from the server contain any cache-control directives that do not allow caching, HTTP-Express accelerator does not cache any content from these responses. The cache-control attributes include:

- no-cache
- no-store
- max-age or s-maxage=0
- must-revalidate and proxy-revalidate

DRE Hints

The DRE module operates on chunks of data and helps to avoid transferring redundant information on the WAN, irrespective of the Layer 7 protocol.

HTTP-Express accelerator can pass DRE hints to the DRE module at any point during a session. The hints are provided to the DRE module based on payload, resulting in better compression and improved overall DRE efficiency. Since HTTP-Express accelerator parses the Layer 7 content, the accelerator can provide the following useful hints to the DRE module:

- Apply LZ or Not

When the response from the server is already compressed, such as in the form of a jpeg or gzip file, HTTP-Express accelerator can instruct the DRE module to not apply LZ compression again. This can save some CPU cycles on WAAS Express.

- Skip Bytes

Multiple HTTP requests that request for the same file can have different headers even if the file being transferred is the same. To improve DRE compression in these cases, HTTP-Express accelerator can instruct DRE to skip the header bytes.

Server Encoding Suppression

Servers commonly serve web pages with minimal variations to different clients. Such pages, when compressed by the server, can vary considerably even when the uncompressed files are very similar. This reduces the effectiveness of DRE.

This problem can be solved by suppressing the server side encoding and allowing the core side WAAS device to apply LZ compression on the data after using DRE. If the client side WAAS Express removes any Accept-Encoding header from HTTP requests before sending them to the server, the server will not compress the data it sends. WAAS Express uses this mechanism to provide better compression on HTTP response from the server and also frees the server from the additional computation required to compress responses.

SSL-Express Accelerator

WAAS Express optimizes Secure Sockets Layer (SSL) web-based applications by applying SSL-Express acceleration along with generic optimization and compression techniques (TFO/DRE/LZ). By adding the capability to encrypt and decrypt SSL traffic, SSL-Express accelerator provides the benefits of DRE, LZ, and TFO to the SSL traffic between the client and the server. If SSL-Express accelerator is not enabled, the WAAS Express DRE optimizations are not effective on SSL-encrypted traffic. SSL-Express accelerator enables WAAS Express to decrypt data and apply optimizations while maintaining the security of the connection.



Note SSL-Express accelerator is applicable only if LZ and DRE optimizations are successfully negotiated and applied to the connection; otherwise, only TFO is applied to the connection.

SSL and TLS protocols encrypt TCP segments by using various symmetric cryptographic algorithms, such as Advanced Encryption Standard (AES) and Data Encryption Standard (DES). SSL-Express accelerator supports SSL Version 3.0 and TLS Version 1.0. If a client attempts to initiate an SSL session with a higher version, the core device will attempt to downgrade the session to a lower supported version.

SSL-Express accelerator operates only on flows where WAAS Express is configured on an edge device; SSL-Express accelerator is not applied to a connection if the WAAS Express device acts as a core device for an SSL session. In addition, WAAS Express applies SSL-Express acceleration only if the connections originate from the branch; the connections originating from the data center are not optimized.

You can enable SSL-Express accelerator by using the **enable** command in WAAS SSL configuration mode and disable the accelerator by using the **no** form of the command. Enter the WAAS SSL configuration mode by using the **accelerator ssl-express** command in parameter map configuration mode. You can create a cipher list, specify the version of the SSL protocol, and associate the accelerator with a valid trustpoint label to configure SSL-Express accelerator. Default values are used if these parameters are not configured. You can also configure peering service to control the secure communications established by SSL-Express accelerator between WAAS Express devices while optimizing SSL connections.

When an SSL connection is optimized by SSL-Express accelerator, WAAS Express generates an empty SSL fragment and sends it to a client as the first encrypted message. This behavior can impact interoperability with older versions of the client applications such as Internet Explorer 6. You can disable the generation and sending of this empty SSL fragment using the **no empty-ssl-fragment-insertion** command in WAAS SSL configuration mode.

SSL Sessions

An SSL session comprises the following phases: SSL handshake, SSL rehandshake, SSL data transfer, and SSL shutdown or alerts. Since SSL is an end-to-end security protocol, decryption of data by any device in between the end hosts is not possible without the knowledge of the private key of the server. The session keys derived by the core device are sent to the edge device over the WAN-to-WAN SSL session set up between the WAAS devices.

There are two modes of operation: one where the server's private key and certificate are installed on the WAAS core device, and the other where the WAAS core device uses its own private key and certificate during the SSL handshake. The limitation of the second mode of operation is that the client browser displays a warning message stating that the server certificate received is not that of the server that the client is trying to reach. After the WAAS core device has knowledge of the private keys, it can derive the necessary SSL session keys and pass them on to the WAAS edge device. After this point, any data received over the SSL session at the edge device can be decrypted, compressed, reencrypted, and sent to the WAAS core device. At the WAAS core device again, the received message can be decrypted, uncompressed, and the original data can be sent encrypted to the server.

SSL-Express accelerator leverages the public key infrastructure (PKI) on the WAAS Express device. PKI provides comprehensive certificate management (required for the WAN-to-WAN session), including the management of self-signed and certificate authority (CA) certificates, and certificate verification and certificate revocation checks using the Simple Certificate Enrollment Protocol (SCEP).



Note SSL-Express accelerator uses buffers from the public buffer pools for encrypt and decrypt operations. Ensure that you tune the public pool for minimal impact due to trims, misses, and failures.

After the SSL handshake is complete, SSL-Express accelerator examines the application data in the LAN-to-WAN connection to determine whether HTTP is being used. If HTTP is being used and HTTP-Express accelerator is enabled, HTTP acceleration is applied.

Cipher Lists

Cipher lists are sets of cipher suites that you can assign to an SSL-Express accelerator configuration. A cipher suite is an SSL encryption method that includes the key exchange algorithm, the encryption algorithm, and the secure hash algorithm.

SSL-Express accelerator supports the following cipher suites for WAN-to-WAN peering sessions:

- RSA_WITH_3DES_EDE_CBC_SHA
- RSA_WITH_AES_128_CBC_SHA
- RSA_WITH_DES_CBC_SHA
- RSA_WITH_RC4_128_MD5
- RSA_WITH_RC4_128_SHA

WAAS Express and WAVE can interoperate only when common cipher suites are used. If the WAAS Express device does not support the cipher suite that is negotiated for the client-to-server connection, WAAS Express resets the connection. The cipher suite supported by a WAAS Express device for a client-to-server session depends on the capability of the crypto engine available on the device.

SSL-Express accelerator supports the following cipher suites for client-to-server sessions:

- dhe-rsa-with-3des-ede-cbc-sha
- dhe-rsa-with-aes-128-cbc-sha
- dhe-rsa-with-aes-256-cbc-sha
- dhe-rsa-with-des-cbc-sha
- rsa-with-3des-ede-cbc-sha
- rsa-with-aes-128-cbc-sha
- rsa-with-aes-256-cbc-sha
- rsa-with-des-cbc-sha
- rsa-with-rc4-128-md5
- rsa-with-rc4-128-sha

WAAS Express Application Policies

The WAAS Express software includes over 150 default application policies that help WAAS Express to classify and optimize some of the most common types of traffic in a network.

The table below lists the default applications and classifiers that WAAS Express either optimizes or passes through based on the policies that are provided with the system.

The WAAS Express software supports the following optimization actions based on the type of traffic it encounters:

- TFO—A collection of optimization technologies such as automatic windows scaling, increased buffering, and selective acknowledgment that optimize all TCP traffic over a network.
- DRE—A compression technology that reduces the size of transmitted data by removing redundant information before sending the shortened data stream over the WAN. DRE operates on significantly larger streams and maintains a much larger compression history than LZ compression.
- LZ—A compression technology that operates on smaller data streams and keeps limited compression history compared to DRE.
- Application accelerators—A collection of individual accelerators for the following traffic types: CIFS, HTTP, and SSL.

We recommend that you review the default policies and modify them as appropriate before you create a new application policy. Often, modifying an existing policy is easier than creating a new one.

When reviewing the table below, note the following:

- The subheadings represent the application names. The associated classifiers are listed under these subheadings. For example, Authentication is a type of application and Kerberos is a classifier for that application.
- The word *monitored* indicates that the applications are monitored by the WAAS Central Manager (WCM), which can display statistics for only 20 applications at a time.

Table 2: Default Traffic Policies

Classifier	WAAS Express Action	Destination Ports
Authentication		
Apple-SASL	Passthrough	3659
auth	Passthrough	113
Kerberos	Passthrough	88, 888, 2053
kerberos-adm	Passthrough	749
klogin	Passthrough	543
kpasswd	Passthrough	464
kshell	Passthrough	544
TACACS	Passthrough	49
tell	Passthrough	754
Backup (monitored)		
Amanda	TFO	10080
BackupExpress	TFO	6123
CommVault	TFO	8400-8403

Classifier	WAAS Express Action	Destination Ports
connected	TFO	16384
IBM-TSM	TFO+LZ+DRE	1500–1502
Legato-NetWorker	TFO	7937, 7938, 7939
Legato-RepliStor	TFO	7144, 7145
Veritas-BackupExec	TFO	1125, 3527, 6101, 6102, 6106
Veritas-NetBackup	TFO	13720, 13721, 13782, 13785
Call Management		
Cisco-CallManager	Passthrough	2443, 2748
cisco-q931-backhaul	Passthrough	2428
cisco-sccp	Passthrough	2000-2002
h323hostcall	Passthrough	1720
h323hostcallsc	Passthrough	1300
mgcp-callagent	Passthrough	2727
mgcp-gateway	Passthrough	2427
sip	Passthrough	5060
sip-tls	Passthrough	5060
VoIP-Control	Passthrough	1718, 1719, 11000–11999
Computer-Aided Design (CAD)		
PDMWorks	TFO+LZ+DRE	30000, 40000
Conferencing		
cuseeme	Passthrough	7640, 7642, 7648, 7649
ezMeeting	Passthrough	10101–10103, 26260, 26261
MS-NetMeeting	Passthrough	522, 1503, 1731
proshare	Passthrough	5713–5717
PSOM-MTLS	Passthrough	8057
VocalTec	Passthrough	1490, 6670, 22555, 25793
Console		
cmd	Passthrough	514

Classifier	WAAS Express Action	Destination Ports
exec	Passthrough	512
login	Passthrough	513
sshell	Passthrough	614
Telnet	Passthrough	23, 107
Telnets	Passthrough	992
Content Management (<i>monitored</i>)		
dmdocbroker	TFO+LZ+DRE	1489
Filenet	TFO+LZ+DRE	32768–32774
Directory Services (<i>monitored</i>)		
LDAP	TFO+LZ+DRE	389, 8404
ldaps	Passthrough	636
msft-gc	TFO+LZ+DRE	3268
msft-gc-ssl	Passthrough	3269
E-mail and Messaging		
ccmail	TFO+LZ+DRE	3264
groupwise	TFO+LZ+DRE	1677, 2800, 3800, 7100, 7101, 7180, 7181, 7205, 9850
imap	TFO +LZ+DRE	143
imap3	TFO+LZ+DRE	220
imaps	TFO	993
iso-tsap	TFO+LZ+DRE	102
lotusnote	TFO+LZ+DRE	1352
MDaemon	TFO+LZ+DRE	3000, 3001
NNTP	TFO+LZ+DRE	119
nntp	TFO	563
openmail	TFO+LZ+DRE	5755, 5757, 5766, 5767, 5768, 5729
pcmail-srv	TFO+LZ+DRE	158
pop3	TFO+LZ+DRE	110

Classifier	WAAS Express Action	Destination Ports
pop3s	TFO+LZ+DRE	995
QMP	TFO+LZ+DRE	209
smtp	TFO+LZ+DRE	25
smtps	TFO	465
Enterprise Applications <i>(monitored)</i>		
MS-GROOVE	TFO	2492
SAP	TFO+LZ+DRE	3200–3204, 3206–3219, 3221–3224, 3226–3259, 3261–3263, 3265–3267, 3270–3282, 3284–3305, 3307–3351, 3353–3388, 3390–3399, 3600–3658, 3662–3699
Siebel	TFO+LZ+DRE	2320, 2321, 8448
File System <i>(monitored)</i>		
afpovertcp	TFO+LZ+DRE	548
afs3	TFO+LZ+DRE	7000–7009
ncp	TFO+LZ+DRE	524
NFS	TFO+LZ+DRE	2049
sunrpc	Passthrough	111
File Transfer <i>(monitored)</i>		
BFTP	TFO+LZ+DRE	152
ftp	Passthrough	21
ftps-data ¹	TFO+LZ+DRE	20 (source port)
FTPS ²	Passthrough	989 (source port)
ftps	TFO	990
sftp	TFO+LZ+DRE	115
TFTP	TFO+LZ+DRE	69
TFTPS	TFO	3713
Instant Messaging		

Classifier	WAAS Express Action	Destination Ports
AOL	Passthrough	5190–5193
Apple-iChat	Passthrough	5297, 5298
ircs	Passthrough	994
ircu	Passthrough	531, 6660–6665, 6667–6669
msnp	Passthrough	1863, 6891–6900
sametime	Passthrough	1533
talk	Passthrough	517
xmpp-client	Passthrough	5222
xmpp-server	Passthrough	5269
Yahoo-Messenger	Passthrough	5000, 5001, 5050, 5100
Name Services		
DNS	Passthrough	53
isns	Passthrough	3205
nameserver	Passthrough	42
netbios	Passthrough	137
svrloc	Passthrough	427
WINS	Passthrough	1512
Other (monitored)		
Basic-TCP-services	Passthrough	1–19
BGP	Passthrough	179
corba-iiop-ssl	Passthrough	684
epmap	TFO	135
msmq	TFO+LZ+DRE	1801, 2101, 2103, 2105
NTP	Passthrough	123
Other-Secure	Passthrough	261, 448
ssc-agent	TFO+LZ+DRE	2847, 2848, 2967, 2968, 38037, 38292
Peer-to-peer (P2P) (monitored)		

Classifier	WAAS Express Action	Destination Ports
BitTorrent	Passthrough	6881–6889, 6969
eDonkey	Passthrough	4661, 4662
Gnutella	Passthrough	5634, 6346–6349, 6355
Grouper	Passthrough	8038
HotLine	Passthrough	5500-5503
Kazaa	Passthrough	1214
Laplink-ShareDirect	Passthrough	2705
Napster	Passthrough	6666, 6677, 6688, 6700, 7777, 8875
Qnext	Passthrough	44, 5555
SoulSeek	Passthrough	2234, 5534
WASTE	Passthrough	1337
WinMX	Passthrough	6699
Printing (<i>monitored</i>)		
hp-pdl-datastr	TFO+LZ+DRE	9100
IPP	TFO+LZ+DRE	631
printer	TFO+LZ+DRE	515
print-srv	TFO+LZ+DRE	170
xprint-server	TFO+LZ+DRE	8100
Remote Desktop (<i>monitored</i>)		
Altiris-CarbonCopy	Passthrough	1680
citrixadmin	TFO+LZ+DRE	2513
citrixima	TFO+LZ+DRE	2512
citriximaclient	TFO+LZ+DRE	2598
ControlIT	TFO	799
Danware-NetOp	TFO	6502
ica	TFO+LZ+DRE	1494
laplink	TFO+LZ+DRE	1547

Classifier	WAAS Express Action	Destination Ports
Laplink-surfup-HTTPS	TFO	1184
ms-wbt-server	TFO	3389
net-assistant	Passthrough	3283
netrjs-3	TFO	73
pcanywheredata	TFO	5631, 5632, 65301
radmin-port	TFO	4899
Remote-Anything	TFO	3999, 4000
timbuktu	TFO	407
timbuktu-srv	TFO	1417-1420
Vmware-VMConsole	TFO	902
VNC	TFO	5800–5809, 5900–5909
x11	TFO	6000–6063
Replication (<i>monitored</i>)		
Double-Take	TFO+LZ+DRE	1100, 1105
EMC-Celerra-Replicator	TFO+LZ+DRE	8888
ms-content-repl-srv	TFO	507, 560
netapp-snapmirror	TFO+LZ+DRE	10565–10569
pcsync-http	TFO+LZ+DRE	8444
pcsync-https	TFO	8443
rrac	TFO	5678
Rsync	TFO+LZ+DRE	873
Secure Sockets Layer (SSL) (<i>monitored</i>)		
HTTPS	TFO	443
Secure Shell (SSH)		
SSH	TFO	22
Storage (<i>monitored</i>)		
EMC-SRDFA-IP	TFO+LZ+DRE	1748

Classifier	WAAS Express Action	Destination Ports
FCIP	TFO+LZ+DRE	3225
iFCP	TFO+LZ+DRE	3420
iSCSI	TFO+LZ+DRE	3260
Streaming (monitored)		
Liquid-Audio	TFO+LZ+DRE	18888
ms-streaming	TFO+LZ+DRE	1755
RTSP	TFO+LZ+DRE	554, 8554
Structured Query Language (SQL) (monitored)		
gds_db	TFO+LZ+DRE	3050
IBM-DB2	TFO+LZ+DRE	523
intersys-cache	TFO+LZ+DRE	1972
ms-olap4	TFO	2383
ms-sql-m	TFO+LZ+DRE	1434
ms-sql-s	TFO+LZ+DRE	1433
MySQL	TFO+LZ+DRE	3306
Oracle	TFO+LZ+DRE	66
orasrv	TFO+LZ+DRE	1521, 1525
Pervasive-SQL	TFO+LZ+DRE	1583
PostgreSQL	TFO+LZ+DRE	5432
sqlexec	TFO+LZ+DRE	9088, 9089
sql-net	TFO+LZ+DRE	150
sqlserv	TFO+LZ+DRE	118
sqlsrv	TFO+LZ+DRE	156
ssql	TFO+LZ+DRE	3352
sybase-sqlany	TFO+LZ+DRE	1498, 2439, 2638, 3968
UniSQL	TFO+LZ+DRE	1978, 1979
Systems Management (monitored)		

Classifier	WAAS Express Action	Destination Ports
BMC-Patrol	Passthrough	6161, 6162, 6767, 6768, 8160, 8161, 10128
eTrust-policy-Compliance	TFO	1267
flowmonitor	TFO+LZ	7878
HP-OpenView	Passthrough	7426–7431, 7501, 7510
LANDesk	TFO+LZ+DRE	9535, 9593–9595
NetIQ	Passthrough	2220, 2735, 10113-10116
Netopia-netOctopus	Passthrough	1917, 1921
netviewdm	Passthrough	729–731
novadigm	TFO+LZ+DRE	3460, 3461, 3464
novell-zen	TFO+LZ+DRE	1761–1763, 2037, 2544, 8039
objcall	TFO+LZ+DRE	94, 627, 1965, 1580, 1581
WBEM	Passthrough	5987–5990
Version Management (<i>monitored</i>)		
Clearcase	TFO+LZ+DRE	371
cvspserver	TFO+LZ+DRE	2401
VPN		
CIFS	TFO+LZ+DRE	139, 445
HTTP	TFO+LZ+DRE+HTTP-Express accelerator	80, 3128, 8000, 8080, 8088
HTTPS	TFO+LZ+DRE+SSL-Express accelerator	443
L2TP	TFO	1701
OpenVPN	TFO	1194
PPTP	TFO	1723

¹ These classifiers identify the source port instead of the destination port.

² These classifiers identify the source port instead of the destination port.

Multiple WAN Links

WAAS Express Phase 2 extends support for active/active and active/standby configurations on multiple WAN links.

IP Cisco Express Forwarding can efficiently use multiple parallel links without additional hardware multiplexers. The load balancing functionality on a device distributes packets across multiple links based on Layer 3 routing information. The load balancing functionality is inherent to the forwarding mechanism of a device. Some of the load balancing features supported by Cisco software include:

- **Per-destination load balancing:** In this type of load balancing, all packets for a given destination are forwarded along the same path. This preserves packet order, with potential unequal usage of links. If one host receives major part of traffic, all packets use one link, leaving bandwidth on other links unused. A larger number of destination addresses lead to more equally used links. You can enable per-destination load balancing on WAN interfaces by configuring the **ip load-sharing per-destination** command
- **Per-packet load balancing:** This type of load balancing guarantees equal load across all links. However, the packets may arrive out-of-order at the destination because of differential delay in the network. You can enable per-packet load balancing on WAN interfaces by configuring the **ip load-sharing per-packet** command.

Both per-destination and per-packet load balancing require the **ip cef** command to be configured.



Note

WAAS Express does not support interoperability with a per-packet load balancing configuration or any other configuration that results in sending packets from one flow to be sent to a different WAN link.

WAAS Express supports backup interface configuration. During switch over between active and standby interfaces, you will need to re-establish a session.

SNMP Support for WAAS Express

Simple Network Management Protocol (SNMP) is an interoperable, application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language that is used for monitoring and managing devices in a network.

SNMP allows for monitoring of WAAS Express. WAAS Express supports only SNMPv2c for SNMP traps; WAAS Express SNMP notification will fail if the correct SNMP version is not used.

Troubleshooting Tips

To troubleshoot SNMP WAAS Express notifications, use the following commands:

- **debug waas mibs**—Displays WAAS Express MIB-related messages.
- **debug snmp detail**—Displays SNMP debug messages.
- **debug snmp packet**—Displays information about every SNMP packet sent or received by a device.

How to Configure WAAS Express

Configuring WAN Optimization Parameters

Use the **tfo optimize** command to configure the type of compression required. The remaining steps in this task are optional.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **parameter-map type waas** *parameter-map-name*
4. **tfo optimize** {full | dre {yes | no compression {lz | none}}}
5. **tfo auto-discovery blacklist** {enable | hold-time *minutes*}
6. **cpu-threshold** *maximum-threshold*
7. **lz entropy-check**
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	parameter-map type waas <i>parameter-map-name</i> Example: Device(config)# parameter-map type waas waas_global	Configures a parameter map of the type waas and enters parameter map configuration mode. Note The only supported parameter map of the type waas is waas_global .
Step 4	tfo optimize {full dre {yes no compression {lz none}}} Example: Device(config-profile)# tfo optimize dre no compression lz	Configures the compression for WAAS Express.
Step 5	tfo auto-discovery blacklist {enable hold-time <i>minutes</i> } Example: Device(config-profile)# tfo auto-discovery blacklist hold-time 1000	(Optional) Enables, configures, and integrates a blacklist with autodiscovery for WAAS Express.

	Command or Action	Purpose
Step 6	cpu-threshold <i>maximum-threshold</i> Example: Device(config-profile)# cpu-threshold 90	(Optional) Sets the CPU threshold limit.
Step 7	lz entropy-check Example: Device(config-profile)# lz entropy-check	(Optional) Enables adaptive LZ through entropy checking.
Step 8	exit Example: Device(config-profile)# exit	Exits parameter map configuration mode.

Defining WAAS Express Policies



Note You can configure WAAS Express by using the default class and policy maps. Alternatively, you can first define the class and policy maps and then configure WAAS Express.

Perform these tasks to define class and policy maps if you do not want to use the default class and policy maps that are created when WAAS Express is enabled:

Defining Class Maps

The table in the “WAAS Application Policies” section lists the predefined applications and classifiers that WAAS will either optimize or pass through based on the policies that are provided with the system. We recommend that you review the default policies and modify them as appropriate before you create a new application policy. Often, modifying an existing policy is easier than creating a new one.

Perform this task to define a new policy (class map). Class maps help to classify traffic into groups based on a protocol, application, or other criterion. After defining a class map, associate it with a policy map. See the “Associating Class Maps with Policy Maps” task. Policy maps help to give a singular treatment to the created class maps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **class-map type waas** [match-any] *class-map-name*
4. **match tcp** {any | destination | source} {ip *ip-address* [*inverse mask*] | port *start-port-number1* [*end-port-number2*]}
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	class-map type waas [match-any] class-map-name Example: Device(config)# class-map type waas waas_global	Defines a class map of the type waas and enters QoS class-map configuration mode.
Step 4	match tcp {any destination source} {ip ip-address [inverse mask] port start-port-number1 [end-port-number2]} Example: Device(config-cmap)# match tcp destination port 7000 7009	Matches traffic based on the specified criteria. <ul style="list-style-type: none"> • any—Matches all TCP traffic. • destination—Matches TCP traffic with the specified destination IP address or port number. • source—Matches TCP traffic with the specified source IP address or port number. • ip ip-address—Refers to the IP address of the source and destination. If NAT is used, the IP address refers to the inside local address and the outside global address. • port port-number—Refers to the port number of the source and destination. <p>Note A class map of the type waas combines filters by using the match-any logical operator. The match-all logical operator is not supported by a class map of the type waas. This means that if one match criterion (filter) is matched, the entire class map is matched.</p>
Step 5	exit Example: Device(config-cmap)# exit	Exits QoS class-map configuration mode.

Example

The following example shows how to match traffic that has the destination TCP port number between 7000 and 7009:

```

Device(config)# class-map type waas waas_global
Device(config-cmap)# match tcp destination port 7000 7009
Device(config-cmap)# exit
Device(config)# class-map type waas waas_global
Device(config-cmap)# match tcp destination ip 209.165.200.225 0.0.0.31 port 80 80
Device(config-cmap)# match tcp destination ip 209.165.200.225 0.0.0.31 port 8080 8080

```

In this example, traffic is matched if either one of the following conditions is true:

- The destination IP address is in the range 209.165.200.225, and the destination TCP port is 80.
- The destination IP address is in the range 209.165.200.225, and the destination TCP port is 8080.

Associating Class Maps with Policy Maps

Before enabling WAAS Express on a device, perform this task to associate a class map with a policy map.



Note Any changes to the policy configuration (global policy map and class maps of the type waas) are persistent. For instance, if you modify the policy configuration, disable WAAS Express on interfaces, and re-enable WAAS Express, the policy configuration changes will still be visible.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map type waas** *policy-name*
4. **sequence-interval** *number*
5. **class** *class-map-name*
6. **optimize tfo** {dre | lz} **application** *application-name* **accelerate** {cifs-express | http-express }
7. **passthrough application** *application-name*
8. **end**
9. **show policy-map type waas**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	policy-map type waas <i>policy-name</i> Example:	Defines a policy map of the type waas and enters QoS policy-map configuration mode.

	Command or Action	Purpose
	Device(config)# <code>policy-map type waas waas_global</code>	
Step 4	sequence-interval <i>number</i> Example: Device(config-pmap)# <code>sequence-interval 100</code>	Assigns sequential numbers to class maps at the specified interval.
Step 5	class <i>class-map-name</i> Example: Device(config-pmap)# <code>class waas_global</code>	Specifies the class on which optimization must be performed and enters QoS policy-map class configuration mode.
Step 6	optimize tfo {dre lz} application <i>application-name</i> accelerate {cifs-express http-express } Example: Device(config-pmap-c)# <code>optimize tfo dre application web accelerate http-express</code>	Applies WAN optimization for the matching traffic.
Step 7	passthrough application <i>application-name</i> Example: Device(config-pmap-c)# <code>passthrough application web</code>	Passes through matched traffic and does not apply WAN optimization to the matching traffic. Note passthrough is the default WAN optimization for matching traffic.
Step 8	end Example: Device(config-pmap-c)# <code>end</code>	Exits QoS policy-map class configuration mode and returns to privileged EXEC mode.
Step 9	show policy-map type waas Example: Device# <code>show policy-map type waas</code>	(Optional) Displays the policy-map information.

Example

The following example shows how to create a new policy with actions and application tagging:

```
Device(config)# policy-map type waas waas_global
Device(config-pmap)# class AFS
Device(config-pmap-c)# optimize tfo lz application Web
Device(config-pmap-c)# exit
Device(config-pmap)# class Http
Device(config-pmap-c)# optimize tfo lz application Filesystem
Device(config-pmap-c)# exit
Device(config-pmap)# class class-default
Device(config-pmap-c)# exit
Device(config-pmap)# exit
```

The following output from the **show policy-map type waas** command shows the policy map created in the previous example:

```
Device# show policy-map type waas

Policy Map type waas waas_global
```

```

Class AFS
  optimize dre lz application Web
Class Http
  optimize lz application Filesystem
Class class-default

```

Troubleshooting Tips

To clear the DRE cache, enable WAAS Express and use the **no waas enable** command with the *forced* argument on the interface.

Enabling WAAS Express

The **waas enable** command must be explicitly applied on each WAN interface. You can enable WAAS Express by using either the default class and policy maps created automatically or the class and policy maps that you define.

The global policy map governs the behavior of optimization on a WAN interface. All traffic exiting or entering a WAN interface is screened for optimization as per the global policy map. WAAS Express supports flows that travel over multiple WAN interfaces, entering one interface and exiting another. Traffic on other interfaces is not optimized by WAAS Express.

Perform this task to enable WAAS Express on a WAN interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface-type/number*
4. **waas enable**
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none">• Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	interface <i>interface-type/number</i> Example: Device(config)# interface GigabitEthernet0/0	Specifies an interface for configuration and enters interface configuration mode
Step 4	waas enable Example:	Enables WAAS Express on the WAN interface.

	Command or Action	Purpose
	<code>Device(config-if)# waas enable</code>	
Step 5	exit Example: <code>Device(config-if)# exit</code>	Exits interface configuration mode.

Troubleshooting Tips

To troubleshoot the WAAS Express configuration, use the following commands:

- **debug waas**—Detects errors.
- **monitor**—Monitors and collects packet capture.
- **show waas**—Verifies the configuration.

The **no waas enable** command does not remove the WAAS Express configuration. This command neither affects the existing execute flows that are already optimized by WAAS Express on an interface, nor it removes the default maps from the WAAS device. To remove the WAAS Express configuration and disable WAAS Express, use the **no waas enable remove-config** command. To terminate the optimization flows and disable WAAS Express, use the **no waas enable forced** command.

Use the **waas config remove-all** command to remove default maps from the device. Use the **waas config restore-default** command to replace the policy configuration you defined with the default policy configuration.



Note You can use the **waas config remove-all** and **waas config restore-default** commands only if WAAS Express is disabled.

Configuring Upload DRE

Upload DRE is enabled by default. Upload DRE is a CPU-intensive operation. If you perceive that the benefits provided by upload DRE is not worth the CPU utilization, you can disable upload DRE. Perform this task to disable upload DRE.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **parameter-map type waas** *parameter-map-name*
4. **no dre upload**
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: Device> enable	<ul style="list-style-type: none"> Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	parameter-map type waas <i>parameter-map-name</i> Example: Device(config)# parameter-map type waas waas_global	Configures a parameter map of the type waas and enters parameter map configuration mode. Note The only supported parameter map of the type waas is waas_global .
Step 4	no dre upload Example: Device(config-profile)# no dre upload	Disables the upload DRE operation.
Step 5	exit Example: Device(config-profile)# exit	Exits parameter map configuration mode.

Configuring CIFS-Express Accelerator

CIFS-Express accelerator is disabled by default. Perform this task to enable CIFS-Express accelerator and configure other optimization parameters that require minimum memory and CPU resources.



Note You can configure the global CIFS-Express accelerator parameters while CIFS-Express accelerator is disabled.

SUMMARY STEPS

1. enable
2. configure terminal
3. parameter-map type waas *parameter-map-name*
4. accelerator cifs-express
5. enable
6. read-ahead enable
7. read-ahead size *value*
8. async-write enable
9. async-write quota-threshold *value*
10. ads-negative-cache enable
11. ads-negative-cache timeout *seconds*
12. end
13. show waas accelerator cifs-express

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	parameter-map type waas <i>parameter-map-name</i> Example: Device(config)# parameter-map type waas waas_global	Configures a parameter map of the type waas and enters parameter map configuration mode. Note The only supported parameter map of the type waas is waas_global .
Step 4	accelerator cifs-express Example: Device(config-profile)# accelerator cifs-express	Enters WAAS CIFS configuration mode and allows the configuration of CIFS-Express accelerator parameters.
Step 5	enable Example: Device(config-waas-cifs)# enable	Enables CIFS-Express accelerator. <ul style="list-style-type: none"> • The CIFS-Express accelerator is enabled on ports 139 and 445. • Disabling CIFS-Express accelerator will disable all CIFS-Express accelerator parameters.
Step 6	read-ahead enable Example: Device(config-waas-cifs)# read-ahead enable	(Optional) Enables the read ahead feature.
Step 7	read-ahead size <i>value</i> Example: Device(config-waas-cifs)# read-ahead size 200	(Optional) Configures the amount of data to read ahead per file.
Step 8	async-write enable Example: Device(config-waas-cifs)# async-write enable	(Optional) Enables async write operation.
Step 9	async-write quota-threshold <i>value</i> Example: Device(config-waas-cifs)# async-write quota-threshold 100	(Optional) Configures the quota threshold for async write to perform the optimization.
Step 10	ads-negative-cache enable Example:	(Optional) Enables negative caching for alternate data streams.

	Command or Action	Purpose
	Device(config-waas-cifs)# ads-negative-cache enable	
Step 11	ads-negative-cache timeout <i>seconds</i> Example: Device(config-waas-cifs)# ads-negative-cache timeout 10	(Optional) Configures the timeout value for negative caching entries.
Step 12	end Example: Device(config-waas-cifs)# end	Exits WAAS CIFS configuration mode and returns to privileged EXEC mode.
Step 13	show waas accelerator cifs-express Example: Device# show waas accelerator cifs-express	(Optional) Displays the status and configuration of all CIFS-Express accelerator parameters.

Configuring HTTP-Express Accelerator

SUMMARY STEPS

1. enable
2. configure terminal
3. parameter-map type waas *parameter-map-name*
4. accelerator http-express
5. enable
6. dre-hints enable
7. suppress-server-encoding enable
8. end
9. show waas accelerator http-express

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	parameter-map type waas <i>parameter-map-name</i> Example:	Configures a parameter map of the type waas and enters parameter map configuration mode.

	Command or Action	Purpose
	<code>Device(config)# parameter-map type waas waas_global</code>	Note The only supported parameter map of the type waas is waas_global .
Step 4	accelerator http-express Example: <code>Device(config-profile)# accelerator http-express</code>	Enters WAAS HTTP configuration mode and allows the configuration of HTTP-Express accelerator parameters.
Step 5	enable Example: <code>Device(config-waas-http)# enable</code>	Enables HTTP-Express accelerator. • Disabling HTTP-Express accelerator will disable all HTTP-Express accelerator parameters.
Step 6	dre-hints enable Example: <code>Device(config-waas-http)# dre-hints enable</code>	(Optional) Enables HTTP-Express accelerator to pass DRE hints to the DRE module.
Step 7	suppress-server-encoding enable Example: <code>Device(config-waas-http)# suppress-server-encoding enable</code>	(Optional) Suppresses server side encoding.
Step 8	end Example: <code>Device(config-waas-http)# end</code>	Exits WAAS HTTP configuration mode and returns to privileged EXEC mode.
Step 9	show waas accelerator http-express Example: <code>Device# show waas accelerator http-express</code>	(Optional) Displays the status and configuration of all HTTP-Express accelerator parameters.

Configuring HTTP Metadata Caching

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **parameter-map type waas** *parameter-map-name*
4. **accelerator http-express**
5. **enable**
6. **metadacache enable**
7. **metadacache https enable**
8. **metadacache max-age** *seconds*
9. **metadacache min-age** *num*
10. **metadacache filter-extension** *ext1, ext2,...*
11. **metadacache request-ignore-no-cache enable**

12. `metadatabacache response-ignore-no-cache enable`
13. `metadatabacache redirect-response enable`
14. `metadatabacache unauthorized-response enable`
15. `metadatabacache conditional-response enable`
16. `end`
17. `show waas cache http-express metadatabacache all`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>enable</code> Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none">• Enter your password if prompted.
Step 2	<code>configure terminal</code> Example: Device# configure terminal	Enters global configuration mode.
Step 3	<code>parameter-map type waas parameter-map-name</code> Example: Device(config)# parameter-map type waas waas_global	Configures a parameter map of the type <code>waas</code> and enters parameter map configuration mode. Note The only supported parameter map of the type <code>waas</code> is <code>waas_global</code> .
Step 4	<code>accelerator http-express</code> Example: Device(config-profile)# accelerator http-express	Enters WAAS HTTP configuration mode and allows the configuration of HTTP-Express accelerator parameters.
Step 5	<code>enable</code> Example: Device(config-waas-http)# enable	Enables HTTP-Express accelerator. <ul style="list-style-type: none">• Disabling HTTP-Express accelerator will disable all HTTP-Express accelerator parameters.
Step 6	<code>metadatabacache enable</code> Example: Device(config-waas-http)# metadatabacache enable	(Optional) Enables HTTP metadata caching.
Step 7	<code>metadatabacache https enable</code> Example: Device(config-waas-http)# metadatabacache https enable	(Optional) Enables HTTPS metadata caching.
Step 8	<code>metadatabacache max-age seconds</code> Example: Device(config-waas-http)# metadatabacache max-age 1000	(Optional) Configures the maximum time, in seconds, to retain cache entries in the metadata cache table.

	Command or Action	Purpose
Step 9	metadatatcache min-age <i>num</i> Example: Device(config-waas-http)# metadatatcache min-age 1000	(Optional) Configures the minimum time, in seconds, to retain cache entries in the metadata cache table.
Step 10	metadatatcache filter-extension <i>ext1, ext2,...</i> Example: Device(config-waas-http)# metadatatcache filter-extension html,css,jpg	(Optional) Configures the metadata cache to store only the file extensions specified in the list.
Step 11	metadatatcache request-ignore-no-cache enable Example: Device(config-waas-http)# metadatatcache request-ignore-no-cache enable	(Optional) Configures the metadata cache to ignore cache-control on requests.
Step 12	metadatatcache response-ignore-no-cache enable Example: Device(config-waas-http)# metadatatcache response-ignore-no-cache enable	(Optional) Configures the metadata cache to ignore cache-control on response.
Step 13	metadatatcache redirect-response enable Example: Device(config-waas-http)# metadatatcache redirect-response enable	(Optional) Enables the HTTP URL redirect feature.
Step 14	metadatatcache unauthorized-response enable Example: Device(config-waas-http)# metadatatcache unauthorized-response enable	(Optional) Enables the HTTP authentication-redirect feature.
Step 15	metadatatcache conditional-response enable Example: Device(config-waas-http)# metadatatcache conditional-response enable	(Optional) Enables responses for the HTTP conditional requests feature.
Step 16	end Example: Device(config-waas-http)# end	Exits WAAS HTTP configuration mode and returns to privileged EXEC mode.
Step 17	show waas cache http-express metadatatcache all Example: Device# show waas cache http-express metadatatcache all	(Optional) Displays HTTP-Express accelerator metadata cache entries.

Configuring SSL-Express Accelerator

SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `parameter-map type waas parameter-map-name`
4. `accelerator ssl-express`
5. `enable`
6. `no empty-ssl-fragment-insertion`
7. `waas-ssl-trustpoint label`
8. `cipher-list list-name`
9. `cipher cipher-suite`
10. `exit`
11. `services host-service peering`
12. `peer-ssl-version ssl-tls-version`
13. `peer-cipherlist list-name`
14. `peer-cert-verify enable`
15. `end`
16. `show waas statistics accelerator ssl-express ciphers`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>enable</code> Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none">• Enter your password if prompted.
Step 2	<code>configure terminal</code> Example: Device# configure terminal	Enters global configuration mode.
Step 3	<code>parameter-map type waas <i>parameter-map-name</i></code> Example: Device(config)# parameter-map type waas waas_global	Configures a parameter map of the type waas and enters parameter map configuration mode. Note The only supported parameter map of the type waas is waas_global .
Step 4	<code>accelerator ssl-express</code> Example: Device(config-profile)# accelerator ssl-express	Enters WAAS SSL configuration mode and allows the configuration of SSL-Express accelerator parameters.
Step 5	<code>enable</code> Example: Device(config-waas-ssl)# enable	Enables SSL-Express accelerator. <ul style="list-style-type: none">• Disabling SSL-Express accelerator will disable all SSL-Express accelerator parameters.

	Command or Action	Purpose
Step 6	<p>no empty-ssl-fragment-insertion</p> <p>Example:</p> <pre>Device(config-waas-ssl)# no empty-ssl-fragment-insertion</pre>	<p>(Optional) Disables the sending of an empty SSL fragment as the first encrypted message sent to the client.</p> <p>Note To solve issues of interoperability with older versions of client applications such as Internet Explorer 6, you can disable the sending of this empty SSL fragment with this step.</p>
Step 7	<p>waas-ssl-trustpoint <i>label</i></p> <p>Example:</p> <pre>Device(config-waas-ssl)# waas-ssl-trustpoint ssl-tp</pre>	Associates the specified trustpoint with SSL-Express accelerator.
Step 8	<p>cipher-list <i>list-name</i></p> <p>Example:</p> <pre>Device(config-waas-ssl)# cipher-list clist</pre>	Creates a cipher list for a WAAS-to-WAAS session and enters cipher list configuration mode.
Step 9	<p>cipher <i>cipher-suite</i></p> <p>Example:</p> <pre>Device(config-waas-cipher-list)# cipher rsa-with-3des-edc-cbc-sha</pre>	Adds the specified cipher suite to a cipher list.
Step 10	<p>exit</p> <p>Example:</p> <pre>Device(config-waas-cipher-list)# exit</pre>	Exits cipher list configuration mode and returns to WAAS SSL configuration mode.
Step 11	<p>services host-service peering</p> <p>Example:</p> <pre>Device(config-waas-ssl)# services host-service peering</pre>	Configures host peering service and enters SSL peering service configuration mode.
Step 12	<p>peer-ssl-version <i>ssl-tls-version</i></p> <p>Example:</p> <pre>Device(config-waas-ssl-peering)# peer-ssl-version ssl3</pre>	<p>Configures the SSL version to be used for WAAS-to-WAAS sessions.</p> <p>Note Ensure that the version configured on the WAAS Express device matches the version configured on the peer WAAS device(s).</p>
Step 13	<p>peer-cipherlist <i>list-name</i></p> <p>Example:</p> <pre>Device(config-waas-ssl-peering)# peer-cipherlist clist</pre>	<p>Configures the cipher list to be used for WAAS-to-WAAS sessions.</p> <p>Note Ensure that the cipher list configured on the WAAS Express device overlaps the one configured on the peer WAAS device(s).</p>
Step 14	<p>peer-cert-verify enable</p> <p>Example:</p>	Enables the verification of the peer certificate.

	Command or Action	Purpose
	Device(config-waas-ssl-peering)# peer-cert-verify enable	Note If the peer WAAS device is using a self-enrolled certificate, then disable the verification of the peer certification on this device; otherwise, all SSL connections being optimized with that peer will be reset due to WAAS-to-WAAS session SSL handshake failure.
Step 15	end Example: Device(config-waas-ssl-peering)# end	Exits SSL peering service configuration mode and returns to privileged EXEC mode.
Step 16	show waas statistics accelerator ssl-express ciphers Example: Device# show waas statistics accelerator ssl-express ciphers	(Optional) Displays statistics about the cipher suites being used for different SSL sessions.

Configuring SNMP Traps for WAAS Express

WAAS Express supports only SNMPv2c for SNMP traps. WAAS Express SNMP notification will fail if the correct SNMP version is not used.

SUMMARY STEPS

1. enable
2. configure terminal
3. snmp-server enable traps waas [cpu-throttle-off] [cpu-throttle-on] [license-deleted] [license-expired] [license-revoked] [peer-overload] [tfo-conn-overload]
4. snmp-server community *community-string* [ro | rw]
5. snmp-server source-interface [traps | informs] *interface*
6. snmp-server host {*hostname* | *ip-address*} [vrf *vrf-name* | informs | traps | version {1 | 2c | 3}] [auth | noauth | priv]] *community-string* [udp-port *port* [notification-type]]
7. exit
8. show snmp mib

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.

	Command or Action	Purpose
Step 3	snmp-server enable traps waas [cpu-throttle-off] [cpu-throttle-on] [license-deleted] [license-expired] [license-revoked] [peer-overload] [tfo-conn-overload] Example: Device(config)# snmp-server enable traps waas peer-overload	Enables SNMP traps for WAAS Express.
Step 4	snmp-server community <i>community-string</i> [ro rw] Example: Device(config)# snmp-server community public ro	Sets up the community access string to permit access to SNMP.
Step 5	snmp-server source-interface [traps informs] <i>interface</i> Example: Device(config)# snmp-server source-interface traps gigabitethernet5/3	Specifies the interface from which an SNMP trap originates the notifications or traps.
Step 6	snmp-server host { <i>hostname</i> <i>ip-address</i> } [vrf <i>vrf-name</i> informs traps version {1 2c 3}] [auth noauth priv]] <i>community-string</i> [udp-port <i>port</i> [<i>notification-type</i>]] Example: Device(config)# snmp-server host 10.1.1.1 version 2c public waas	Specifies the recipient of an SNMP notification. <ul style="list-style-type: none"> • WAAS Express requires version 2c to be configured.
Step 7	exit Example: Device(config)# exit	Exits global configuration mode and returns to privileged EXEC mode.
Step 8	show snmp mib Example: Device# show snmp mib	(Optional) Displays a list of MIB module instance identifiers (OIDs) registered on your device.

Configuration Examples for WAAS Express

Example: Associating Class Maps with Policy Maps

The following example shows how to create a policy map. The **insert-before** keyword will insert the class policy being created before the specified class. The **class-default** keyword helps to match unclassified packets.

```
Device(config)# policy-map type waas waas_global
Device(config-pmap)# class AFS
Device(config-pmap-c)# optimize tfo lz application Filesystem
Device(config-pmap-c)# exit
Device(config-pmap)# class Http insert-before AFS
Device(config-pmap-c)# optimize tfo lz application Web
```

Example: Configuring WAAS Express

```
Device(config-pmap-c)# exit
Device(config-pmap)# class class-default
Device(config-pmap-c)# exit
Device(config-pmap)# exit
```

The following output from the **show policy-map type waas** command shows the policy map created in the previous example:

```
Device# show policy-map type waas

Policy Map type waas waas_global
Class Http
  optimize dre lz application Web
Class AFS
  optimize lz application Filesystem
Class class-default
```

The following example shows how to create a policy map with sequence numbers:

```
Device(config)# policy-map type waas waas_global
Device(config-pmap)# sequence-interval 10
Device(config-pmap)# class AFS
Device(config-pmap-c)# optimize tfo lz application Web
Device(config-pmap-c)# exit
Device(config-pmap)# class Http
Device(config-pmap-c)# optimize tfo lz application Filesystem
Device(config-pmap-c)# exit
Device(config-pmap)# class class-default
Device(config-pmap-c)# exit
Device(config-pmap)# exit
```

The following output from the **show policy-map type waas** command shows the policy map created in the previous example:

```
Device# show policy-map type waas

Policy Map type waas waas_global
  sequence-interval 10
10 Class AFS
  optimize dre lz application Web
20 Class Http
  optimize lz application Filesystem
30 Class class-default
```

Example: Configuring WAAS Express

```
Device(config)# class-map type waas match-any http
Device(config-cmap)# match tcp destination port 80 80
Device(config-cmap)# match tcp destination port 8080 8082
Device(config-cmap)# exit
Device(config)# class-map type waas waas_global
Device(config-cmap)# match tcp destination port 5190 5193
Device(config-cmap)# exit
Device(config)# class-map type waas match-any bittorrent
Device(config-cmap)# match tcp destination port 6969
Device(config-cmap)# match tcp destination port 6881 6889
Device(config-cmap)# exit
Device(config)# policy-map type waas waas_global
Device(config-pmap)# class http
Device(config-pmap-c)# optimize tfo lz application web-traffic
Device(config-pmap-c)# exit
Device(config-pmap)# class aol
```



```

Device(config-pmap-c) # optimize tfo lz application IM
Device(config-pmap-c) # exit
Device(config-pmap) # class bittorrent
Device(config-pmap-c) # optimize tfo lz application p2p
Device(config-pmap-c) # exit
Device(config-pmap) # exit
Device(config) # interface E0
Device(config-if) # description WAN Connection
Device(config-if) # waas enable
Device(config-if) # exit

```

Example: Configuring CIFS-Express Accelerator

The following example shows how to enable CIFS-Express accelerator and configure read ahead, async write, and negative caching features:

```

Device(config) # parameter-map type waas waas_global
Device(config-profile) # accelerator cifs-express
Device(config-waas-cifs) # enable
Device(config-waas-cifs) # read-ahead size 100
Device(config-waas-cifs) # async-write quota-threshold 500
Device(config-waas-cifs) # ads-negative-cache timeout 15
Device(config-waas-cifs) # end

```

Example: Configuring HTTP-Express Accelerator

The following example shows how to enable HTTP-Express accelerator, enable server encoding suppression, and configure various metadata caching parameters:

```

Device(config) # parameter-map type waas waas_global
Device(config-profile) # accelerator http-express
Device(config-waas-http) # enable
Device(config-waas-http) # suppress-server-encoding enable
Device(config-waas-http) # metadacache enable
Device(config-waas-http) # metadacache max-age 10000
Device(config-waas-http) # metadacache min-age 100
Device(config-waas-http) # metadacache redirect-response enable
Device(config-waas-http) # metadacache unauthorized-response enable
Device(config-waas-http) # metadacache conditional-response enable
Device(config-waas-http) # end

```

Example: Configuring SSL-Express Accelerator

The following example shows how to enable SSL-Express accelerator, associate it with a trustpoint, create a cipher list, and configure peering services.

```

Device(config) # parameter-map type waas waas_global
Device(config-profile) # accelerator ssl-express
Device(config-waas-ssl) # enable
Device(config-waas-ssl) # waas-ssl-trustpoint ssl-trust
Device(config-waas-ssl) # cipher-list clist
Device(config-waas-cipher-list) # cipher rsa-with-des-cbc-sha
Device(config-waas-cipher-list) # exit
Device(config-waas-ssl) # services host-service peering
Device(config-waas-ssl-peering) # peer-ssl-version ssl3

```

Example: Configuring SNMP Traps for WAAS Express

```
Device(config-waas-ssl-peering)# peer-cipherlist clist
Device(config-waas-ssl-peering)# peer-cert-verify enable
Device(config-waas-ssl-peering)# exit
Device(config-waas-ssl)# end
```

Example: Configuring SNMP Traps for WAAS Express

```
Device(config)# snmp-server enable traps waas peer-overload
Device(config)# snmp-server community public rw
Device(config)# snmp-server host 10.0.0.0 public
Device(config)# end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Master Commands List, All Releases
WAN commands: complete command syntax, command mode, defaults, usage guidelines and examples	Wide-Area Networking Command Reference
Cisco Wide Area Application Services	<ul style="list-style-type: none"> • Cisco Wide Area Application Services Quick Configuration Guide, Software Version 4.2.1 • Cisco Wide Area Application Services Configuration Guide, Software Version 4.2.1 • Cisco WAAS Installation and Configuration Guide for ACNS on a Virtual Blade (ACNS-VB)

Standards and RFCs

Standard/RFC	Title
RFC 1323	<i>TCP Extensions for High Performance</i>
RFC 2018	<i>TCP Selective Acknowledgment Options</i>
RFC 3390	<i>Increasing TCP's Initial Window</i>

MIBs

MIB	MIBs Link
CISCO-WAN-OPTIMIZATION-MIB	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for WAAS Express

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 3: Feature Information for WAAS Express

Feature Name	Releases	Feature Information
WAAS Express	15.1(2)T	<p>Cisco's WAAS Express is a key component of the Cisco WAAS product portfolio. WAAS Express is a cost-effective, IOS-based, WAN optimization solution that increases the amount of available bandwidth for small-to-mid-size branch offices and remote locations.</p> <p>The following commands were introduced or modified: class-map type waas, clear waas, cpu-threshold, debug waas, lz entropy-check, match tcp, optimize, parameter-map type waas, passthrough, policy-map type waas, sequence-interval, show waas alarms, show waas auto-discovery, show waas connection, show waas statistics aoim, show waas statistics application, show waas statistics auto-discovery, show waas statistics class, show waas statistics dre, show waas statistics global, show waas statistics lz, show waas statistics pass-through, show waas statistics peer waas, show waas status, show waas token, tfo auto-discovery, tfo optimize, waas cm-register url, waas config, waas enable, and waas export.</p>

Feature Name	Releases	Feature Information
WAAS Express Phase 2	15.2(3)T	<p>WAAS Express Phase 2 extends the feature set of Cisco's WAAS Express to include feature interoperability and application acceleration capabilities. It introduces the following application accelerators: CIFS-Express accelerator, HTTP-Express accelerator, and SSL-Express accelerator.</p> <p>WAAS Express Phase 2 also provides multiple WAN link support and support for upload DRE.</p> <p>The following commands were introduced or modified: accelerator, ads-negative-cache, async-write, cipher, cipher-list, clear waas, debug waas, debug waas accelerator cifs-express, debug waas accelerator http-express, debug waas accelerator ssl-express, dre-hints enable, dre upload, metadatabcache, optimize tfo, peer-cert-verify enable, peer-cipherlist, peer-ssl-version, read-ahead, services host-service peering, show waas accelerator, show waas alarms, show waas cache http-express metadatabcache, show waas connection, show waas statistics accelerator, show waas statistics dre, show waas statistics errors, show waas statistics global, show waas status, snmp-server enable traps waas, suppress-server-encoding enable, and waas-ssl-trustpoint</p>



CHAPTER 3

Configuring Cisco IOS NAM PA for WAAS Express

The Cisco IOS NAM PA for WAAS Express feature is designed to analyze and measure network traffic. The Performance Agent (PA) enables you to baseline, monitor, and troubleshoot application performance. You can analyze and measure the network traffic by the Measurement, Aggregation, and Correlation Engine (MACE). The MACE measures the required metrics on a subset of traffic and exports the necessary metrics to a target. This feature enhances the WAAS Express feature by providing support for application monitoring. Monitoring capability for Wide-Area Application Services (WAAS) Express allows the analysis and measurement of TCP-based client-server messages to provide transaction- and session-based analytics. The Cisco IOS NAM PA for WAAS Express feature works independently of WAAS Express to provide users with application visibility. This concept module provides a brief overview of mDNS service discovery for kWAAS and IP networking on mDNS for kWAAS.

- [Finding Feature Information, on page 55](#)
- [Restrictions for Cisco IOS NAM PA for WAAS Express, on page 56](#)
- [Information About Cisco IOS NAM PA for WAAS Express, on page 56](#)
- [How to Configure Cisco IOS NAM PA for WAAS Express, on page 59](#)
- [Configuration Examples for Cisco IOS NAM PA for WAAS Express, on page 64](#)
- [Additional References, on page 65](#)
- [Feature Information for Cisco IOS NAM PA for WAAS Express, on page 66](#)

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see [Bug Search Tool](#) and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for Cisco IOS NAM PA for WAAS Express

MACE does not interoperate with Network Address Translation (NAT) on the ingress (LAN) interface if the **ip nat inside** command is configured on the ingress interface. However, MACE interoperates with NAT on the egress (WAN) interface if the **ip nat outside** command is configured on the egress interface.

Information About Cisco IOS NAM PA for WAAS Express

NetFlow Overview

NetFlow is a Cisco IOS application that provides statistics about packets that flow through a device.

NetFlow identifies packet flows for both ingress and egress IP packets. It does not involve any connection-setup protocol—either between devices or to any other networking device or end station. NetFlow does not require any external change—either to the packets themselves or to any networking device. NetFlow is completely transparent to the existing network, including end stations and application software and network devices such as LAN switches. Also, NetFlow capture and export operations are performed independently on each internetworking device; NetFlow need not be operational on each device in the network.

For more information, see the *NetFlow Configuration Guide*.

MACE Metrics

The Measurement, Aggregation, and Correlation Engine (MACE) provides the following metrics:

- MACE metrics—Metrics that are extracted or calculated by the MACE engine itself, such as the number of packets and bytes.
- ART metrics—Metrics that are extracted or calculated by the Application Response Time (ART) engine, such as network delay. These metrics are available only for TCP flows.
- WAAS metrics—Metrics that are extracted or calculated by Wide-Area Application Services (WAAS), such as Data Redundancy Elimination (DRE) input bytes. These metrics are available only when WAAS is configured and MACE is monitoring the WAAS traffic.

MACE Configuration Plane

The Measurement, Aggregation, and Correlation Engine (MACE) can be configured either through an independent and new policy-map type or as part of the Wide-Area Application Services (WAAS) policy.

The table below lists the categories of MACE configuration.

Table 4: MACE Configuration Categories

Configuration	Description
Global set of metrics	Metrics that need to be collected.

Configuration	Description
Filters	Subset of traffic for which metrics need be collected. You can configure the MACE to monitor specific traffic. The MACE uses filters to classify traffic that has to be analyzed.
Timers	Frequency with which data needs to be exported. You can configure timer values for exporting flow metrics. After the timer expires, flow metrics are exported using NetFlow Data Export Version 9 (NDE v9). This timer has a default value of 5 minutes.
NetFlow Collector's details	Details of the NetFlow Collector where data needs to be exported. You can configure information from the NetFlow Collector to export flow metrics. You can configure more than one exporter for the same set of metrics, in which metrics are exported to all NetFlow collectors.

The MACE collects the required metrics by using the metric template that contains a specific set of metric fields and exports them by using the Flexible NetFlow (FNF) infrastructure.

WAAS Express

The Cisco WAN optimization system consists of Wide Area Application Services (WAAS) Express devices and Wide-Area Application Engines (WAEs) that work together to optimize TCP traffic in your network. When client and server applications attempt to communicate with each other, the network intercepts the traffic and acts on behalf of the client application and the destination server. The WAAS Express devices and WAEs examine the traffic and use built-in application policies to determine whether the traffic in the network can be optimized.

WAAS Express provides the following benefits:

- Complements the Cisco WAN optimization system by adding capability to the branch devices.
- Provides branch office employees with LAN-like access to information and applications across a geographically distributed network.
- Minimizes unnecessary WAN bandwidth consumption by using advanced compression algorithms.
- Virtualizes print and other local services for branch office users.
- Improves application performance over the WAN by addressing the following issues:
 - Low data rates (constrained bandwidth)
 - Slow delivery of frames (high network latency)
 - Higher rates of packet loss (low reliability)

For more information about WAAS Express, see the *Configuring WAAS Express* module.

ART Engine

The Measurement, Aggregation, and Correlation Engine (MACE) data plane forwards packets to the Application Response Time (ART) engine in the same order in which the MACE receives them. The ART engine checks every packet forwarded by the MACE.

The ART engine saves some data from each packet in its own data structures and performs the required calculations. It aggregates the flows based on the following Layer 7 (L7) information:

- Destination address
- Destination port
- Layer 4 protocol
- Segment ID
- Source address

When the export timer expires, the ART engine provides its flows and flow metrics to the MACE Exporter.

MACE Exporter

The Measurement, Aggregation, and Correlation Engine (MACE) Exporter receives the Flexible NetFlow (FNF) templates from the MACE configuration plane and builds FNF records based on these templates. It then passes the flow templates along with each record to the NetFlow infrastructure. FNF requires these templates to understand the layout of the records so that it can export the correct fields at the time of export.

The MACE Exporter allows you to configure the export time interval. The intervals 1, 2, 5, 10, and 15, in minutes, are supported. The export timer starts when the MACE is enabled. There are two ways to enable MACE: by using the MACE policy or by using the MACE along with the WAAS policy. To synchronize the export time of multiple devices that run the MACE across the network with the collector, the export timer expires when the current time modulo configured interval is zero. For instance, if a user configures a 5 minute interval at 10:07, the first export timer will expire at 10:10 (because 10:10 modulo 5 is 0) and subsequently at a gap of every 5 minutes (10:15, 10:20, and so on).



Note Modulo is the resulting remainder when one number is divided by another. For example, the modulo of 5 and 4 is 1 because 5 divided by 4 leaves a remainder of 1.

This export mechanism ensures that the time when the first export interval expires is independent from the time when the MACE policy was applied to the target. Any future update to the timeout interval causes the current timer to stop, and a new timer starts. The timer also stops when the policy is removed from the interface.



Note The MACE Exporter works on a best-effort basis. Also, MACE being a monitoring tool, the export process does execute with a high priority.

When the MACE Exporter timer expires, all engines are notified to process the metrics. After this notification, a second set of calls are sent to collect the processed metrics. The MACE Exporter receives the metrics data from various sources, aggregates them into a single FNF record, and passes it to the NetFlow component. Aggregation is done on the basis of Layer 7 keys. Application ID (Network-Based Application Recognition [NBAR]) is provided as a metric only when requested through the configuration.

How to Configure Cisco IOS NAM PA for WAAS Express

Enabling MACE on an Interface

You can enable the Cisco IOS NAM PA for WAAS Express feature on both ingress and egress interfaces so that MACE can capture and monitor traffic in both directions. After enabling MACE in one direction, the same policy is internally configured in the other direction as well. Perform this task to enable MACE on an interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **flow record type mace** *name*
4. **collect art all**
5. **exit**
6. **flow exporter** *exporter-name*
7. **export-protocol netflow-v9**
8. **destination** *ip-address*
9. **exit**
10. **flow monitor type mace** *name*
11. **record** *record-name*
12. **exporter** *exporter-name*
13. **exit**
14. **class-map type waas** *class-map-name*
15. **exit**
16. **policy-map type mace** *name*
17. **class** *name*
18. **flow monitor** *monitor-name*
19. **exit**
20. **exit**
21. **interface** *type number* [*name-tag*]
22. **mace enable**
23. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example:	Enters global configuration mode.

	Command or Action	Purpose
	Device# configure terminal	
Step 3	flow record type mace <i>name</i> Example: Device(config)# flow record type mace my-flow-record	Configures a flow record for MACE and enters Flexible NetFlow flow record configuration mode.
Step 4	collect art <i>all</i> Example: Device(config-flow-record)# collect art all	Collects all Application Response Time (ART) metrics.
Step 5	exit Example: Device(config-flow-record)# exit	Exits Flexible NetFlow flow record configuration mode.
Step 6	flow exporter <i>exporter-name</i> Example: Device(config)# flow exporter my-flow-exporter	Creates an FNF flow exporter and enters Flexible NetFlow flow exporter configuration mode.
Step 7	export-protocol <i>netflow-v9</i> Example: Device(config-flow-exporter)# export-protocol netflow-v9	Configures NetFlow Version 9 export as the export protocol.
Step 8	destination <i>ip-address</i> Example: Device(config-flow-exporter)# destination 209.165.201.1	Configures the IP address of the workstation to which you want to send the NetFlow information.
Step 9	exit Example: Device(config-flow-exporter)# exit	Exits Flexible NetFlow flow exporter configuration mode.
Step 10	flow monitor type mace <i>name</i> Example: Device(config)# flow monitor type mace my-flow-monitor	Configures an FNF flow monitor of type MACE and enters Flexible NetFlow flow monitor configuration mode.
Step 11	record <i>record-name</i> Example:	Specifies the name of a user-defined flow record that was previously configured.

	Command or Action	Purpose
	Device(config-flow-monitor)# record my-flow-record	
Step 12	exporter <i>exporter-name</i> Example: Device(config-flow-monitor)# exporter my-flow-exporter	Specifies the name of a flow exporter that was previously configured.
Step 13	exit Example: Device(config-flow-monitor)# exit	Exits Flexible NetFlow flow monitor configuration mode.
Step 14	class-map type waas <i>class-map-name</i> Example: Device(config)# class-map type waas my-waas-class	Configures a WAAS Express class map and enters class map configuration mode.
Step 15	exit Example: Device(config-cmap)# exit	Exits class-map configuration mode.
Step 16	policy-map type mace <i>name</i> Example: Device(config)# policy-map type mace mace_global	Configures a MACE policy map and enters policy-map configuration mode.
Step 17	class <i>name</i> Example: Device(config-pmap)# class my-waas-class	Configures a class name and enters policy-map class configuration mode.
Step 18	flow monitor <i>monitor-name</i> Example: Device(config-pmap-c)# flow monitor my-flow-monitor	Configures a flow monitor name.
Step 19	exit Example: Device(config-pmap-c)# exit	Exits policy-map class configuration mode.
Step 20	exit Example:	Exits policy-map configuration mode.

	Command or Action	Purpose
	Device(config-pmap)# exit	
Step 21	interface <i>type number</i> [<i>name-tag</i>] Example: Device(config)# interface ethernet0/0	Configures an interface type and enters interface configuration mode.
Step 22	mace enable Example: Device(config-if)# mace enable	Applies the global MACE policy on an interface.
Step 23	end Example: Device(config-if)# end	Exits interface configuration mode and returns to privileged EXEC mode.

Enabling MACE on WAAS

MACE is invoked immediately before and after WAAS is enabled in both ingress and egress directions. This allows for measurements to be captured with no interference from any other feature. However, in the absence of WAAS, the before-WAAS and after-WAAS traffic is identical. Perform this task to enable MACE on WAAS.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **flow record type mace** *name*
4. **collect art all**
5. **exit**
6. **flow exporter** *exporter-name*
7. **export-protocol netflow-v9**
8. **destination** *ip-address*
9. **exit**
10. **flow monitor type mace** *name*
11. **record** *record-name*
12. **exporter** *exporter-name*
13. **exit**
14. **mace monitor waas** {**all** | **optimized**} *name*
15. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	flow record type mace <i>name</i> Example: Device(config)# flow record type mace my-flow-record	Configures a flow record for MACE and enters Flexible NetFlow flow record configuration mode.
Step 4	collect art all Example: Device(config-flow-record)# collect art all	Collects all Application Response Time (ART) metrics.
Step 5	exit Example: Device(config-flow-record)# exit	Exits Flexible NetFlow flow record configuration mode.
Step 6	flow exporter <i>exporter-name</i> Example: Device(config)# flow exporter my-flow-exporter	Creates a Flexible NetFlow flow exporter and enters Flexible NetFlow flow exporter configuration mode.
Step 7	export-protocol netflow-v9 Example: Device(config-flow-exporter)# export-protocol netflow-v9	Configures NetFlow Version 9 export as the export protocol.
Step 8	destination <i>ip-address</i> Example: Device(config-flow-exporter)# destination 209.165.201.1	Configures the IP address of the workstation to which you want to send the NetFlow information.
Step 9	exit Example:	Exits Flexible NetFlow flow exporter configuration mode.

	Command or Action	Purpose
	<code>Device(config-flow-exporter)# exit</code>	
Step 10	<p>flow monitor type mace <i>name</i></p> <p>Example:</p> <pre>Device(config)# flow monitor type mace my-flow-monitor</pre>	Configures a Flexible NetFlow flow monitor of type MACE and enters Flexible NetFlow flow monitor configuration mode.
Step 11	<p>record <i>record-name</i></p> <p>Example:</p> <pre>Device(config-flow-monitor)# record my-flow-record</pre>	Specifies the name of a user-defined flow record that was previously configured.
Step 12	<p>exporter <i>exporter-name</i></p> <p>Example:</p> <pre>Device(config-flow-monitor)# exporter my-flow-exporter</pre>	Specifies the name of a flow exporter that was previously configured.
Step 13	<p>exit</p> <p>Example:</p> <pre>Device(config-flow-monitor)# exit</pre>	Exits Flexible NetFlow flow monitor configuration mode.
Step 14	<p>mace monitor waas {all optimized} <i>name</i></p> <p>Example:</p> <pre>Device(config)# mace monitor waas all my-flow-monitor</pre>	Enables MACE on WAAS for a flow monitor that was previously configured.
Step 15	<p>end</p> <p>Example:</p> <pre>Device(config)# end</pre>	Exits global configuration mode and returns to privileged EXEC mode.

Configuration Examples for Cisco IOS NAM PA for WAAS Express

Example: Enabling MACE on an Interface

```
Device> enable
Device# configure terminal
Device(config)# flow record type mace my-flow-record
Device(config-flow-record)# collect art all
```

```

Device(config-flow-record) # exit
Device(config) # flow exporter my-flow-exporter
Device(config-flow-exporter) # export-protocol netflow-v9
Device(config-flow-exporter) # destination 209.165.201.1
Device(config-flow-exporter) # exit
Device(config) # flow monitor type mace my-flow-monitor
Device(config-flow-monitor) # record my-flow-record
Device(config-flow-monitor) # exporter my-flow-exporter
Device(config-flow-monitor) # exit
Device(config) # class-map type waas my-waas-class
Device(config-cmap) # exit
Device(config) # policy-map type mace mace_global
Device(config-pmap) # class my-waas-class
Device(config-pmap-c) # flow monitor my-flow-monitor
Device(config-pmap-c) # exit
Device(config-pmap) # exit
Device(config) # interface ethernet0/0
Device(config-if) # mace enable
Device(config-if) # end

```

Example: Enabling MACE on WAAS

```

Device> enable
Device# configure terminal
Device(config) # flow record type mace my-flow-record
Device(config-flow-record) # collect art all
Device(config-flow-record) # exit
Device(config) # flow exporter my-flow-exporter
Device(config-flow-exporter) # export-protocol netflow-v9
Device(config-flow-exporter) # destination 209.165.201.1
Device(config-flow-exporter) # exit
Device(config) # flow monitor type mace my-flow-monitor
Device(config-flow-monitor) # record my-flow-record
Device(config-flow-monitor) # exporter my-flow-exporter
Device(config-flow-monitor) # exit
Device(config) # mace monitor waas all my-flow-monitor
Device(config) # end

```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Flexible NetFlow commands	<i>Cisco IOS Flexible NetFlow Command Reference</i>
NetFlow configuration tasks	<i>Cisco IOS NetFow Configuration Guide</i>

Related Topic	Document Title
WAN configuration tasks	<ul style="list-style-type: none"> • <i>Wide-Area Networking Configuration Guide: Frame Relay</i> • <i>Wide-Area Networking Configuration Guide: Layer 2 Services</i> • <i>Wide-Area Networking Configuration Guide: SMDS and X.25 and LAPB</i> • <i>Wide-Area Networking Configuration Guide: Wide-Area Application Services</i>
WAN commands	<i>Cisco IOS Wide-Area Networking Command Reference</i>

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Cisco IOS NAM PA for WAAS Express

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 5: Feature Information for Configuring Cisco IOS NAM PA for WAAS Express

Feature Name	Releases	Feature Information
Cisco IOS NAM PA for WAAS Express	15.1(4)M	<p>The Cisco IOS NAM PA for WAAS Express feature is designed to analyze and measure network traffic. The Performance Agent (PA) enables you to baseline, monitor, and troubleshoot application performance.</p> <p>The following commands were introduced or modified: collect art, collect waas, flow monitor type mace, flow record type mace, mace enable, mace monitor waas, policy-map type mace, show flow monitor type mace, show flow record type, show mace metrics, show policy-map type mace.</p>



CHAPTER 4

Configuring AVC to Monitor MACE Metrics

This feature is designed to analyze and measure network traffic for WAAS Express.

Application Visibility and Control (AVC) provides visibility for various applications and the network to central network management stations. MACE (Measurement, Aggregation, and Correlation Engine) provides AVC services by measuring metrics on a subset of traffic and exporting those metrics to a target. This enables the traffic to be measured and analyzed and the applications' performance to be base-lined, monitored, and troubleshot .

This feature expands on the original enhancement of the WAAS Express feature that provided support for application monitoring. Monitoring capability for Wide-Area Application Services (WAAS) Express allows the analysis and measurement of TCP-based client-server messages to provide transaction- and session-based analytics. This feature works independently of WAAS Express to provide users with application visibility.

- [Finding Feature Information, on page 67](#)
- [Restrictions for Configuring AVC to Monitor MACE Metrics, on page 67](#)
- [Information about Configuring AVC to Monitor MACE Metrics, on page 68](#)
- [How to Configure AVC to Monitor MACE Metrics, on page 72](#)
- [Additional References, on page 82](#)
- [Feature Information for Configuring AVC to Monitor MACE Metrics , on page 82](#)

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to <http://www.cisco.com/go/cfn> . An account on Cisco.com is not required.

Restrictions for Configuring AVC to Monitor MACE Metrics

MACE does not interoperate with Network Address Translation (NAT) on the ingress (LAN) interface if the **ip nat inside** command is configured on the ingress interface. However, MACE interoperates with NAT on the egress (WAN) interface if the **ip nat outside** command is configured on the egress interface.

Information about Configuring AVC to Monitor MACE Metrics

New Functionality for MACE Phase 2

Phase 2 of Measurement, Aggregation, and Correlation Engine (MACE) provides the following additional support:

- Monitoring of IPv6 flows
- MACE metrics for UDP flows.
- Two new NBAR option templates
- New option templates for class and policy information
- Use of the IPFIX protocol for flow exporters

The following collect commands can now be used to monitor IPv6 flows

- collect art response time sum
- collect art response time minimum
- collect art response time maximum
- collect art server response time sum
- collect art server response time minimum
- collect art server response time maximum
- collect art network time sum
- collect art network time minimum
- collect art network time maximum
- collect art client network time sum
- collect art client network time minimum
- collect art client network time maximum
- collect art server network time sum
- collect art server network time minimum
- collect art server network time maximum
- collect art total response time sum
- collect art total response time minimum
- collect art total response time maximum
- collect art total transaction time sum
- collect art total transaction time minimum

- collect art total transaction time maximum
- collect art count transactions
- collect art server packets
- collect art server bytes
- collect art count retrans
- collect art client packets
- collect art client bytes
- collect art count new connections
- collect art count responses
- collect art count late responses
- collect art count responses histogram
- collect art all
- collect datalink mac source address input
- collect ip dscp
- collect application name
- collect counter client bytes
- collect counter server bytes
- collect counter client packets
- collect counter server packets
- collect application http uri statistics
- collect application http host
- collect policy qos classification hierarchy
- collect policy qos queue drops
- collect time inter-packet-gap histogram

The following commands for new option templates are now supported

- option application-attributes
- option sub-application-table
- option class-qos-table
- option policy-qos-table

NetFlow Overview

NetFlow is a Cisco IOS application that provides statistics about packets that flow through a device.

NetFlow identifies packet flows for both ingress and egress IP packets. It does not involve any connection-setup protocol—either between devices or to any other networking device or end station. NetFlow does not require any external change—either to the packets themselves or to any networking device. NetFlow is completely transparent to the existing network, including end stations and application software and network devices such as LAN switches. Also, NetFlow capture and export operations are performed independently on each internetworking device; NetFlow need not be operational on each device in the network.

For more information, see the *NetFlow Configuration Guide*.

MACE Metrics

The Measurement, Aggregation, and Correlation Engine (MACE) provides the following metrics:

- MACE metrics—Metrics that are extracted or calculated by the MACE engine itself, such as the number of packets and bytes.
- ART metrics—Metrics that are extracted or calculated by the Application Response Time (ART) engine, such as network delay. These metrics are available only for TCP flows.
- WAAS metrics—Metrics that are extracted or calculated by Wide-Area Application Services (WAAS), such as Data Redundancy Elimination (DRE) input bytes. These metrics are available only when WAAS is configured and MACE is monitoring the WAAS traffic.

MACE Configuration Plane

The Measurement, Aggregation, and Correlation Engine (MACE) can be configured either through an independent and new policy-map type or as part of the Wide-Area Application Services (WAAS) policy.

The table below lists the categories of MACE configuration.

Table 6: MACE Configuration Categories

Configuration	Description
Global set of metrics	Metrics that need to be collected.
Filters	Subset of traffic for which metrics need be collected. You can configure the MACE to monitor specific traffic. The MACE uses filters to classify traffic that has to be analyzed.
Timers	Frequency with which data needs to be exported. You can configure timer values for exporting flow metrics. After the timer expires, flow metrics are exported using NetFlow Data Export Version 9 (NDE v9). This timer has a default value of 5 minutes.
NetFlow Collector's details	Details of the NetFlow Collector where data needs to be exported. You can configure information from the NetFlow Collector to export flow metrics. You can configure more than one exporter for the same set of metrics, in which metrics are exported to all NetFlow collectors.

The MACE collects the required metrics by using the metric template that contains a specific set of metric fields and exports them by using the Flexible NetFlow (FNF) infrastructure.

WAAS Express

Cisco's WAAS Express software interoperates with WAN optimization headend applications from Cisco. Cisco WAAS Express improves WAN access and use by optimizing applications, such as backup (is backup an application or a mechanism?), that require high bandwidth or are bound to a LAN.

WAAS Express helps enterprises meet the following objectives:

- Complement the Cisco WAN optimization system by adding the capability to branch routers.
- Provide branch office employees with LAN-like access to information and applications across a geographically distributed network.
- Minimize unnecessary WAN bandwidth consumption through the use of advanced compression algorithms.
- Virtualize print and other local services to branch office users.
- Improve application performance over WAN by addressing the following common issues:
 - Low data rates (constrained bandwidth)
 - Slow delivery of frames (high network latency)
 - Higher rates of packet loss (low reliability)

The Network Analysis Module (NAM) Performance Agent (PA) for WAAS Express analyzes and measures network traffic. The PA enables baselining, monitoring, and troubleshooting of application performance. The analysis and measurement of network traffic is done by the Measurement, Aggregation, and Correlation Engine (MACE). MACE performs the required measurements on a subset of traffic and exports the necessary metrics to a target.

ART Engine

The Measurement, Aggregation, and Correlation Engine (MACE) data plane forwards packets to the Application Response Time (ART) engine in the same order in which the MACE receives them. The ART engine checks every packet forwarded by the MACE.

The ART engine saves some data from each packet in its own data structures and performs the required calculations. It aggregates the flows based on the following Layer 7 (L7) information:

- Destination address
- Destination port
- Layer 4 protocol
- Segment ID
- Source address

When the export timer expires, the ART engine provides its flows and flow metrics to the MACE Exporter.

MACE Exporter

The Measurement, Aggregation, and Correlation Engine (MACE) Exporter receives the Flexible NetFlow (FNF) templates from the MACE configuration plane and builds FNF records based on these templates. It then passes the flow templates along with each record to the NetFlow infrastructure. FNF requires these templates to understand the layout of the records so that it can export the correct fields at the time of export.

The MACE Exporter allows you to configure the export time interval. The intervals 1, 2, 5, 10, and 15, in minutes, are supported. The export timer starts when the MACE is enabled. There are two ways to enable MACE: by using the MACE policy or by using the MACE along with the WAAS policy. To synchronize the export time of multiple devices that run the MACE across the network with the collector, the export timer expires when the current time modulo configured interval is zero. For instance, if a user configures a 5 minute interval at 10:07, the first export timer will expire at 10:10 (because 10:10 modulo 5 is 0) and subsequently at a gap of every 5 minutes (10:15, 10:20, and so on).



Note Modulo is the resulting remainder when one number is divided by another. For example, the modulo of 5 and 4 is 1 because 5 divided by 4 leaves a remainder of 1.

This export mechanism ensures that the time when the first export interval expires is independent from the time when the MACE policy was applied to the target. Any future update to the timeout interval causes the current timer to stop, and a new timer starts. The timer also stops when the policy is removed from the interface.



Note The MACE Exporter works on a best-effort basis. Also, MACE being a monitoring tool, the export process does execute with a high priority.

When the MACE Exporter timer expires, all engines are notified to process the metrics. After this notification, a second set of calls are sent to collect the processed metrics. The MACE Exporter receives the metrics data from various sources, aggregates them into a single FNF record, and passes it to the NetFlow component. Aggregation is done on the basis of Layer 7 keys. Application ID (Network-Based Application Recognition [NBAR]) is provided as a metric only when requested through the configuration.

How to Configure AVC to Monitor MACE Metrics

Configuring MACE for WAAS

MACE phase 2 can be invoked immediately before and after WAAS is enabled in both ingress and egress directions. This allows for measurements to be captured with no interference from any other feature. However, in the absence of WAAS, the before-WAAS and after-WAAS traffic is identical. Perform this task to enable MACE phase 2 on WAAS.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **flow record type mace *name***

4. **collect art all**
5. **collect application http host**
6. **collect application http uri statistics**
7. **collect policy qos classification hierarchy**
8. **collect policy qos queue drops**
9. **collect time inter-packet-gap histogram**
10. **exit**
11. **flow exporter *exporter-name***
12. **export-protocol ipfix**
13. **option application-attributes**
14. **option sub-application-table**
15. **option class-qos-table**
16. **option policy-qos-table**
17. **destination *ip-address***
18. **exit**
19. **flow monitor type mace *name***
20. **record *record-name***
21. **exporter *exporter-name***
22. **exit**
23. **mace monitor waas {all | optimized} *name***
24. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	flow record type mace <i>name</i> Example: Device(config)# flow record type mace my-flow-record	Configures a flow record for MACE and enters Flexible NetFlow flow record configuration mode.
Step 4	collect art all Example: Device(config-flow-record)# collect art all	Collects all Application Response Time (ART) metrics.

	Command or Action	Purpose
Step 5	collect application http host Example: Device(config-flow-record)# collect http host	Collects all Application Response Time (ART) metrics.
Step 6	collect application http uri statistics Example: Device(config-flow-record)# collect http uri statistics	Collects application HTTP URI statistics.
Step 7	collect policy qos classification hierarchy Example: Device(config-flow-record)# collect policy qos classification hierarchy	Collects the QoS policy classification hierarchy.
Step 8	collect policy qos queue drops Example: Device(config-flow-record)# collect policy qos queue drops	Collects the number of QoS policy queue drops.
Step 9	collect time inter-packet-gap histogram Example: Device(config-flow-record)# collect time inter-packet-gap histogram	Collects the inter-packet-gap time histogram.
Step 10	exit Example: Device(config-flow-record)# exit	Exits Flexible NetFlow flow record configuration mode.
Step 11	flow exporter <i>exporter-name</i> Example: Device(config)# flow exporter my-flow-exporter	Creates a Flexible NetFlow flow exporter and enters Flexible NetFlow flow exporter configuration mode.
Step 12	export-protocol ipfix Example: Device(config-flow-exporter)# export-protocol ipfix	Configures IPFIX as the export protocol.
Step 13	option application-attributes Example:	Configures an option template.

	Command or Action	Purpose
	Device(config-flow-exporter)# option application-attributes	
Step 14	option sub-application-table Example: Device(config-flow-exporter)# option sub-application-table	Configures an option template.
Step 15	option class-qos-table Example: Device(config-flow-exporter)# option class-qos-table	Configures an option template.
Step 16	option policy-qos-table Example: Device(config-flow-exporter)# option policy-qos-table	Configures an option template.
Step 17	destination ip-address Example: Device(config-flow-exporter)# destination 209.165.201.1	Configures the IP address of the workstation to which you want to send the NetFlow information.
Step 18	exit Example: Device(config-flow-exporter)# exit	Exits Flexible NetFlow flow exporter configuration mode.
Step 19	flow monitor type mace name Example: Device(config)# flow monitor type mace my-flow-monitor	Configures a Flexible NetFlow flow monitor of type MACE and enters Flexible NetFlow flow monitor configuration mode.
Step 20	record record-name Example: Device(config-flow-monitor)# record my-flow-record	Specifies the name of a user-defined flow record that was previously configured.

	Command or Action	Purpose
Step 21	exporter <i>exporter-name</i> Example: <pre>Device(config-flow-monitor)# exporter my-flow-exporter</pre>	Specifies the name of a flow exporter that was previously configured.
Step 22	exit Example: <pre>Device(config-flow-monitor)# exit</pre>	Exits Flexible NetFlow flow monitor configuration mode.
Step 23	mace monitor waas {all optimized} <i>name</i> Example: <pre>Device(config)# mace monitor waas all my-flow-monitor</pre>	Enables MACE on WAAS for a flow monitor that was previously configured.
Step 24	end Example: <pre>Device(config)# end</pre>	Exits global configuration mode and returns to privileged EXEC mode.

Configuring MACE for an Interface

You can enable the Cisco IOS NAM PA for WAAS Express feature on both ingress and egress interfaces so that MACE can capture and monitor traffic in both directions. After enabling MACE in one direction, the same policy is internally configured in the other direction as well. Perform this task to enable MACE on an interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **flow record type mace** *name*
4. **collect art** all
5. **collect application** *name*
6. **collect counter** client bytes
7. **collect counter** server bytes
8. **collect counter** client packets
9. **collect counter** client packets
10. **collect application** http host
11. **collect application** http uri statistics
12. **collect policy** qos classification hierarchy
13. **collect policy** qos queue drops
14. **collect time** inter-packet-gap histogram
15. **exit**

16. **flow exporter** *exporter-name*
17. **export-protocol** **ipfix**
18. **option** **application-attributes**
19. **option** **sub-application-table**
20. **option** **class-qos-table**
21. **option** **policy-qos-table**
22. **destination** *ip-address*
23. **exit**
24. **flow monitor type mace** *name*
25. **record** *record-name*
26. **exporter** *exporter-name*
27. **exit**
28. **class-map type waas** *class-map-name*
29. **exit**
30. **policy-map type mace** *name*
31. **class** *name*
32. **flow monitor** *monitor-name*
33. **exit**
34. **exit**
35. **interface** *type number* [*name-tag*]
36. **mace enable**
37. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	flow record type mace <i>name</i> Example: Device(config)# flow record type mace my-flow-record	Configures a flow record for MACE and enters Flexible NetFlow flow record configuration mode.
Step 4	collect art all Example: Device(config-flow-record)# collect art all	Collects all Application Response Time (ART) metrics.

	Command or Action	Purpose
Step 5	collect application name Example: <pre>Device(config-flow-record)# collect application name</pre>	Collects the application name.
Step 6	collect counter client bytes Example: <pre>Device(config-flow-record)# collect counter client bytes</pre>	Collects the total number of bytes from the client.
Step 7	collect counter server bytes Example: <pre>Device(config-flow-record)# collect counter server bytes</pre>	Collects the total number of bytes from the server.
Step 8	collect counter client packets Example: <pre>Device(config-flow-record)# collect counter client packets</pre>	Collects the total number of bytes from the server.
Step 9	collect counter client packets Example: <pre>Device(config-flow-record)# collect counter server packets</pre>	Collects the total number of packets from the server.
Step 10	collect application http host Example: <pre>Device(config-flow-record)# collect http host</pre>	Collects all Application Response Time (ART) metrics.
Step 11	collect application http uri statistics Example: <pre>Device(config-flow-record)# collect http uri statistics</pre>	Collects application HTTP URI statistics.
Step 12	collect policy qos classification hierarchy Example: <pre>Device(config-flow-record)# collect policy qos classification hierarchy</pre>	Collects the QoS policy classification hierarchy.

	Command or Action	Purpose
Step 13	collect policy qos queue drops Example: <pre>Device(config-flow-record)# collect policy qos queue drops</pre>	Collects the number of QoS policy queue drops.
Step 14	collect time inter-packet-gap histogram Example: <pre>Device(config-flow-record)# collect time inter-packet-gap histogram</pre>	Collects the inter-packet-gap time histogram.
Step 15	exit Example: <pre>Device(config-flow-record)# exit</pre>	Exits Flexible NetFlow flow record configuration mode.
Step 16	flow exporter <i>exporter-name</i> Example: <pre>Device(config)# flow exporter my-flow-exporter</pre>	Creates an FNF flow exporter and enters Flexible NetFlow flow exporter configuration mode.
Step 17	export-protocol ipfix Example: <pre>Device(config-flow-exporter)# export-protocol ipfix</pre>	Configures IPFIX as the export protocol.
Step 18	option application-attributes Example: <pre>Device(config-flow-exporter)# option application-attributes</pre>	Configures an option template.
Step 19	option sub-application-table Example: <pre>Device(config-flow-exporter)# option sub-application-table</pre>	Configures an option template.
Step 20	option class-qos-table Example: <pre>Device(config-flow-exporter)# option class-qos-table</pre>	Configures an option template.

	Command or Action	Purpose
Step 21	option policy-qos-table Example: <pre>Device(config-flow-exporter)# option policy-qos-table</pre>	Configures an option template.
Step 22	destination ip-address Example: <pre>Device(config-flow-exporter)# destination 209.165.201.1</pre>	Configures the IP address of the workstation to which you want to send the NetFlow information.
Step 23	exit Example: <pre>Device(config-flow-exporter)# exit</pre>	Exits Flexible NetFlow flow exporter configuration mode.
Step 24	flow monitor type mace name Example: <pre>Device(config)# flow monitor type mace my-flow-monitor</pre>	Configures an FNF flow monitor of type MACE and enters Flexible NetFlow flow monitor configuration mode.
Step 25	record record-name Example: <pre>Device(config-flow-monitor)# record my-flow-record</pre>	Specifies the name of a user-defined flow record that was previously configured.
Step 26	exporter exporter-name Example: <pre>Device(config-flow-monitor)# exporter my-flow-exporter</pre>	Specifies the name of a flow exporter that was previously configured.
Step 27	exit Example: <pre>Device(config-flow-monitor)# exit</pre>	Exits Flexible NetFlow flow monitor configuration mode.
Step 28	class-map type waas class-map-name Example: <pre>Device(config)# class-map type waas my-waas-class</pre>	Configures a WAAS Express class map and enters class map configuration mode.

	Command or Action	Purpose
Step 29	exit Example: Device(config-cmap)# exit	Exits class-map configuration mode.
Step 30	policy-map type mace name Example: Device(config)# policy-map type mace mace_global	Configures a MACE policy map and enters policy-map configuration mode.
Step 31	class name Example: Device(config-pmap)# class my-waas-class	Configures a class name and enters policy-map class configuration mode.
Step 32	flow monitor monitor-name Example: Device(config-pmap-c)# flow monitor my-flow-monitor	Configures a flow monitor name.
Step 33	exit Example: Device(config-pmap-c)# exit	Exits policy-map class configuration mode.
Step 34	exit Example: Device(config-pmap)# exit	Exits policy-map configuration mode.
Step 35	interface type number [name-tag] Example: Device(config)# interface ethernet0/0	Configures an interface type and enters interface configuration mode.
Step 36	mace enable Example: Device(config-if)# mace enable	Applies the global MACE policy on an interface.
Step 37	end Example: Device(config-if)# end	Exits interface configuration mode and returns to privileged EXEC mode.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Flexible NetFlow commands	<i>Cisco IOS Flexible NetFlow Command Reference</i>
NetFlow configuration tasks	<i>Cisco IOS NetFow Configuration Guide</i>
WAN configuration tasks	<ul style="list-style-type: none"> • <i>Wide-Area Networking Configuration Guide: Frame Relay</i> • <i>Wide-Area Networking Configuration Guide: Layer 2 Services</i> • <i>Wide-Area Networking Configuration Guide: SMDS and X.25 and LAPB</i> • <i>Wide-Area Networking Configuration Guide: Wide-Area Application Services</i>
WAN commands	<i>Cisco IOS Wide-Area Networking Command Reference</i>

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Configuring AVC to Monitor MACE Metrics

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 7: Feature Information for MACE Phase 2

Feature Name	Releases	Feature Information
MACE Phase 2	15.1(4)M2	<p>This feature provides support for IPv6 flows, MACE metrics for UDP flows, two new NBAR option templates, new option templates for class and policy information, and the use of IPFIX for flow exporters.</p> <p>The following commands were introduced or modified: collect application http host, collect application http uri statistics, collect policy qos classification hierarchy, collect policy qos queue drops, collect time inter-packet-gap histogram, export-protocol ipfix, option application-attributes, option sub-application-table, option class-qos-table, and option policy-qos-table.</p>

