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

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 logos 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 *(monitored)*		
dmdocbroker	TFO+LZ+DRE	1489
Filenet	TFO+LZ+DRE	32768–32774
Directory Services *(monitored)*		
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
nntps	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
QMTP	TFO+LZ+DRE	209
smtp	TFO+LZ+DRE	25
smtps	TFO	465
Enterprise Applications *(monitored)*		
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 *(monitored)*		
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 *(monitored)*		
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 *(monitored)*		
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 *(monitored)*		
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 *(monitored)*		
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) *(monitored)*		
HTTPS	TFO	443
Secure Shell (SSH)		
SSH	TFO	22
Storage *(monitored)*		
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 *(monitored)*		
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** *parameter-map-name* **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** *minutes*} **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** *maximum-threshold* **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. • 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. • **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. **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.
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. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**policy-map type waas** *policy-name* **Example:**	Defines a policy map of the type waas and enters QoS policy-map configuration mode.

	Command or Action	Purpose
	`Device(config)# policy-map type waas waas_global`	
Step 4	**sequence-interval** *number* **Example:** `Device(config-pmap)# sequence-interval 100`	Assigns sequential numbers to class maps at the specified interval.
Step 5	**class** *class-map-name* **Example:** `Device(config-pmap)# class waas_global`	Specifies the class on which optimization must be performed and enters QoS policy-map class configuration mode.
Step 6	**optimize tfo** {**dre** | **lz**} **application** *application-name* **accelerate** {**cifs-express** | **http-express** } **Example:** `Device(config-pmap-c)# optimize tfo dre application web accelerate http-express`	Applies WAN optimization for the matching traffic.
Step 7	**passthrough application** *application-name* **Example:** `Device(config-pmap-c)# passthrough application web`	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)# end`	Exits QoS policy-map class configuration mode and returns to privileged EXEC mode.
Step 9	**show policy-map type waas** **Example:** `Device# show policy-map type waas`	(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. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-type/number* **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
	`Device(config-if)# waas enable`	
Step 5	**exit** **Example:** `Device(config-if)# exit`	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`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**parameter-map type waas** *parameter-map-name* **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. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**parameter-map type waas** *parameter-map-name* **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. • 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** *value* **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** *value* **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** *seconds* **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. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**parameter-map type waas** *parameter-map-name* **Example:**	Configures a parameter map of the type waas and enters parameter map configuration mode.

	Command or Action	Purpose
	`Device(config)# parameter-map type waas waas_global`	**Note** The only supported parameter map of the type waas is **waas_global**.
Step 4	**accelerator http-express** **Example:** `Device(config-profile)# accelerator http-express`	Enters WAAS HTTP configuration mode and allows the configuration of HTTP-Express accelerator parameters.
Step 5	**enable** **Example:** `Device(config-waas-http)# enable`	Enables HTTP-Express accelerator. • Disabling HTTP-Express accelerator will disable all HTTP-Express accelerator parameters.
Step 6	**dre-hints enable** **Example:** `Device(config-waas-http)# dre-hints enable`	(Optional) Enables HTTP-Express accelerator to pass DRE hints to the DRE module.
Step 7	**suppress-server-encoding enable** **Example:** `Device(config-waas-http)# suppress-server-encoding enable`	(Optional) Suppresses server side encoding.
Step 8	**end** **Example:** `Device(config-waas-http)# end`	Exits WAAS HTTP configuration mode and returns to privileged EXEC mode.
Step 9	**show waas accelerator http-express** **Example:** `Device# show waas accelerator http-express`	(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. **metadatacache enable**
7. **metadatacache https enable**
8. **metadatacache max-age** *seconds*
9. **metadatacache min-age** *num*
10. **metadatacache filter-extension** *ext1*, *ext2*,...
11. **metadatacache request-ignore-no-cache enable**

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

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** *parameter-map-name* **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 http-express** **Example:** `Device(config-profile)# accelerator http-express`	Enters WAAS HTTP configuration mode and allows the configuration of HTTP-Express accelerator parameters.
Step 5	**enable** **Example:** `Device(config-waas-http)# enable`	Enables HTTP-Express accelerator. • Disabling HTTP-Express accelerator will disable all HTTP-Express accelerator parameters.
Step 6	**metadatacache enable** **Example:** `Device(config-waas-http)# metadatacache enable`	(Optional) Enables HTTP metadata caching.
Step 7	**metadatacache https enable** **Example:** `Device(config-waas-http)# metadatacache https enable`	(Optional) Enables HTTPS metadata caching.
Step 8	**metadatacache max-age** *seconds* **Example:** `Device(config-waas-http)# metadatacache 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	**metadatacache min-age** *num* **Example:** `Device(config-waas-http)# metadatacache min-age 1000`	(Optional) Configures the minimum time, in seconds, to retain cache entries in the metadata cache table.
Step 10	**metadatacache filter-extension** *ext1*, *ext2*,... **Example:** `Device(config-waas-http)# metadatacache filter-extension html,css,jpg`	(Optional) Configures the metadata cache to store only the file extensions specified in the list.
Step 11	**metadatacache request-ignore-no-cache enable** **Example:** `Device(config-waas-http)# metadatacache request-ignore-no-cache enable`	(Optional) Configures the metadata cache to ignore cache-control on requests.
Step 12	**metadatacache response-ignore-no-cache enable** **Example:** `Device(config-waas-http)# metadatacache response-ignore-no-cache enable`	(Optional) Configures the metadata cache to ignore cache-control on response.
Step 13	**metadatacache redirect-response enable** **Example:** `Device(config-waas-http)# metadatacache redirect-response enable`	(Optional) Enables the HTTP URL redirect feature.
Step 14	**metadatacache unauthorized-response enable** **Example:** `Device(config-waas-http)# metadatacache unauthorized-response enable`	(Optional) Enables the HTTP authentication-redirect feature.
Step 15	**metadatacache conditional-response enable** **Example:** `Device(config-waas-http)# metadatacache 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 metadatacache all** **Example:** `Device# show waas cache http-express metadatacache 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	**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** *parameter-map-name* **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 ssl-express** **Example:** `Device(config-profile)# accelerator ssl-express`	Enters WAAS SSL configuration mode and allows the configuration of SSL-Express accelerator parameters.
Step 5	**enable** **Example:** `Device(config-waas-ssl)# enable`	Enables SSL-Express accelerator. • Disabling SSL-Express accelerator will disable all SSL-Express accelerator parameters.

	Command or Action	Purpose
Step 6	**no empty-ssl-fragment-insertion** **Example:** `Device(config-waas-ssl)# no empty-ssl-fragment-insertion`	(Optional) Disables the sending of an empty SSL fragment as the first encrypted message sent to the client. **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.
Step 7	**waas-ssl-trustpoint** *label* **Example:** `Device(config-waas-ssl)# waas-ssl-trustpoint ssl-tp`	Associates the specified trustpoint with SSL-Express accelerator.
Step 8	**cipher-list** *list-name* **Example:** `Device(config-waas-ssl)# cipher-list clist`	Creates a cipher list for a WAAS-to-WAAS session and enters cipher list configuration mode.
Step 9	**cipher** *cipher-suite* **Example:** `Device(config-waas-cipher-list)# cipher rsa-with-3des-ede-cbc-sha`	Adds the specified cipher suite to a cipher list.
Step 10	**exit** **Example:** `Device(config-waas-cipher-list)# exit`	Exits cipher list configuration mode and returns to WAAS SSL configuration mode.
Step 11	**services host-service peering** **Example:** `Device(config-waas-ssl)# services host-service peering`	Configures host peering service and enters SSL peering service configuration mode.
Step 12	**peer-ssl-version** *ssl-tls-version* **Example:** `Device(config-waas-ssl-peering)# peer-ssl-version ssl3`	Configures the SSL version to be used for WAAS-to-WAAS sessions. **Note** Ensure that the version configured on the WAAS Express device matches the version configured on the peer WAAS device(s).
Step 13	**peer-cipherlist** *list-name* **Example:** `Device(config-waas-ssl-peering)# peer-cipherlist clist`	Configures the cipher list to be used for WAAS-to-WAAS sessions. **Note** Ensure that the cipher list configured on the WAAS Express device overlaps the one configured on the peer WAAS device(s).
Step 14	**peer-cert-verify enable** **Example:**	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. • 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** *community-string* [**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**] *interface* **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** {*hostname* | *ip-address*} [**vrf** *vrf-name* | **informs** | **traps** | **version** {**1** | **2c** | **3**} [**auth** | **noauth** | **priv**]] *community-string* [**udp-port** *port* [*notification-type*]] **Example:** `Device(config)# snmp-server host 10.1.1.1 version 2c public waas`	Specifies the recipient of an SNMP notification. • 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
```

```
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)# metadatacache enable
Device(config-waas-http)# metadatacache max-age 10000
Device(config-waas-http)# metadatacache min-age 100
Device(config-waas-http)# metadatacache redirect-response enable
Device(config-waas-http)# metadatacache unauthorized-response enable
Device(config-waas-http)# metadatacache 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
```

```
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	• 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	*TCP Extensions for High Performance*
RFC 2018	*TCP Selective Acknowledgment Options*
RFC 3390	*Increasing TCP's Initial Window*

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	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. 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**.

Feature Name	Releases	Feature Information
WAAS Express Phase 2	15.2(3)T	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. WAAS Express Phase 2 also provides multiple WAN link support and support for upload DRE. 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**, **metadatacache**, **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 metadatacache**, **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**