riverbed®

SteelHead™ Deployment Guide - Protocols

July 2019

© 2019 Riverbed Technology, Inc. All rights reserved.

Riverbed and any Riverbed product or service name or logo used herein are trademarks of Riverbed. All other trademarks used herein belong to their respective owners. The trademarks and logos displayed herein cannot be used without the prior written consent of Riverbed or their respective owners.

Akamai® and the Akamai wave logo are registered trademarks of Akamai Technologies, Inc. SureRoute is a service mark of Akamai. Apple and Mac are registered trademarks of Apple, Incorporated in the United States and in other countries. Cisco is a registered trademark of Cisco Systems, Inc. and its affiliates in the United States and in other countries. EMC, Symmetrix, and SRDF are registered trademarks of EMC Corporation and its affiliates in the United States and in other countries. IBM, iSeries, and AS/400 are registered trademarks of IBM Corporation and its affiliates in the United States and in other countries. Juniper Networks and Junos are registered trademarks of Juniper Networks, Incorporated in the United States and other countries. Linux is a trademark of Linus Torvalds in the United States and in other countries. Microsoft, Windows, Vista, Outlook, and Internet Explorer are trademarks or registered trademarks of Microsoft Corporation in the United States and in other countries. Oracle and JInitiator are trademarks or registered trademarks of Oracle Corporation in the United States and in other countries. UNIX is a registered trademark in the United States and in other countries, exclusively licensed through X/Open Company, Ltd. VMware, ESX, ESXi are trademarks or registered trademarks of VMware, Inc. in the United States and in other countries.

This product includes Windows Azure Linux Agent developed by the Microsoft Corporation (http://www.microsoft.com/). Copyright 2016 Microsoft Corporation.

This product includes software developed by the University of California, Berkeley (and its contributors), EMC, and Comtech AHA Corporation. This product is derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

The SteelHead Mobile Controller (virtual edition) includes VMware Tools. Portions Copyright © 1998-2016 VMware, Inc. All Rights Reserved.

NetApp Manageability Software Development Kit (NM SDK), including any third-party software available for review with such SDK which can be found at http://communities.netapp.com/docs/DOC-1152, and are included in a NOTICES file included within the downloaded files.

For a list of open source software (including libraries) used in the development of this software along with associated copyright and license agreements, see the Riverbed Support site at https//support.riverbed.com.

This documentation is furnished “AS IS” and is subject to change without notice and should not be construed as a commitment by Riverbed. This documentation may not be copied, modified or distributed without the express authorization of Riverbed and may be used only in connection with Riverbed products and services. Use, duplication, reproduction, release, modification, disclosure or transfer of this documentation is restricted in accordance with the Federal Acquisition Regulations as applied to civilian agencies and the Defense Federal Acquisition Regulation Supplement as applied to military agencies. This documentation qualifies as “commercial computer software documentation” and any use by the government shall be governed solely by these terms. All other use is prohibited. Riverbed assumes no responsibility or liability for any errors or inaccuracies that may appear in this documentation.



Riverbed Technology
680 Folsom Street
San Francisco, CA 94107
www.riverbed.com

Part Number
712-00127-10

Contents

Welcome 9

About this guide 9
Audience 9
Types of SteelHeads 10
Document conventions 11
Documentation and release notes 11
Contacting Riverbed 11
What's new 12

1 - CIFS Optimization 13

Overview of CIFS protocol 14
CIFS file locking and oplocks 15
Safe reading and writing 16
Security signatures 17
Overview of SMB versions 2, 2.1, 3, 3.02, and 3.1.1 18
More information 19
RiOS CIFS optimization techniques 19

2 - Microsoft Exchange Email Optimization 21

MAPI client and server communication 22
Autodiscovery and MAPI connections 23
RiOS MAPI optimization 24
Encrypted MAPI optimization 25
MAPI admission control for Microsoft Outlook 28
MAPI optimization with SteelHeads in a serial cluster or a parallel deployment 29
MAPI optimization with Exchange clusters 29
Outlook Anywhere optimization 30
Microsoft Exchange 2013 optimization 36
MAPI over HTTP 38
MAPI over HTTP requirements 39
MAPI over HTTP down negotiation 40
MAPI destination port handling 40
MAPI multiple context 42

3 - Signed SMB and Encrypted MAPI Optimization 45
Overview of SteelHeads optimizing with secure Windows traffic 45
Windows security concepts 46
Domain relationships 47
Choosing an authentication mode for the server-side SteelHead 49
Overview of configuring SMB signing and encrypted MAPI 50
SMB3 optimization with Windows 8, Windows 2012 Server, and later operating systems 51
SMB 3.1.1 optimization 52
Joining a SteelHead to a domain 53
One-way trust configuration 55
Enabling Kerberos in a restricted trust environment 56
Kerberos 56
Overview of Kerberos 57
Optimization in a native Kerberos environment 59
Domain user with replication privileges 60
Configuring traffic optimization for HTTP (SharePoint), encrypted MAPI, and signed SMB, SMB2, and SMB3 61
Configuring the server-side SteelHead for Active Directory integrated (Windows 2003/2008) 61
Best practices for the SteelHead in a secure Windows deployment 62
Domain authentication scaling 63
When to use domain authentication scaling 64
General improvements in RiOS 8.6 64
Domain controller load balancing 64
Domain health check and domain authentication automatic configuration 65
Domain health check 66
Domain authentication automatic configuration 70
Single domain example configuration 72

4 - HTTP Optimization 75
HTTP and browser behavior 76
Multiple TCP connections and pipelining 77
HTTP authentication 78
Connection jumping 80
HTTP proxy servers 81
Configuring HTTP SSL proxy interception 83
RiOS HTTP optimization techniques 85
Primary content optimization methods 86
HTTP vary headers 87
Connection pooling 89
HTTP authentication optimization 90
HTTP automatic configuration 91
HTTP Settings for common applications 94

HTTP optimization for SharePoint 95
HTTP optimization module and internet-bound traffic 96
HTTP and IPv6 97
Overview of the web proxy feature 97
Tuning Microsoft IIS server 98
Determining the current authentication scheme on IIS 98
Per-connection or per-request NTLM authentication 99
Per-connection or per-request Kerberos authentication 99
Changing the authentication scheme 100
Changing the per-connection/per-request NTLM authentication mode 100
Changing the per-connection/per-request Kerberos authentication mode 100
HTTP authentication settings 101
HTTP optimization module and proxy servers 101
Determining the effectiveness of the HTTP optimization module 102
Info-level logging 103
Use case 103

5 - Citrix ICA Optimization 107
Overview of Citrix ICA 107
Citrix version support 108
Citrix ICA traffic optimization with SteelHeads 108
Citrix SecureICA encryption 110
Citrix drive-mapping optimizations 110
Citrix multi-stream ICA traffic optimization with SteelHeads 112
Citrix virtual channels and traffic priorities 112
Single-stream and multi-stream ICA 112
QoS classification for Citrix traffic 117
Automatic negotiation of multi-stream ICA traffic for QoS enforcement 121
Reduction for Citrix small packet real-time traffic 123
Citrix ICA optimization over SSL 123

6 - Secure SMTP Optimization 127
Overview of SMTP and TLS 127
Configuring Microsoft Exchange servers for secure SMTP 128
Configuring the SteelHead for START-TLS support 130

7 - FTP Optimization 135
Overview of FTP 135
Active mode 135
Passive mode 136

Configuring in-path rules 137
Optimizing FTP 137
Passing through FTP 137
QoS classification for the FTP data channel 138
Active FTP classification 138
Passive FTP classification 138
FTP optimization considerations 139
SteelCentral Controller for SteelHead Mobile FTP considerations 140

8 - Other Protocol Optimization 141

Oracle Forms optimization 141
Determining the deployment mode 141
NFS optimization 142
Implementing NFS optimization 142
Configuring IP aliasing 143
Lotus Notes optimization 144
Optimizing encrypted Lotus Notes 144

9 - Video Optimization 153

Overview of Video optimization 153
HTTP stream splitting 154
Video on-demand with HTTP prepopulation 157
On-demand video caching 157

10 - CIFS and HTTP Prepopulation 159

CIFS prepopulation 159
Design considerations 163
HTTP prepopulation 165
Microsoft Silverlight, Apple HLS, and Adobe Flash 166

11 - SSL Deployments 169

The Riverbed SSL solution 169
Overview of SSL 171
How SteelHeads terminate an optimized SSL connection 172
Configuring SSL optimization on SteelHeads 174
SSL optimization required components 174
Setting up a simple SSL optimization deployment 177
Generating the proxy certificate and private key pair 181
SteelHead secure peering scenarios 183
Deploying secure peering for all optimized traffic 186
Advanced SSL features 188

Client certificate support 188
Proxy server support 190
Mid-session SSL support 190
Server name indication 191
SSL optimization with SteelHead Mobile 192
Troubleshooting and verification 192
Interacting with SSL-Enabled web servers 193
Obtaining the server certificate and private key 193
Generating self-signed certificates 195
Deploying a SteelHead with a SafeNet Network HSM 196
Requirements 197
Limitations 197

12 - Configuring SCEP and Managing CRLs 207

Using SCEP to configure on-demand and automatic reenrollment 207
Configuring on-demand enrollment 209
Configuring automatic reenrollment 209
Viewing SCEP settings and alarms 210
Managing Certificate Revocation Lists 211
Managing CRLs 213
Viewing CRL alarm status 214

Welcome

About this guide

This guide describes why and how to configure Riverbed appliances with common protocols and includes information relevant to the following products and product features:

- Riverbed Optimization System (RiOS)
- Riverbed SteelHead (SteelHead)
- Riverbed SteelHead CX (SteelHead CX)
- Riverbed SteelHead EX (SteelHead EX)
- Riverbed SteelHead (virtual edition) (SteelHead-v)
- Riverbed SteelHead (in the cloud) (SteelHead-c)
- Riverbed SteelHead SaaS (SteelHead SaaS)
- Riverbed SteelCentral Controller for SteelHead (Controller or SCC)
- Riverbed SteelCentral Controller for SteelHead (virtual edition) (Controller-v)
- Riverbed SteelCentral Controller for SteelHead Mobile software (SteelCentral Controller for SteelHead Mobile)
- Riverbed SteelCentral Controller for SteelHead Mobile appliance (Mobile Controller)
- Riverbed Mobile Controller (virtual edition) (Mobile Controller-v)
- Riverbed SteelHead Mobile (SteelHead Mobile)
- Riverbed SteelHead Interceptor (Interceptor)
- Riverbed Virtual Services Platform (VSP)
- Riverbed Services Platform (RSP)

Audience

This guide is written for storage and network administrators familiar with administering and managing WANs. You must be familiar with TCP, CIFS, Citrix, HTTP, Lotus Notes, MAPI, NFS, FTP, and SSL.

You must also be familiar with:

- the Management Console. For details, see the *SteelHead User Guide*.
- connecting to the RiOS CLI. For details, see the *Riverbed Command-Line Interface Reference Manual*.
- the installation and configuration process for the SteelHead. For details, see the *SteelHead Installation and Configuration Guide* and the *SteelHead (Virtual Edition) Installation Guide*.
- the Interceptor. For details, see the *SteelHead Interceptor User Guide*.
- the SCC. For details, see the *SteelCentral Controller for SteelHead User's Guide*.
- the Mobile Controller. For details, see the *SteelCentral Controller for SteelHead Mobile User Guide*.

Types of SteelHeads

The SteelHead product line includes several types of devices. Most of the information in the *SteelHead Deployment Guide - Protocols* applies to the following appliances:

- **SteelHead CX** - is a WAN optimization-only device. Desktop models have two in-path interfaces. For details, see the *SteelHead User Guide* for SteelHead CX (xx55).
- **SteelHead EX** - includes WAN optimization and VSP. The Riverbed SteelFusion product family, which provides branch storage services, is available with an additional license. For details, see the *SteelHead User Guide fo*r SteelHead EX (xx60).
- **SteelHead-v** - is a virtualized version of the SteelHead that runs on VMware ESX/ESXi and the Cisco Services-Ready Engine (SRE) platform. For details, see the *SteelHead (Virtual Edition) Installation Guide*.
- **SteelHead-c and SteelHead SaaS** - are the SteelHeads for public cloud computing environments. You deploy the SteelHead-c and SteelHead SaaS differently from the SteelHead and the SteelHead-v. For details, see the *SteelHead (in the Cloud) User Guide* and the *SteelHead SaaS User Guide (for Legacy Cloud Accelerator)*.
- **SteelHead Mobile** - optimizes network traffic from remote users who are accessing the enterprise network using any type of remote access such as dial-up, broadband, and wireless. For details, see the *SteelCentral Controller for SteelHead Mobile User Guide*.
- **Mobile Controller** - provides management functionality for SteelHead Mobiles. For details, see the *SteelCentral Controller for SteelHead Mobile User Guide*.
- **SCC** - provides management functionality for various Riverbed products, including SteelHeads, Mobile Controllers, and SteelHead Interceptors. For details, see the *SteelCentral Controller for SteelHead User's Guide*.

For more information about the SteelHead family, go to http://www.riverbed.com/us/products/steelhead_appliance/.

Document conventions

This guide uses the following standard set of typographical conventions.

Convention	Meaning
italics	Within text, new terms and emphasized words appear in *italic* typeface.
boldface	Within text, CLI commands, CLI parameters, and REST API properties appear in **bold** typeface.

Courier	Code examples appear in Courier font:
	`amnesiac > enable`
	`amnesiac # configure terminal`

< >	Values that you specify appear in angle brackets: **interface <ip-address>**
[ ]	Optional keywords or variables appear in brackets: **ntp peer <ip-address> [version <number>]**
{ }	Elements that are part of a required choice appear in braces: **{<interface-name> | ascii <string> | hex <string>}**
|	The pipe symbol separates alternative, mutually exclusive elements of a choice. The pipe symbol is used in conjunction with braces or brackets; the braces or brackets group the choices and identify them as required or optional: **{delete <filename> | upload <filename>}**

Documentation and release notes

The most current version of all Riverbed documentation can be found on the Riverbed Support site at https://support.riverbed.com.

See the Riverbed Knowledge Base for any known issues, how-to documents, system requirements, and common error messages. You can browse titles or search for keywords and strings. To access the Riverbed Knowledge Base, log in to the Riverbed Support site at https://support.riverbed.com.

Each software release includes release notes. The release notes list new features, known issues, and fixed problems. To obtain the most current version of the release notes, go to the Software and Documentation section of the Riverbed Support site at https://support.riverbed.com.

Examine the release notes before you begin the installation and configuration process.

Contacting Riverbed

This section describes how to contact departments within Riverbed.

- **Technical support** - Problems installing, using, or replacing Riverbed products? Contact Riverbed Support or your channel partner who provides support. To contact Riverbed Support, open a trouble ticket by calling 1-888-RVBD-TAC (1-888-782-3822) in the United States and Canada or +1 415-247-7381 outside the United States. You can also go to https://support.riverbed.com.
- **Professional services** - Need help with planning a migration or implementing a custom design solution? Contact Riverbed Professional Services. Email proserve@riverbed.com or go to http://www.riverbed.com/services/index.html.
- **Documentation** - Have suggestions about Riverbed's online documentation or printed materials? Send comments to techpubs@riverbed.com.

What's new

Since the *SteelHead Deployment Guide - Protocols* version in January 2018, the following changes were made:

- **HTTP/2 caching support**. SteelHead appliances can now support optimization of HTTP/2 cached traffic. See "HTTP Optimization" on page 75.
- **MPEG-DASH support**. SteelHead appliances can now support Dynamic Adaptive Streaming over HTTP (DASH), also known as MPEG-DASH. See "Video Optimization" on page 153.
- **SSL OCSP Stapling**. Server-side SteelHead appliances can now validate SSL certificates returned by the server using the online certificate status protocol (OCSP). See "SSL Deployments" on page 169.

For a list of new features added in RiOS 9.8, see the *SteelHead Installation and Configuration Guide*.

1

CIFS Optimization

This chapter describes the Common Internet File System (CIFS) protocol optimization module on the SteelHead. This chapter includes the following sections:

- "Overview of CIFS protocol" on page 14
- "RiOS CIFS optimization techniques" on page 19

CIFS, also referred to as the Server Message Block (SMB) protocol, has been in existence since the early 1990s. The protocol provides shared access to files and printers, along with other communication between hosts on a network, including an authenticated Inter-Process Communication (IPC) mechanism. It is one of the most common protocols used for network file access by Microsoft operating systems.

The protocol originated as SMB and was created by combined work from IBM and Microsoft. SMB was initially designed to run on top of NetBIOS/NETBEUI but has run directly on top of TCP since the Microsoft Windows 2000 version. SMB was primarily a LAN-based protocol, which accounts for some of the challenges you see when it is used across WAN links. The rename and relaunch from SMB to CIFS occurred in 1996, which was the same time that Sun Microsystems announced WebNFS.

Although CIFS is known as the generic name for the protocol, technical discussions and documents continue to use the term SMB. Since the initial release of SMB, Microsoft has continued to change and enhance the protocol, each time assigning a new version number to SMB. The most recent versions are SMB2, SMB 2.1, SMB3, SMB 3.02, and SMB 3.1.1.

For information about SMB versions, see "Overview of SMB versions 2, 2.1, 3, 3.02, and 3.1.1" on page 18.

Note: For the purpose of this chapter, the terms SMB and CIFS are used interchangeably.

RiOS has provided CIFS optimization since version 1.0, with ongoing enhancements in each subsequent version. CIFS optimization focuses on reducing the impact of WAN round-trip latency of common application and system behavior, including the Microsoft Office product suite, general file access, and remote printing. Ongoing work has been necessary to adjust to new incremental changes introduced to the protocol from Microsoft, typically with each new Microsoft operating system version. Additionally, new features have been added that enhance and broaden the applicability of the CIFS optimization.

RiOS includes optimization for *SMB signed* traffic. *SMB signed* refers to an optional feature in which the client and server, using the CIFS protocol, use cryptographic techniques to sign each protocol datagram exchanged in a CIFS session. This technique does not encrypt data passing through the network; it merely authenticates that the client and server receive datagrams that have not been modified by unauthorized middle devices. Through specific configuration and integration into the Windows security domain, RiOS can perform full latency and bandwidth optimization securely for SMB signed traffic, while still maintaining the end-to-end authentication that SMB signing was designed to achieve.

While SMB signing does not encrypt the data, SMB3 can encrypt traffic. If you use the correct RiOS release with the appropriate configuration settings, RiOS can continue to provide full-latency and bandwidth optimization while maintaining a secure client-server communication.

For information about configuring CIFS optimization, see the *SteelHead User Guide*.

Overview of CIFS protocol

This section describes the base CIFS protocol and how RiOS performs CIFS optimization. For example, for a client to access a file on a CIFS server, the following steps generally occur on the client:

1. Parse the full filename to determine the server name.
2. Establish a TCP connection to the server.
3. Negotiate what type of SMB dialect and option is used with the server.
4. Transfer credentials and authenticate access to the server.
5. Open the file for operations by sending the filename and desired access to the server.
6. Perform read, write, or other operations on the file, typically in a serial fashion.
7. Close the file for operation.

After a connection to the server is established, client systems tend to leave the connection open for some amount of time, even when there are no open files by the client. This has an impact on CIFS optimization, because RiOS can optimize CIFS connections only when it detects the beginning of the CIFS connection. This is due to the negotiation of SMB options and SMB dialect that allow RiOS to understand what client and server operating systems are in use, which revision of the CIFS protocol is in use, and other options relevant to optimizing the connection.

The following table shows CIFS protocol commands that traverse the network during a session set up and subsequent simple file read.

Client command	Server response
SMB_COM_NEGOTIATE	Must be the first message sent by a client to the server. Includes a list of SMB dialects supported by the client. Server response indicates which SMB dialect should be used.
SMB_COM_SESSION_SETUP_ANDX	Transmits the user authentication information. Successful server response has the user ID (UID) field set in the SMB header and is used for subsequent SMBs on behalf of this user.

Client command	Server response
SMB_COM_TREE_CONNECT_ANDX	Transmits the name of the disk share the client wants to access. Successful server response has a Tree Connect ID (TID) field set in the SMB header that is used for subsequent SMBs referring to this resource.
SMB_COM_OPEN_ANDX	Transmits the name of the file, that relative to the TID, the client wants to open. Successful server response includes a file ID (FID) that the client supplies for subsequent operations on this file.
SMB_COM_READ_ANDX	Client supplies TID, FID, file offset, and number of bytes to read. Successful server response includes the requested file data.
SMB_COM_CLOSE	Client closes the file represented by TID and FID. Server responds with success code.
SMB_COM_TREE_DISCONNECT	Client disconnects from resource represented by TID.

This sequence represents the example protocol flow for a specific use case. Notice that SMB_COM_SESSION_SETUP_ANDX contains the user's authentication information. This is also particularly relevant when there are signed SMB sessions and Windows domains as part of the client-server environment. For details, see "Windows security concepts" on page 46.

READ_ANDX and other commands can be used (in some cases) several hundreds of times before the *end of file* is reached. There are many additional round trips across a WAN link that contribute to the overall performance degradation compared to the same operation across a LAN.

CIFS file locking and oplocks

The CIFS protocol allows for different types of file locking and concurrent access to be used by client and servers. Separately, the CIFS protocol allows servers to notify clients (when requested) when they are the only client accessing a file, or when all clients for a file are performing only reads.

Requesting an *oplock* is when a client opens a potentially shared file and requests to be notified by the server when it is the only client accessing the file, or when it is accessing the file with other clients that are performing only reads (and no writes). When a client is granted an oplock on a file, it can perform some types of actions on the file completely locally, without notifying the server. Oplock breaks occur when servers notify a client if additional clients request access to the file and remote operations performed at the client must be synchronized with the server before granting that access.

The following types of oplocks are available:

- **Batch** - This oplock is generally used by command-line applications that repeatedly open many of the same files. The term *batch* is used for this type of access, which is most often experienced by batch (or script) command sets. Batch applocks allow clients to perform file opens completely, without notifying the server.
- **Exclusive** - This oplock is granted to a client if the file is not already opened by another client. This means that the client has sole access to the file, so it is safe to cache changes until the file closes or when requested by the server.

- **Level 2** - This oplock is never requested by a client directly. Instead, a client that holds an exclusive oplock might be downgraded to a level 2 oplock at any time by the server, indicating that some additional client has accessed the file but has not yet performed any write operations.

Note: Do not confuse oplocks with file locks. File locks control access to files, and oplocks notify clients about the state of concurrency for a file.

Clients can request a specific type of shared-access control for a file when they open it. Examples of the types of shared access are do not share, share for read only, share for write only, or share for read/write. Clients can also apply more granular control by requesting locks on specific portions of a file, known as *byte-range locks*. The CIFS server tracks all of the access requests that it detects, and it grants or denies operations based on the requests from live clients. The shared access control, or *shared mode* access, is the most prevalent in most user applications. Typically, only database and similar applications can make use of byte range locks.

RiOS registers these concurrency controls: file locks, share access controls, and oplocks. The RiOS CIFS module tracks the state of every file opened between the client and server, and it uses this state to determine what types of optimized connections can safely be applied to file requests on the clients.

For example, when a client is granted an exclusive oplock on a file, RiOS can safely perform read-ahead and write-behind of file data because it is the only client accessing the file. Some of the CIFS features in RiOS enable more in-depth recognition of concurrency control, like the overlapping open optimization and the applock optimization feature.

Safe reading and writing

The concurrency controls—file locks, shared-access controls, and oplocks—ensure that clients can coordinate their reads and writes in an orderly fashion. In particular, when a client wants to read from a file, it knows that reading from a local cache is safe (for example, through oplocks), or that its reads and writes are coordinated with other clients (for example, through byte range locks).

When a client wants to write data to a file, there are two modes of writing that are supported in CIFS environments: *write-through* and *write-back*. The application using CIFS can decide which of these two modes to use every time it needs to write.

For write-through mode, the process falls into a pattern in which the client sends data to the server for writing, waits for an acknowledgment from the server, sends some more data, waits, and repeats this process, until all the data is written and acknowledged. At this point the client sends a CIFS command to close the file on the server. The acknowledgment from the server that the file is closed signals that all data is safely written to disk. The server releases any lock on the file and makes it available for read-write access to other clients. Although this might be the safest way when writing data, when sending data across the WAN, it is not the most efficient method because the round-trip time across the WAN is typically several magnitudes higher than across the LAN.

Write-back mode is a form of optimization in which the client does not need to wait until the write is acknowledged before proceeding with the next operation. Synchronization is forced by the client flushing its cache or closing the file. Microsoft Word and Excel are good examples of applications that use CIFS in a write-back mode. If for any reason there is a problem when the file is closed, the user is informed and the data is recovered using the automatic save mechanism.

In write-back mode, there are several places where data can linger on its way to the disk: the client machine can gather writes, the server file system can delay pushing to disk, and even the server's physical disk subsystem can cache writes in RAID cards or on memory in the disk itself. RiOS provides another place for this temporary buffering between client and long-term storage to occur during write-back mode.

RiOS honors the client-specified write-back or write-through mode for each write operation. In general, most clients use write-back mode, because the performance of write-through mode—even on local disks, without the added performance impact of WAN based usage—is unacceptable to end users. RiOS can deliver fast write performance over the WAN in a way that is analogous to how file servers and network attached storage (NAS) appliances deliver fast write performance.

Security signatures

The CIFS protocol includes a feature to cryptographically sign every CIFS protocol datagram between the client and server. The file data that traverses the network is still sent in the clear, even when this option is used, so this CIFS feature does not protect against network snooping. However, it does protect against man-in-the-middle attacks.

The security signature feature is typically controlled through Windows group policy settings, with a separate client-side and server-side three-way setting that controls whether signing is used or not. This policy setting is then used during the initial CIFS session by the client and server to determine whether or not to sign their CIFS sessions. The following table shows the result when a client attempts to connect to a server—either the connection fails due to incompatible settings, or the session can sign the CIFS datagrams.

	Server - disabled	Server - enabled	Server - required
Client - disabled	Not signed	Not signed	No connections
Client - enabled	Not signed	Signed	Signed
Client - required	No connections	Signed	Signed

Several options are available for RiOS to optimize connections that are signed:

- **Do nothing** - RiOS can apply bandwidth and TCP-level optimization but does not perform CIFS latency optimized connections. This is because the client-side SteelHead needs to generate CIFS datagrams to the client as if it were the server, and the server-side SteelHead needs to generate CIFS datagrams to the server as if it were the client.
- **Enable the Optimize Connections with Security Signatures (that do not require signing) feature** - This is only useful when neither the client nor the server use the required setting. When you enable this feature, RiOS alters the negotiation of signing between the client and server so that they each detect the disabled setting.
- **Join the server-side SteelHead appliance to the Windows domain, and possibly configure user delegation if necessary** - Although this requires the most configuration, it maintains the end-to-end security guarantees that security signatures allow. Sessions continue to be signed between the clients and client-side SteelHead, between the SteelHeads, and between the server-side SteelHead and server. Maintaining this end-to-end status can be a requirement for some security mandates. For details, see "Windows security concepts" on page 46.

Overview of SMB versions 2, 2.1, 3, 3.02, and 3.1.1

Since the initial release of SMB, Microsoft has continued to change and enhance the SMB protocol. These changes have usually coincided with the release of a new version of the Windows client and server operating system:

- SMB2 was introduced with Windows Vista and Windows Server 2008.
- SMB 2.1 was introduced in Windows 7 and Windows Server 2008 R2.
- SMB3 was introduced with Windows 8 and Windows Server 2012.
- SMB 3.02 was introduced with Windows 8.1 and Windows Server 2012 R2.
- SMB 3.1.1 was introduced with Windows 10 and Windows Server 2016.

Using the appropriate version of RiOS, you can configure your environment so that full-latency and bandwidth optimizations are possible regardless of the SMB version or dialect.

Not all SMB features are relevant when considering optimizing SMB traffic with RiOS; however, consider the following features:

- SMB2
  - Request compounding, which allows multiple SMB 2 requests to be sent as a single network request.
  - Larger reads and writes.
  - Caching of folder and file properties.
  - Improved message signing (HMAC SHA-256 replaces MD5 as hashing algorithm).
- SMB 2.1
  - Client oplock leasing model.
  - Large MTU support.
  - Support for previous versions of SMB (allowing Windows 7 clients and Server 2008 R2 servers to negotiate for backwards compatibility).
- SMB3
  - Secure negotiation (allows the client and server to detect a man-in-the-middle that modifies the negotiation process).
  - Directory leases (allows the client to safely cache directory contents).
  - Encrypted traffic (encryption of the SMB3 traffic on a per-server or per-share basis).
- SMB 3.02
  - Some additional capabilities (for example, Instancing and read and write flags). These capabilities are not relevant for WAN optimization on their own, but because they are in the client-server session setup dialog, the SteelHead needs to understand the dialect to respond correctly.
- SMB 3.1.1
  - Preauthentication integrity (improved protection from man-in-the-middle attacks tampering with SMB connection establishment and authentication).

– Per connection crypto algorithm negotiation (options for AES-128-CCM and AES-128-GCM).

For more information about SMB3, see "SMB3 optimization with Windows 8, Windows 2012 Server, and later operating systems" on page 51.

More information

Although the low-level detail of the CIFS protocol is beyond the scope of this document, there are many public reference materials available. For a starting point, go to http://media.server276.com/codefx/CIFS_Explained.pdf.

RiOS CIFS optimization techniques

The CIFS optimization module comprises many separate techniques. Many of these techniques are enabled by default, so when configuring a new SteelHead (or creating a new SteelCentral Controller for SteelHead Mobile package), minimal, if any, CIFS-specific configuration is required.

Note: Typically, new features in RiOS, even ones that are widely applicable, are not enabled by default until they have been present for several releases. Some optimization features that are not enabled by default can provide benefit in your environment, and you can consider enabling them on an individual basis.

Modern CIFS servers generally listen on TCP port 445; for compatibility, they also listen to port 139. The CIFS optimization module within RiOS is designed to intercept traffic on these two ports for acceleration of file-sharing and remote Windows printing traffic.

After a CIFS connection is intercepted for optimization, the RiOS CIFS optimization module monitors the negotiation and session setup commands to detect which client and servers are in use and what types of CIFS options are understood by each. As clients initiate requests to the server, RiOS applies its optimization techniques to reduce the time that the client awaits a response from the server.

For specific information about CIFS optimization features, see the *SteelHead User Guide*.

2

Microsoft Exchange Email Optimization

This chapter describes the optimization of the Messaging Application Programming Interface (MAPI) used between Microsoft Outlook clients and Exchange servers. The combination of Outlook and Exchange server is one of the most common client and server combinations for email in a corporate business environment. This chapter includes the following sections:

- "MAPI client and server communication" on page 22
- "RiOS MAPI optimization" on page 24
- "MAPI over HTTP" on page 38
- "MAPI destination port handling" on page 40
- "MAPI multiple context" on page 42

Note: The MAPI communication mechanism is also known by a more recent name: Microsoft as Outlook-Exchange Transport Protocol.

MAPI comes in two forms: extended and simple. This chapter discusses the extended form, which is used by Outlook. The simple form is included as part of Microsoft Outlook Express and is not discussed here.

In the last ten years, MAPI has evolved through Exchange 5.5, Exchange 2000, Exchange 2003, Exchange 2007, and Exchange 2010. Outlook versions have followed in close coordination. RiOS can perform optimization for each of these Exchange versions.

For information about Exchange 2013, see "Microsoft Exchange 2013 optimization" on page 36.

Because of the requirements of Exchange 2003, the Exchange server communicates using an encrypted conversation with compatible versions of the Outlook client. By default, the encryption mode is enabled in Exchange 2007 and Exchange 2010. RiOS securely optimizes this encrypted MAPI traffic by using the RiOS MAPI optimization along with the Windows-domain-related RiOS features.

Cached Exchange Mode was introduced with Outlook 2003. Cached Exchange Mode attempts to hide performance issues over the WAN by preemptively downloading new email and their attachments from the Exchange server to a cache of storage on the local hard disk of the Outlook client workstation. Although this gives an appearance of fast performance, it has no effect on emails going to the sent items or trash folder of the user. Using Cached Exchange Mode also means that any mail, including unwanted or junk mail (and attachments), is downloaded to the Outlook client. RiOS provides optimization for Outlook clients using Cached Exchange Mode, both by reducing the increased WAN bandwidth impact of Cached Exchange Mode and decreasing the wait time for arriving and transmitted emails.

For more information about configuring MAPI optimization, see the *SteelHead User Guide*.

MAPI client and server communication

The Exchange server includes an Endpoint Mapper (EPM) that listens on TCP port 135. The Outlook client connects to this port and is assigned random TCP server ports to communicate with the Exchange server using the MAPI protocol. These MAPI connections are used to send and receive emails, calendaring, address lookup, and more. You can configure the Exchange server to use a fixed TCP port for the MAPI connections.

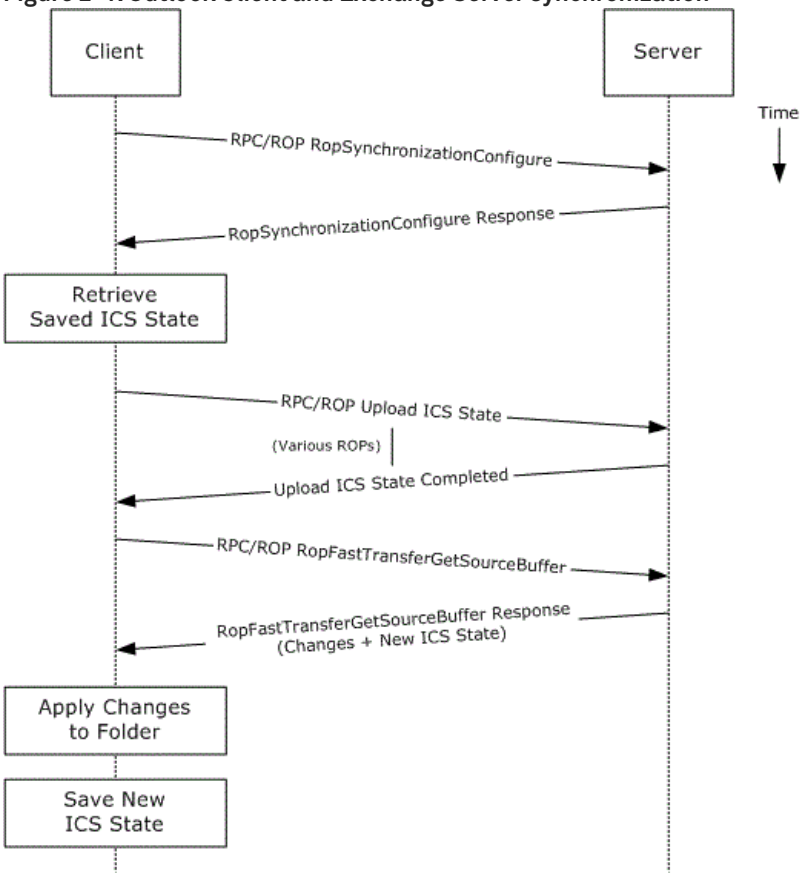
The conversation between the Outlook client and the Exchange server is quite complex. The following is a basic overview for the Exchange server communication:

1. The Outlook client contacts the directory service and DNS to establish the universal resource identifier (URI) for the Exchange server.
2. The Outlook client uses the URI to connect to the EPM server, and then it logs in to the Exchange server.
3. The Outlook client performs remote operations (ROPs) to open (that is, create and write) a new email message, saves the changes, and creates and adds an attachment, if needed.
4. Using Name Service Provider Interface (NSPI), the Outlook client resolves the name of a recipient.
5. The Outlook client submits the message to the Exchange server for sending.

Note: For information about Exchange server communication, go to http://msdn.microsoft.com/en-us/library/cc307725%28EXCHG.80%29.aspx.

Other conversations between the Outlook client and the Exchange server include opening folders for reading or searching, deleting items, and synchronizing folders. Synchronizing folders is required to download and read emails from the Exchange server to the Outlook inbox. Figure 2-1 shows the synchronization process. The incremental change synchronization (ICS) state is how the client knows what new items to download from the Exchange server.

Figure 2-1. Outlook Client and Exchange Server synchronization



Source: http:www.microsoft.com (Sep. 2, 2010)

SteelHead MAPI optimization is based on the streamlining of ROPs.

Autodiscovery and MAPI connections

RiOS intercepts and optimizes the EPM conversation between the Outlook client and Exchange server on TCP port 135. For Outlook client connections that are configured for correct addressing, the client-side SteelHead alters the port that the EPM service sends to the Outlook client for MAPI connections (default is port 7830). For port or full transparency addressing modes, no changes are made to the reported MAPI port, and the Outlook client connects to whichever port EPM assigns to MAPI. RiOS intercepts and optimizes subsequent connections made to the MAPI port and performs MAPI-specific optimization on them.

If the Outlook client connections to the Exchange server are intercepted and optimized, MAPI-specific optimized connections are not used when the client-side SteelHead has an in-path rule to pass-through traffic on TCP port 135 (or the server-side SteelHead has a peering rule to pass-through traffic on TCP port 135).

After RiOS learns the MAPI port for an Exchange server, RiOS no longer goes through the autodiscovery process for connections to the Exchange server IP address and learned MAPI port. Although this gives some performance benefits to new MAPI connections (especially in large latency WANs), it also means that configuring pass-through or other types of in-path rules might not have the immediately desired impact on traffic.

Note: Always use the **in-path probe-mapi-data** command to perform autodiscovery for MAPI connections and to allow in-path rules to have immediate impact, even on Outlook clients that are currently optimized.

When using fixed-target rules to optimize MAPI traffic, configure the client-side SteelHead to optimize traffic where the destination IP address is the Exchange server and the destination port is 7830 (or whatever port is configured on the client-side SteelHead for MAPI optimization). If you want optimization for NSPI connections, a fixed-target rule in which the destination IP address is the Exchange server and where the destination port is 7840 (or whatever port is configured on the client-side SteelHead for NSPI optimization).

RiOS MAPI optimization

The MAPI optimization module comprises several optimization techniques:

- Read Ahead on attachments
- Read Ahead on emails
- Write Behind on attachments
- Write Behind on emails
- Folder Synchronization
- Prepopulation

Note: : Each separate MAPI connection is used by one user only, but it is common for one Outlook to have several MAPI connections active simultaneously—and it usually does. For example, in a branch office where a SteelHead optimizes MAPI traffic for several users, there are several optimized MAPI connections per user, but no single MAPI connection is shared between users. However, when a user closes Outlook and triggers the start of a MAPI prepopulation session, there is only one optimized MAPI prepopulation connection per user.

The following are MAPI optimization methods on the SteelHead:

- **MAPI optimization** - This method is the fundamental component of the MAPI optimization module and includes optimization for Read, Write (Receive, Send), and Synchronize operations. This setting uses port 7830 and is enabled by default. If the Exchange server is not configured to use random ports (allocated by EPM) and needs to use a fixed port configured by the administrator, you must change port 7830 to the fixed port number used by the Exchange server.
- **Encrypted optimization** - Enables the client-side SteelHead functionality required for encrypted MAPI optimization. For details, see “Encrypted MAPI optimization” on page 25.

- **MAPI prepopulation** - Allows a SteelHead to warm its RiOS data store with the data patterns of new mail and attachments that arrive after a user shuts down Outlook. This is especially useful in global organizations where communication to a user continues long after local business hours. MAPI prepopulation considerably reduces the bandwidth consumed by a branch office when users start their day, and it can greatly accelerate the reception of mail when a user first connects to an Exchange server.

  MAPI prepopulation does not use any additional Client Access Licenses (CALs) from the Exchange server. To start a MAPI prepopulation session, the SteelHead automatically holds open an existing authenticated MAPI connection after Outlook has shut down on the client. At the same time, the client-side SteelHead sends a message to the server-side SteelHead to prepare a MAPI prepopulation session. The session begins if the following conditions are met:

  – The server-side SteelHead has MAPI prepopulation enabled

  – The existing total client-side SteelHead active connections for MAPI prepopulation has not reached its maximum configured limit

  – A MAPI prepopulation session has not already been started for the client

  No user credentials are used or saved by the SteelHead when performing prepopulation.

  The remote SteelHead uses these preauthenticated connections to pull the data for mail from the Exchange server over the WAN link, automatically prepopulating the client-side RiOS data store.

  If a user starts a new Outlook session, the MAPI prepopulation session terminates. If for some reason the MAPI prepopulation session does not terminate (for example, the user starts a new session in a location that is different than the SteelHead that has the MAPI prepopulation session active), the MAPI prepopulation session will eventually time-out per the configuration setting.

  In RiOS 6.5 and later, the control of MAPI prepopulation is on the client-side SteelHead. This allows for a higher rate of prepopulated sessions, and it enables MAPI prepopulation to take advantage of the read-ahead mechanisms in the MAPI optimization feature.

  MAPI prepopulation v2 is supported in RiOS 6.0.4 and later, 6.1.2 and later, and 6.5 and later. The client-side and server-side SteelHeads can be at any of these code train levels and provide prepopulation v2 capabilities. For example, a client-side SteelHead running RiOS 6.0.4 connecting to a server-side SteelHead running RiOS 6.5 and later provides prepopulation 2 capabilities. In contrast, a 6.0.1a client-side SteelHead connecting to a RiOS 6.5 and later server-side SteelHead supports prepopulation v1, but it does not provide prepopulation v2.

  MAPI prepopulation v2 tracks the User ID as well as the client IP address. This is helpful in scenarios where the user returns to the branch office and is assigned a different IP address by DHCP. In MAPI prepopulation v1 (where just the IP address was tracked), the prepopulation session could have remained until the connection timed out (default 96 hours).

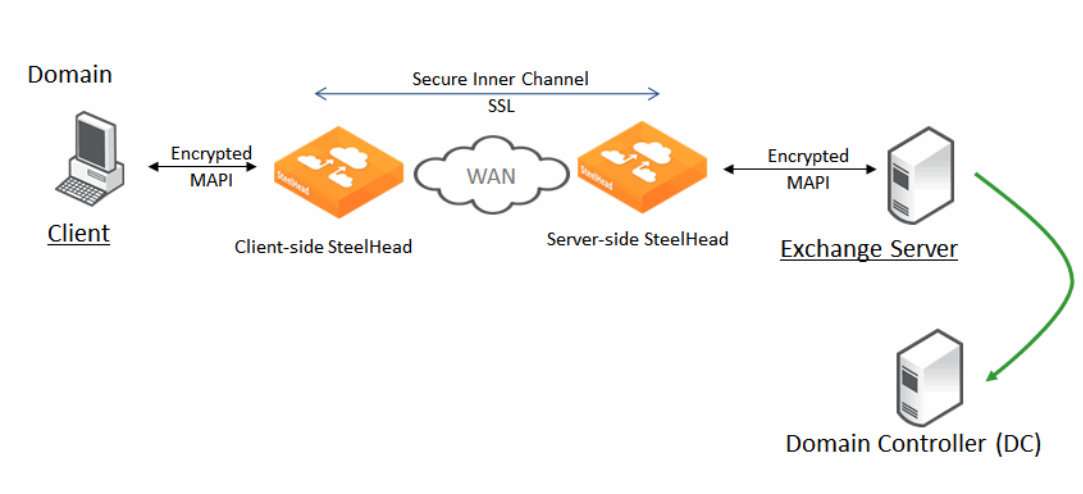
Note: MAPI prepopulation does not occur when using MAPI Outlook Anywhere connection (RPC over HTTP). When a MAPI Outlook Anywhere session disconnects the SteelHead terminates all proxy flows.

Encrypted MAPI optimization

When optimizing encrypted MAPI traffic, normal encryption methods are maintained between the Outlook client and client-side SteelHead and between the Exchange server and server-side SteelHead.

To ensure the optimized MAPI connection between the two SteelHeads is also encrypted, configure RiOS secure inner channel. For detail, see the *SteelHead Deployment Guide*.

Figure 2-2. Encrypted connections between client and server



To enable the SteelHead to optimize encrypted MAPI traffic between the Outlook client and the Exchange server

1. On the server-side SteelHead, choose Optimization > Active Directory: Domain Join.
2. Join the server-side SteelHead to the same Windows domain that the Exchange server belongs to and operates as a member server.

   The server-side SteelHead must be able to ping the domain controller by name.

   You can use an adjacent domain (through Cross-Domain support) if the SteelHead is running RiOS 6.1 or later.
3. Verify that Outlook is encrypting traffic.
4. Enable the Encrypted Optimization option on client-side and server-side SteelHeads involved in optimizing MAPI encrypted traffic. Alternatively, use the **protocol mapi encrypted enable** command.
5. Use transparent mode for all client types. Delegate mode is a legacy feature and should migrated to transparent mode, unless you have a specific need for Kerberos Constrained Delegation.
6. Restart the service on all SteelHeads that have the Encrypted Optimization option enabled.

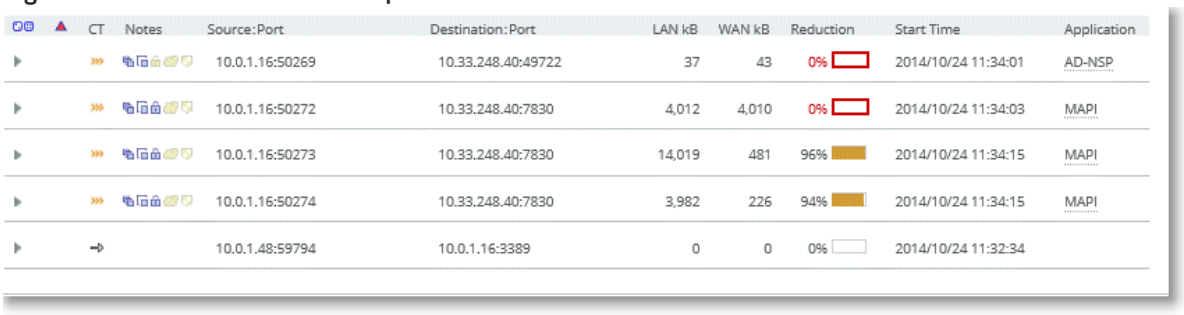
Note: The server-side and client-side SteelHeads must be running RiOS 5.5.x or later.

The Windows Security section also provides information about the relevant, earliest version of RiOS to ensure support for different Windows client operating systems and Windows domain structures.

For more information about how to ensure the SteelHead is joined to the domain, as well as configuring transparent or delegation modes, see “Domain relationships” on page 47.

You can confirm that the SteelHead is successfully performing encrypted MAPI optimization by going to the SteelHead Management Console Current Connections report and looking for the highlighted lock icon listed in the Applications column of the report page. In versions prior to RiOS 8.6, this is listed as ENCRYPT-MAPI.

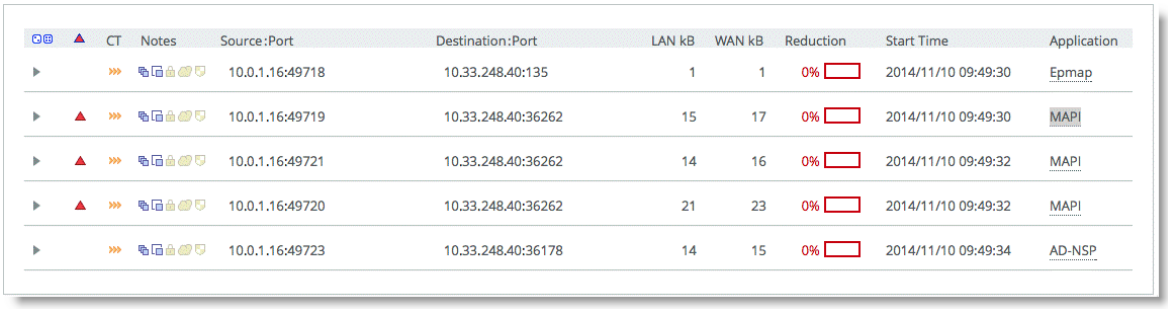
Figure 2-3. Current Connections report



CT	Notes	Source:Port	Destination:Port	LAN kB	WAN kB	Reduction	Start Time	Application
»		10.0.1.16:50269	10.33.248.40:49722	37	43	0%	2014/10/24 11:34:01	AD-NSP
»		10.0.1.16:50272	10.33.248.40:7830	4,012	4,010	0%	2014/10/24 11:34:03	MAPI
»		10.0.1.16:50273	10.33.248.40:7830	14,019	481	96%	2014/10/24 11:34:15	MAPI
»		10.0.1.16:50274	10.33.248.40:7830	3,982	226	94%	2014/10/24 11:34:15	MAPI
→		10.0.1.48:59794	10.0.1.16:3389	0	0	0%	2014/10/24 11:32:34	

If for some reason this is not successful, you see a red triangle in the Notes column.

Figure 2-4. Unsuccessful Current Connections report



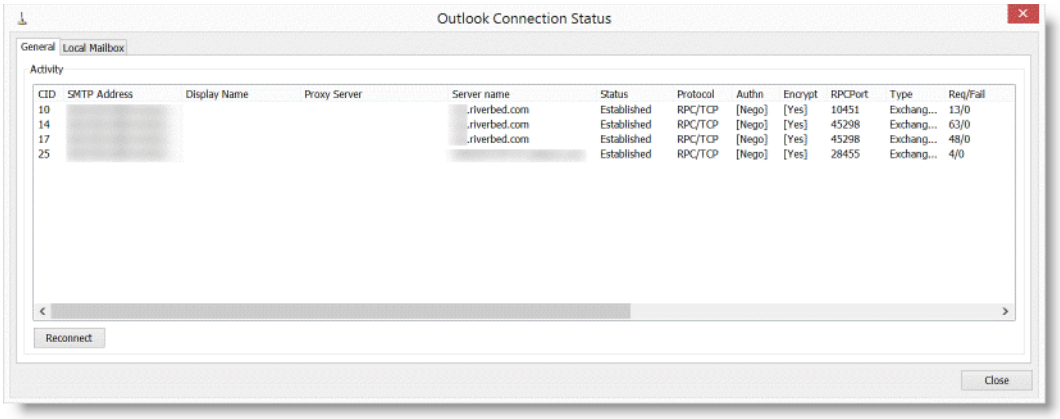
CT	Notes	Source:Port	Destination:Port	LAN kB	WAN kB	Reduction	Start Time	Application
»		10.0.1.16:49718	10.33.248.40:135	1	1	0%	2014/11/10 09:49:30	Epmap
»	▲	10.0.1.16:49719	10.33.248.40:36262	15	17	0%	2014/11/10 09:49:30	MAPI
»	▲	10.0.1.16:49721	10.33.248.40:36262	14	16	0%	2014/11/10 09:49:32	MAPI
»	▲	10.0.1.16:49720	10.33.248.40:36262	21	23	0%	2014/11/10 09:49:32	MAPI
»		10.0.1.16:49723	10.33.248.40:36178	14	15	0%	2014/11/10 09:49:34	AD-NSP

You can see the WAN KB values are higher than the LAN KB values. For information about how to troubleshoot this problem, see the *SteelHead User Guide*.

Enabling support for encrypted MAPI when there is a mixture of both encrypted Outlook clients and native Outlook clients allows the native clients to benefit from optimization.

In RiOS 8.6 and later, applications are defined by the DPI engine. All Exchange connections are displayed as MAPI, specifically DCE-RPC > Exchange-Protocols > MAPI. Encrypted MAPI is noted by the highlighted lock icon. You can also see this on the client machine by holding the Control key and right clicking on Exchange icon in task tray.

Figure 2-5. Outlook connection status



MAPI admission control for Microsoft Outlook

MAPI admission control for Microsoft Outlook are control measures to prevent the SteelHead from entering admission control when there are many TCP connections. A*dmission control* is a state a SteelHead enters when it has stopped optimizing new connections because it has exhausted a certain type of resource. During admission control, new connections pass through, while existing connections are optimized. Upon exiting admission control, the SteelHead once again intercepts and optimizes new connections.

For more information about admission control, see the *SteelHead Deployment Guide* and the *SteelHead User Guide*.

When you use an Exchange server and the clients are using Microsoft Outlook, Outlook initiates and maintains multiple TCP connections with the Exchange server. A *MAPI session* is the collection of multiple TCP connections from an Outlook client.

When Microsoft Outlook connections are optimized, the SteelHead remaps contexts to prevent inadvertent mixing of optimized and nonoptimized traffic. MAPI differs from other protocols because the TCP connections of a session are not independent of other TCP connections from the same session.

Because of how Exchange servers interact with MAPI sessions, you must optimize all client MAPI connections by the same pair of client-side and server-side SteelHeads, or have them all set to pass through. If you optimize only some of the connections, the session fails. If the connections are not optimized by the same pair of SteelHeads, the session fails.

Prior to RiOS 7.0, MAPI admission control v1 used a threshold percentage used to calculate control. For example, if a SteelHead had a 1000 connection limit, the admission control cutoff value of 85% was 850 connections. The remaining 150 connections were made available only for existing MAPI sessions.

MAPI admission control v2 in RiOS 7.0 and later is enhanced to include:

- Server-side SteelHead awareness.

  MAPI admission control v1 was a client-side SteelHead implementation only. MAPI admission control v2 continues as a client-side SteelHead implementation, but it is also aware of the server-side SteelHead.
- MAPI admission control v2 preemptively closes MAPI sessions to reduce the connection count in an attempt to bring the SteelHead out of admission control. MAPI sessions are closed in the following order:
  - MAPI prepopulation connections
  - MAPI sessions with the largest number of connections
  - MAPI sessions with the most idle connections
  - The oldest MAPI session
  - MAPI sessions exceeding the memory threshold

While the SteelHead is in the admission control state, a special handling of MAPI sessions is activated. Because new connections cannot be optimized for an existing client session, the SteelHead closes all connections from this existing client. Outlook reestablishes connectivity to the Exchange server, and because the SteelHead is in an admission control state, the new connections of the client are detected as pass-through connections.

MAPI optimization with SteelHeads in a serial cluster or a parallel deployment

MAPI admission control is helpful when there are two client-side SteelHeads in a serial cluster. MAPI optimization expects all connections from an Outlook client to be optimized by the same SteelHead and not split across two SteelHeads. However, the thresholds in RiOS 7.0 and later significantly reduce the likelihood of this situation occurring.

In a typical parallel deployment in which you configure the SteelHeads as connection forwarding neighbors, the SteelHeads forward new MAPI connections to the SteelHead that detects the initial MAPI connection for the user session. This effectively avoids failure situations in which the MAPI connections for a single session are not optimized on the same SteelHead.

MAPI optimization with Exchange clusters

In larger Microsoft Exchange deployments it is possible that the Exchange server comprises several nodes working in a cluster. With Exchange 2010, the cluster can have some form of load balancer to distribute connections among the various nodes in the cluster, including one or more client access servers (CAS) that act as a front end to the mailbox servers. You must configure the client-side SteelHead with one of the following settings:

- Enable port transparency for MAPI traffic.
- Enable full transparency for MAPI traffic.

- Disable MAPI port remapping with the **no protocol mapi port-remap enable** command.

Note: You must restart the service for this command to take effect.

CAS requires RiOS 6.1.1 or later. There is not a specific configuration required on the server-side SteelHead to support optimization for Exchange clusters.

We recommend RiOS 6.1.4 or 6.5.1 for Exchange clusters that use multiple CAS and Microsoft Network Load Balancing (NLB) for load balancing.

Outlook Anywhere optimization

Outlook Anywhere (OA) is a feature in Exchange 2003 and later. It allows Outlook clients to connect to Exchange over HTTP or HTTPS. Remote users can connect to Exchange without using specialized VPN software. The Outlook Anywhere protocol uses a variant of the existing MAPI protocol transported over HTTP. In releases earlier than RiOS 6.5, the SteelHead was unable to provide latency-specific optimization for this traffic.

Outlook Anywhere optimized traffic uses the new RPC over HTTP optimization engine, as well as the existing MAPI, HTTP, and SSL optimization features.

Note: The SteelHead must be properly licensed to use SSL. For details, see "SSL optimization required components" on page 174.

Outlook Anywhere can leverage both traditional MAPI and encrypted MAPI. If you use encrypted MAPI, the server-side SteelHead must be a member of the domain.

The following table shows encryption types using HTTP and encrypted MAPI.

Tunnel type	Encryption
HTTP tunnel regular MAPI	Not encrypted
HTTPS tunnel regular MAPI	Encrypted once
HTTP tunnel encrypted MAPI	Encrypted once
HTTPS tunnel encrypted MAPI	Encrypted twice

The following list describes requirements for Outlook Anywhere:

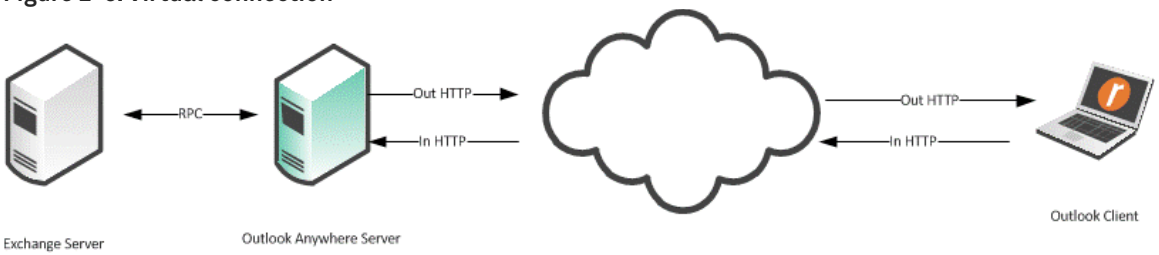
- RiOS 6.5 and later
- Microsoft Outlook and Exchange Server 2003 and later
- Encrypted MAPI requires the server-side SteelHead to join the windows domain of the Exchange server (for details, see "Joining a SteelHead to a domain" on page 53)
- HTTPS requires SSL licensing and an SSL certificate and key from the Outlook Anywhere server

Note: Outlook Anywhere optimized connections cannot start MAPI prepopulation.

The communication flow of Outlook Anywhere is among an Outlook client, the Outlook Anywhere server, and an Exchange server. The Outlook Anywhere server is an RPC Proxy server. The server takes RPC calls to and from the Exchange server, removes the RPC headers, and encapsulates the data in HTTP or HTTPS. Because MAPI is a full duplex protocol, two HTTP/HTTPS connections are required for each Outlook Anywhere connection. This is called a *virtual connection*.

Figure 2-6 shows the RPC connection between the Outlook Anywhere server and the Exchange server. The Outlook Anywhere server strips the RPC headers, and encapsulates them in either HTTP or HTTPS, and sends them to the Outlook client. The Outlook client then strips the HTTP from the session and reads the traffic as RPC and MAPI.

Figure 2-6. Virtual connection



Outlook prefers to make connections with TCP/MAPI. Despite configuring Outlook to use Outlook Anywhere (RPC over HTTP) connections, Outlook sometimes checks to see if it can connect to the Exchange server with a TCP/MAPI connection.

Note: To disable fallback to TCP, enter an in-path rule on the client-side SteelHead to prevent EPM to the Outlook Anywhere server. An in-path rule to deny MAPI TCP is not normally used in most field deployments, as HTTP and HTTPS are the only protocols allowed through the firewall.

To block RPC to the Outlook Anywhere server

- On the client-side SteelHead, connect to the CLI and enter the following command:

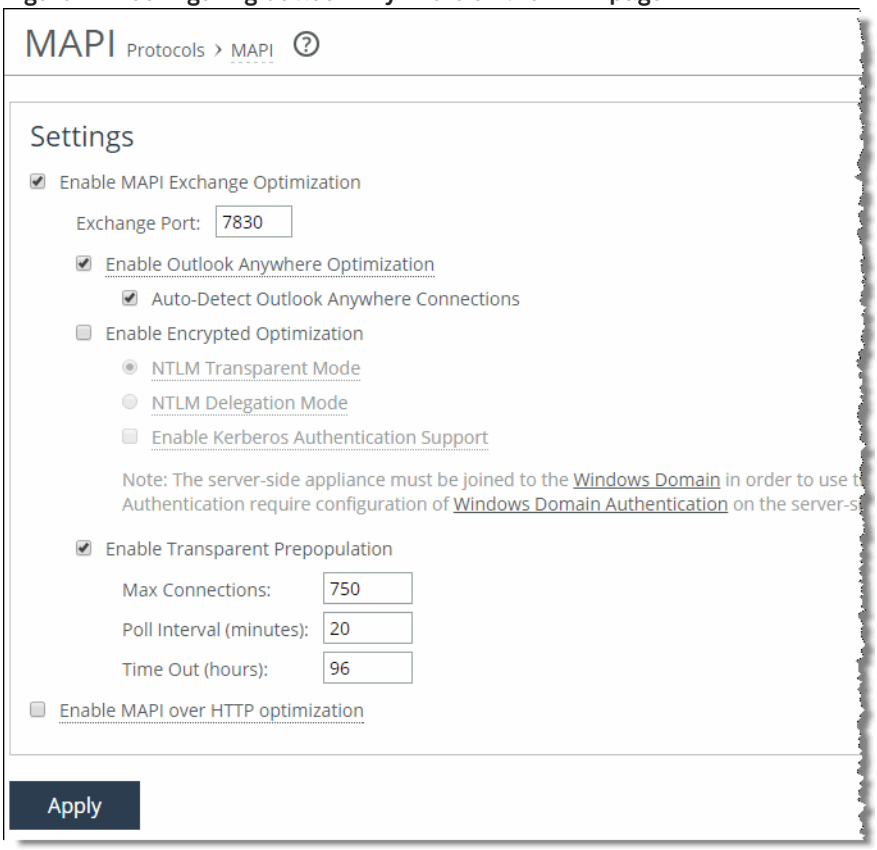
```
in-path rule deny rulenum [number] dstaddr [address] dstport 135 srcaddr 0.0.0.0 description
BlockRPC
```

To configure Outlook Anywhere

1. Configure Outlook Anywhere MAPI:
   - On the client-side and server-side SteelHead, choose Optimization > Protocols: MAPI.
   - Select Enable Outlook Anywhere optimization.
   - Select Auto-Detect Outlook Anywhere Connections.
   - Click **Apply**.

The corresponding CLI commands are **[no] protocol mapi outlook-anywhr enable** and **[no] protocol mapi outlook-anywhr auto-detect**.

Figure 2-7. Configuring Outlook Anywhere on the MAPI page



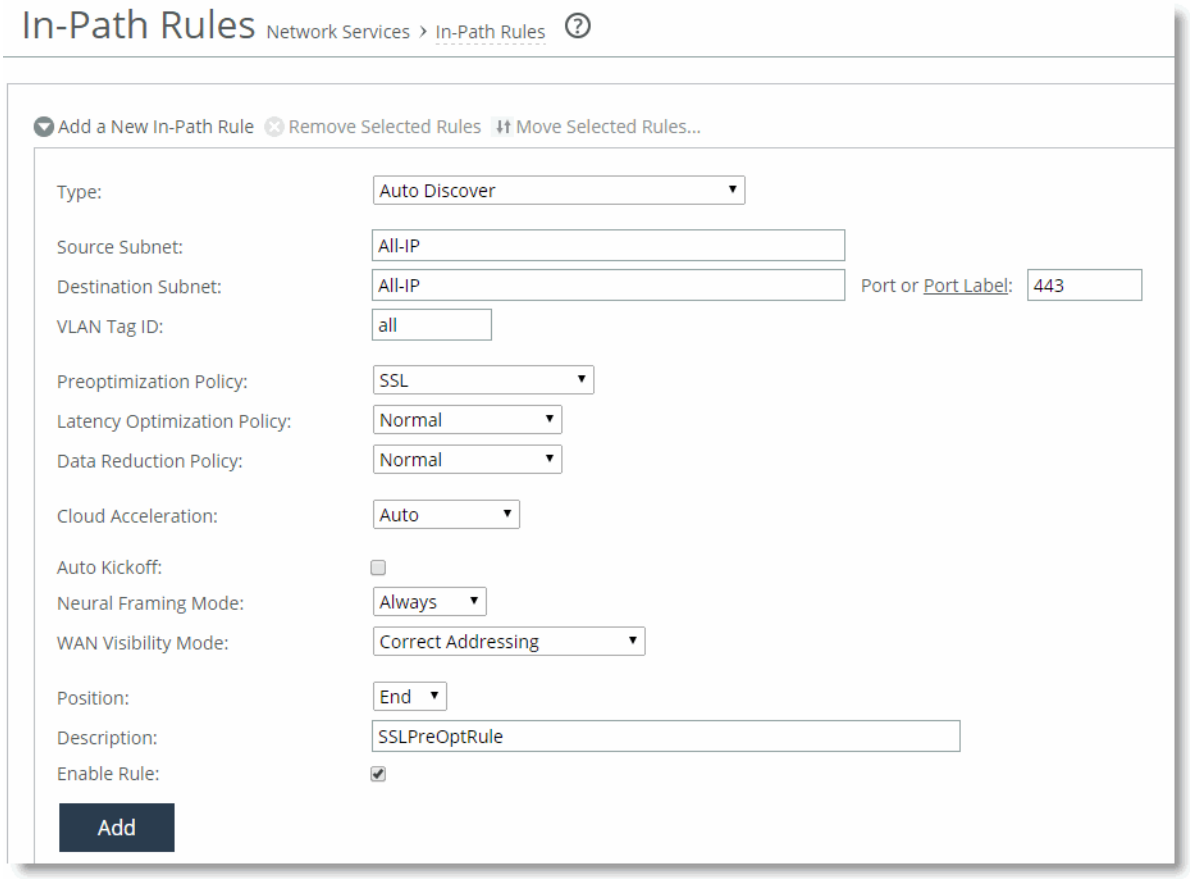
2. Configure an in-path rule for HTTPS connections to enable SSL preoptimization only if the SteelHead has not had port 443 removed from the port label Secure. Normally port 443 is removed as part of the simple SSL configuration. For more details, see "Setting up a simple SSL optimization deployment" on page 177.

   To configure an in-path rule for HTTPS connections:

   - Choose Optimization > Network Services: In-Path Rules.
   - Select Add a New In-Path Rule.
   - Select Auto Discover from the Type drop-down list.
   - Specify port 443.
   - Select SSL from the Preoptimization Policy drop-down list.
   - Click **Add**.

You can configure an in-path rule for HTTPS connections to enable SSL preoptimization through the CLI by entering **in-path rule auto-discover preoptimization ssl dstport 443 rulenum end description SSLPreOptRule**.

Figure 2-8. SSL in-path rule for Outlook Anywhere optimization



3. Enable HTTP optimization the client-side and server-side SteelHead. For details, see “HTTP Optimization” on page 75.

4. Enable SSL on the client-side and server-side SteelHead. The certificate and key from the Outlook Anywhere server must be installed on the server-side SteelHead.

   If you are using an internal CA, the CA root certificate must be installed.

   If you are using encrypted MAPI, you must enable secure inner channel. For details, see “Microsoft Exchange Email Optimization” on page 21.

Note: When in-path rules are applied at the Management Console, or when you use the CLI, you must complete a save operation to save your changes permanently.

Support for Common Access Card

A Common Access Card (CAC) is a type of smart card often used in military and other high-security environments. The cards are usually about the size of a credit card and contain a chip that holds one or more public-key infrastructure (PKI) certificates and an identifier that is unique to the card holder.

You can configure Outlook clients and Exchange servers so that they communicate only with each other after the user has inserted the CAC into a card reader attached to the client and entered a PIN. When a CAC is used, the communication between the Outlook client and the Exchange server is sent over a secure connection known as *S-Channel*.

You can configure SteelHeads to optimize traffic over S-Channel as part of an Outlook Anywhere over SSL (RPC over HTTPS) session. You must first configure your SteelHeads to optimize Outlook Anywhere over SSL, including installing the correct certificates.

After you complete this initial configuration, you must configure the server-side SteelHead to enable support for S-Channel:

```
protocol mapi outlook-anywhr schannel enable
```

Next, configure the client-side SteelHead to enable support for passive key derivation (PKD):

```
protocol ssl client-cer-auth enable
```

Verifying connection status

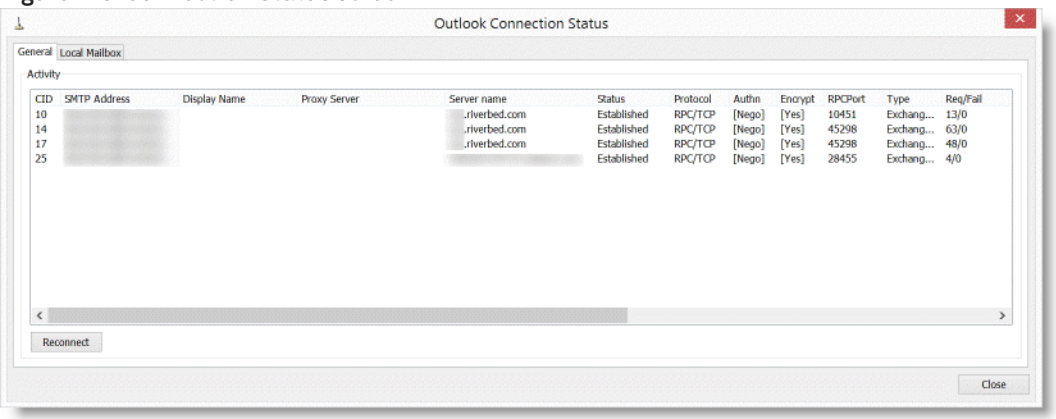
You can verify the Outlook connection status. It is important that you verify that the connection state is HTTP or HTTPS, and not TCP.

To view the Outlook connection status

1. Hold the Control key and click the **Outlook** icon in the task tray.
2. Select Connection Status.

   The Connection Status screen opens.

Figure 2-9. Connection Status screen



View the Reports > Networking: Current Connections page to identify if the active connections are standard MAPI (MAPI-OA) or encrypted MAPI (eMAPI-OA).

Figure 2-10. MAPI-OA Current Connections page

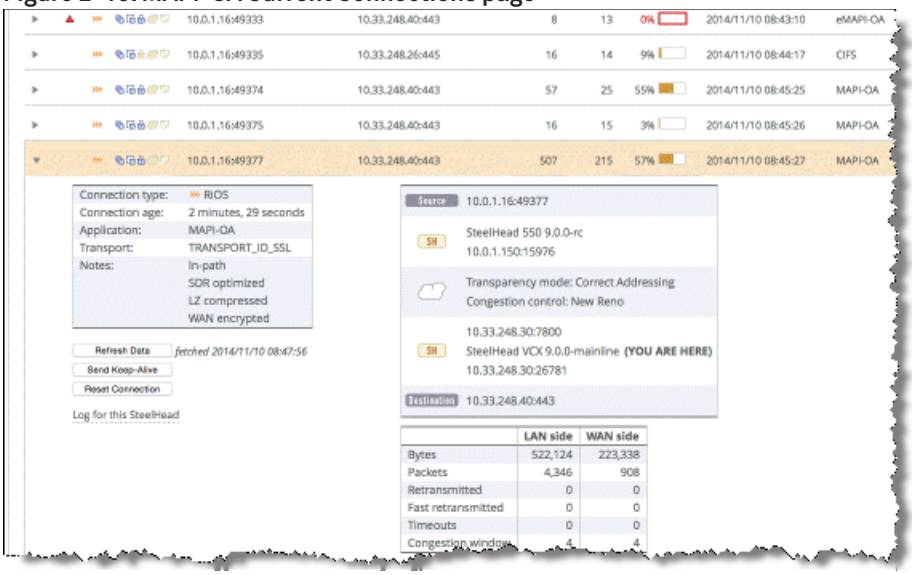
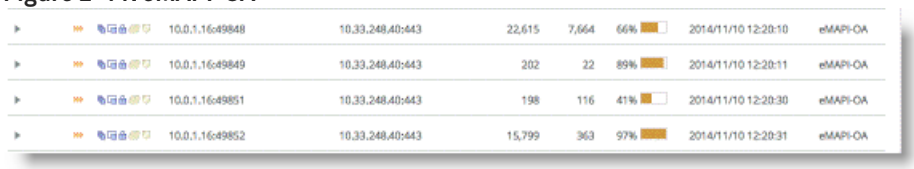


Figure 2-11. eMAPI-OA



Troubleshooting Outlook Anywhere optimized traffic

The following list describes how to troubleshoot issues with Outlook Anywhere optimized traffic.

- Change the Outlook Anywhere server to use HTTP rather than HTTPS
- Disable encrypted MAPI on the Client Access server running Outlook Anywhere

We recommend that you disable SSL and MAPI encryption to facilitate troubleshooting and easily decipher packet captures.

To change the Outlook Anywhere server to use HTTP only

1. Change or create a registry key on the Outlook Anywhere server.

   The key is located at HKLM\Software\Microsoft\Rpc\RpcProxy\AllowAnonymous. It is a DWORD value set to 0x1. Copy and paste the following into a.reg file for faster implementation.

   ```
   Windows Registry Machine Editor Version 5.0
   [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Rpc\RpcProxy]
   "Allow Anonymous"=dword:00000001
   ```

2. Restart the Outlook Anywhere server you modify the registry (by either manual or automated registry modification).

By default, Exchange 2010 only accepts encrypted MAPI connections. This is not a problem for the SteelHead because of the encrypted MAPI feature. However, for troubleshooting purposes, we recommend that you disable encrypted MAPI.

To disable MAPI Encryption on Exchange 2010

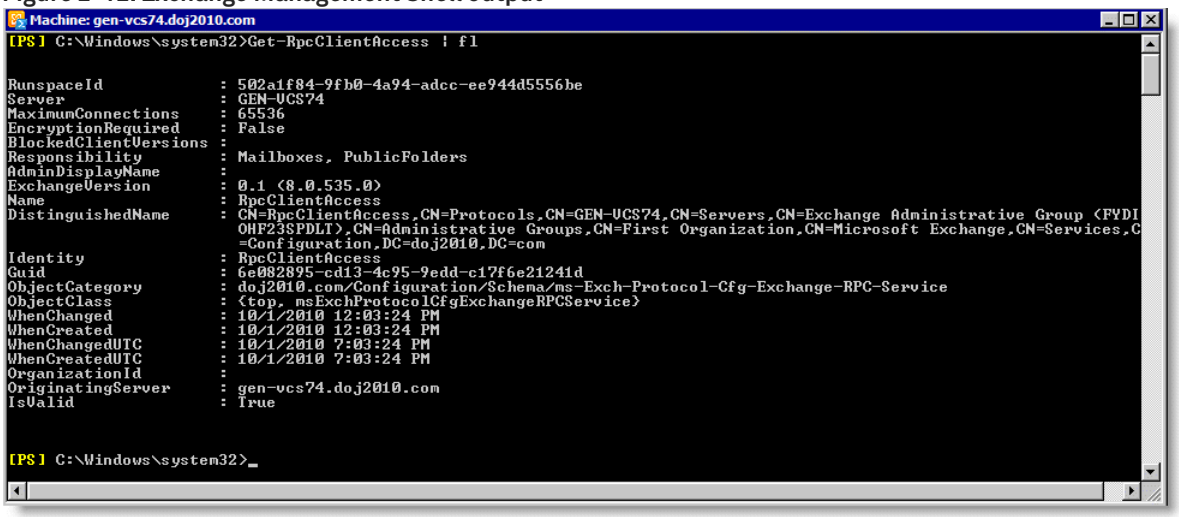
1. To determine if MAPI Encryption is enabled, launch the Exchange server management shell by choosing Start > All Programs > Exchange Server 2010 > Exchange Management Shell.

2. Enter the following command:

   ```
   Get-RpcClientAccess | fl
   ```

   Figure 2-12 shows an example output.

Figure 2-12. Exchange Management Shell output



3. To determine if MAPI encryption is enabled, view the EncryptionRequired field. It has a value of True or False. A True value means that encryption is required and you must change it to False. If a False value is present, you can exit the shell.

4. Enter the following command to change the value to False:

   ```
   set-RpcClientAccess -server mailbox -EncryptionRequired $False
   ```

5. Restart the Exchange server.

Microsoft Exchange 2013 optimization

Microsoft has made significant changes to Outlook MAPI-based traffic optimization in Microsoft Exchange 2013. Client access is simplified by eliminating the traditional EPM and MAPI protocols. This new implementation enables users to more easily access mailbox servers by leveraging well-known protocols HTTPS and HTTP. This change in protocols also enables organizations to define tighter network security for their Exchange environments by allowing only HTTP and HTTPS traffic from client-facing networks.

Figure 2-13 shows an example of client traffic from Exchange 2013 and Outlook 2013 using Wireshark.

Figure 2-13. Client Traffic using Wireshark

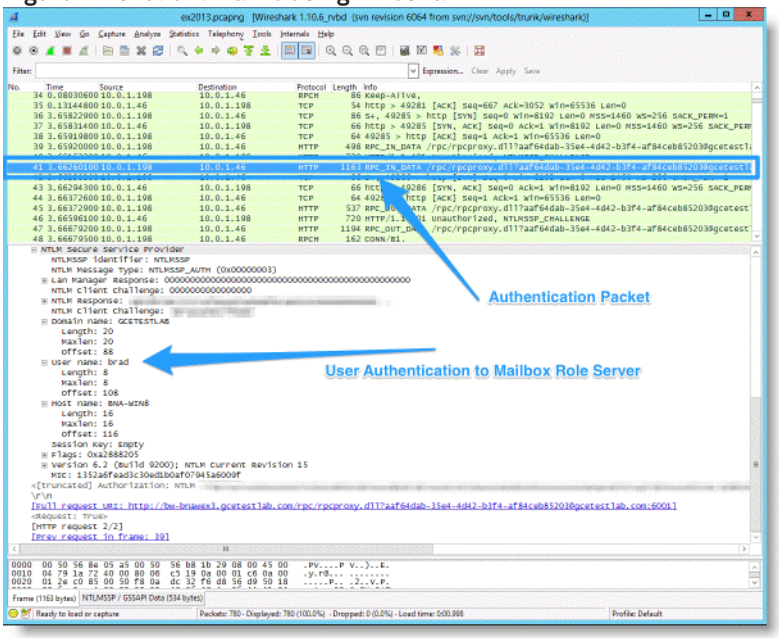


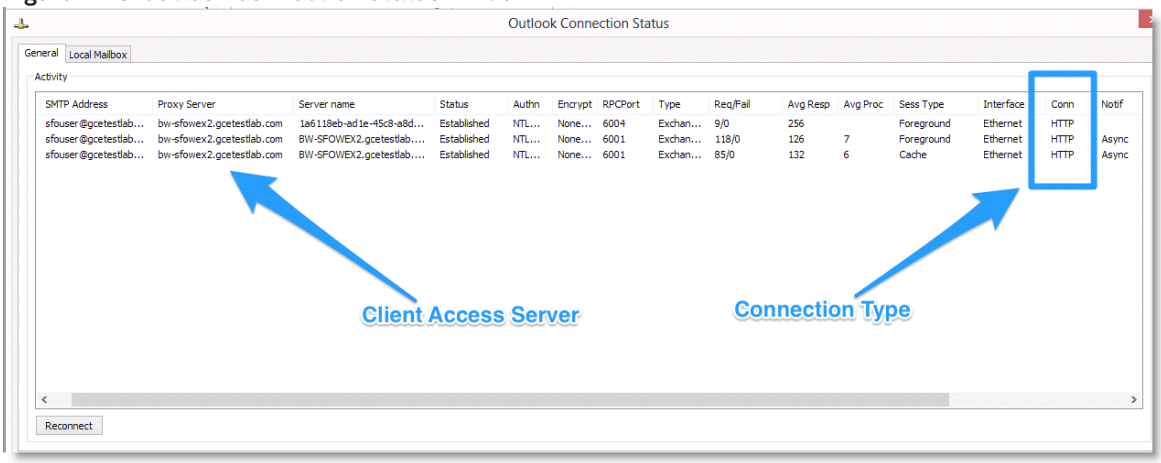
Figure 2-14 shows that the Exchange traffic is traversing HTTP TCP port 80 using a netstat -n output from the client running Outlook 2013.

Figure 2-14. Exchange Traffic on Port 80



Figure 2-15 shows the Outlook Connection Status window.

Figure 2-15. Outlook Connection Status window



To take advantage of the MAPI-based traffic optimization change, you must configure your SteelHeads to optimize Outlook Anywhere traffic. RiOS enables Outlook Anywhere-TCP streams to be initially intercepted by the HTTP protocol optimization function, and then it transparently switches to Remote Procedure Call (RPC) over HTTP.

For information about configuring Outlook Anywhere, see "Outlook Anywhere optimization" on page 30.

MAPI over HTTP

This section provides information about MAPI over HTTP. It contains the following topics:

- "MAPI over HTTP requirements" on page 39
- "MAPI over HTTP down negotiation" on page 40

Upon the release of Outlook 2010 update (KB 2878264), Outlook 2013 SP1, and Exchange Server 2013 SP1, Microsoft added a new dialect to the Exchange communication protocol. Rather than placing a wrapper around remote procedure call (RPC) in HTTP, Microsoft created a full version of MAPI that can be wrapped in HTTP. This new configuration is officially called *MAPI over HTTP transport protocol*.

To use this new dialect with the Outlook and Exchange Server versions, you must enable MAPI over HTTP on all Client Access Servers (CAS) across the Exchange deployment. With Exchange Server 2016, MAPI over HTTP is enabled by default at the organization level. Even with MAPI over HTTP enabled, the Exchange server can still use RPC over HTTP (Outlook Anywhere) for any Outlook clients that are not MAPI over HTTP capable.

MAPI over HTTP reduces the overall load on the Exchange environment by decreasing the client-to-server connections to two per MAPI-over-HTTP connection. MAPI over HTTP makes the entire conversation stateless. A client can change to a different CAS throughout the lifetime of the session to the Exchange infrastructure and not have to authenticate on each new server connection. This protocol change is a significant improvement compared to the legacy Microsoft RCP over HTTP protocol.

In RiOS 9.2 and later, the SteelHead can:

- optimize MAPI over HTTP with full-latency optimization.
- down negotiate MAPI over HTTP to RPC over HTTP.

Full MAPI over HTTP latency optimization is provided on the SteelHead by the MAPI over HTTP feature. This feature includes latency optimization, data reduction, and TCP optimization. MAPI over HTTP has acceleration parity with the legacy native MAPI (RPC over TCP) optimization feature with the exception of MAPI prepopulation.

The down negotiation feature removes information from the Outlook client request that is being sent to the Exchange server and forces the connection to use RPC over HTTP (Outlook Anywhere). RPC over HTTP is still available on the Exchange server to support Outlook clients that are not MAPI-over-HTTP capable. Remember that one of the improvements made by Microsoft with MAPI over HTTP is that it uses fewer TCP connections compared to previous Outlook-to-Exchange dialects. Therefore, when you use down negotiation, you increase the connection count compared to MAPI over HTTP.

For more information about down negotiation, see "MAPI over HTTP down negotiation" on page 40.

MAPI over HTTP requirements

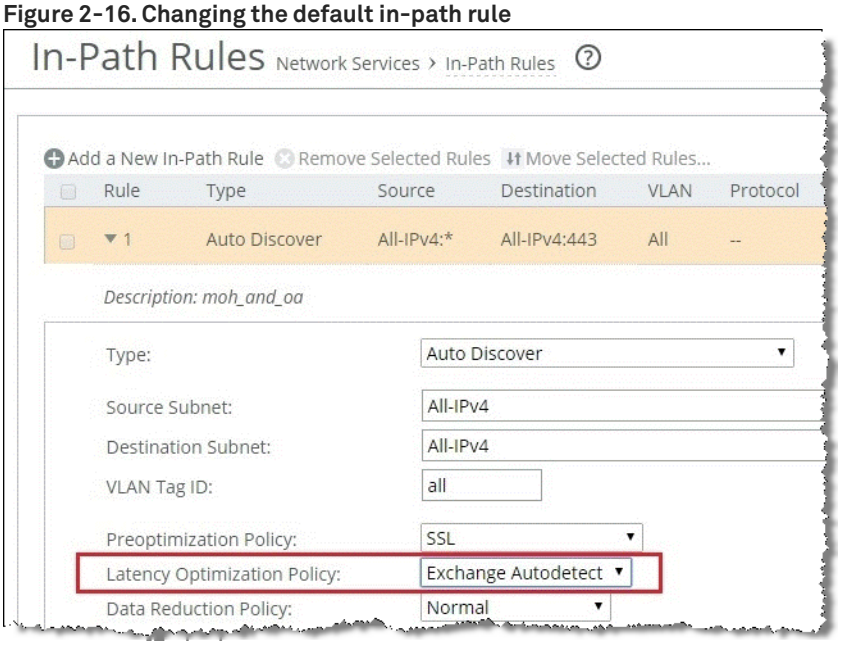
To use MAPI over HTTP, you must have RiOS 9.1 or later on the client-side and server-side SteelHeads. For RiOS versions between 8.5.2a and 9.0, SteelHeads optimize MAPI over HTTP connections, but reports display the application type as HTTP instead of MAPI-HTTP.

Note: For RiOS 9.7 and later, you do not need to explicitly set the latency optimization policy to Exchange Autodetect for MAPI over HTTP traffic optimization.

To optimize MAPI over HTTP traffic for RiOS versions prior to 9.7, you must change the default in-path rule to enable Exchange Autodetect (Figure 2-16), which is a latency optimization policy. Exchange Autodetect automatically detects MAPI transport protocols (Autodiscover, Outlook Anywhere, and MAPI over HTTP) and HTTP traffic. The Exchange Autodetect latency optimization policy should only be applied to HTTP traffic. If you do not enable this policy, the MAPI over HTTP feature does not engage and connections in the current connections page shows as HTTP only, with minimal data reduction.

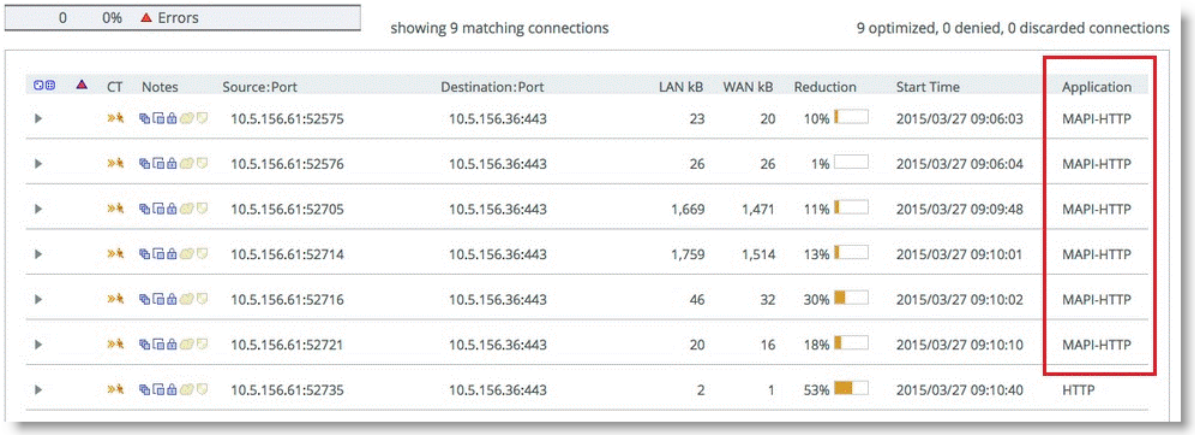
SSL is required for MAPI over HTTP to work properly. For more information about SSL, see "SSL Deployments" on page 169.

Figure 2-16. Changing the default in-path rule



When you properly configure the in-path rule, the SteelHead Current Connections report looks similar to Figure 2-17.

Figure 2-17. MAPI over HTTP connections



CT	Notes	Source:Port	Destination:Port	LAN kB	WAN kB	Reduction	Start Time	Application
		10.5.156.61:52575	10.5.156.36:443	23	20	10%	2015/03/27 09:06:03	MAPI-HTTP
		10.5.156.61:52576	10.5.156.36:443	26	26	1%	2015/03/27 09:06:04	MAPI-HTTP
		10.5.156.61:52705	10.5.156.36:443	1,669	1,471	11%	2015/03/27 09:09:48	MAPI-HTTP
		10.5.156.61:52714	10.5.156.36:443	1,759	1,514	13%	2015/03/27 09:10:01	MAPI-HTTP
		10.5.156.61:52716	10.5.156.36:443	46	32	30%	2015/03/27 09:10:02	MAPI-HTTP
		10.5.156.61:52721	10.5.156.36:443	20	16	18%	2015/03/27 09:10:10	MAPI-HTTP
		10.5.156.61:52735	10.5.156.36:443	2	1	53%	2015/03/27 09:10:40	HTTP

MAPI over HTTP down negotiation

We recommend that you adopt MAPI over HTTP natively. However, there are certain instances in which you might require RPC over HTTP connections. For example, your server-side SteelHeads might not yet be enabled to support MAPI over HTTP optimization.

To enable MAPI over HTTP down negotiation, you must meet the following requirements:

- The client-side SteelHead must be running RiOS 9.1 or later.
- You must enable the Exchange Autodetect latency optimization policy.
- You must enter the **protocol eos moh down-negotiate enable** command on the client-side SteelHead.

Note: Microsoft is deprecating RPC over HTTP in Exchange Online deployments as of October 31, 2017, and this command is not supported in Exchange Online deployments as of that date. Details are at https://support.microsoft.com/en-us/help/3201590/rpc-over-http-deprecated-in-office-365-on-october-31,-2017. This change only affects customers using Outlook for Windows to connect to Exchange Online mailboxes in Office 365. Customers using RPC over HTTP to connect Outlook for Windows and on-premises Exchange Server continue to do so.

To see the status of the MAPI over HTTP engine, enter the **show protocol eos** command.

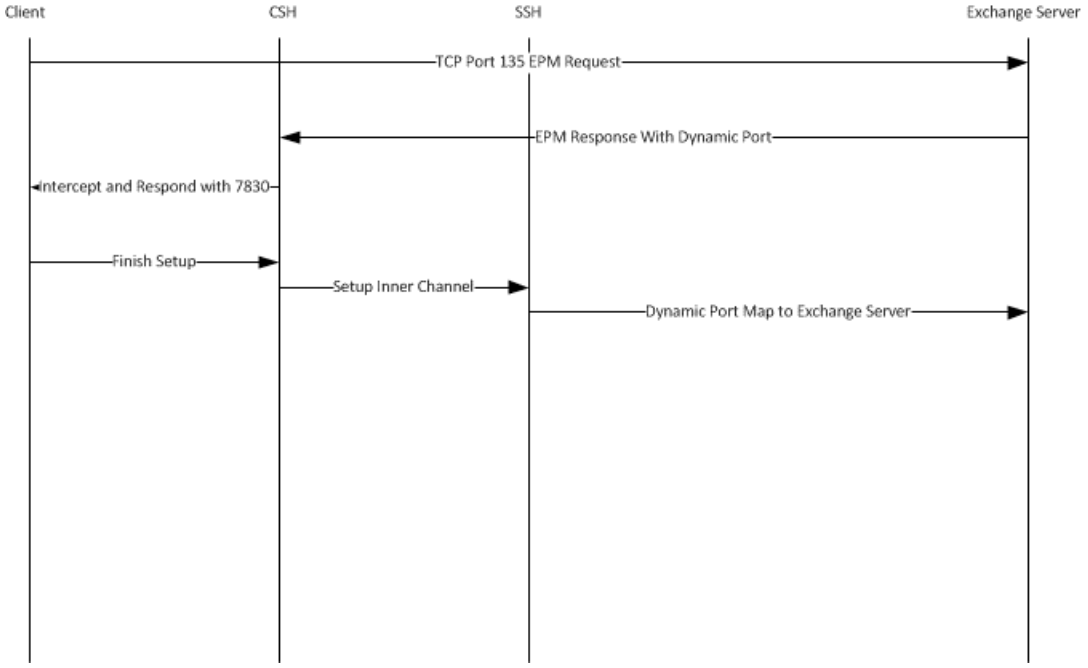
MAPI destination port handling

If you place an Exchange server behind a firewall, you must use static MAPI ports for the firewall to statefully inspect traffic to and from the MAPI servers. Default dynamic port mapping is not available in this scenario. You must enable static MAPI ports to Exchange Client Access servers that are placed behind load balancers for RiOS to correctly track connections.

A typical MAPI connection has the following flow:

1. The client initiates a connection to the Exchange server on port 135. The EPM requests a dynamic port from the Exchange server.
2. The Exchange server responds with a dynamic port from which the client can connect to MAPI.
3. The client-side SteelHead intercepts the dynamic port response and moves it to port 7830. Port 7830 is the standard SteelHead MAPI port.
4. The client finishes initiating the MAPI connection with the client-side SteelHead.
5. The inner channel connection or optimized connection is established between the client-side SteelHead and server-side SteelHead.
6. The server-side SteelHead finishes the connection to the Exchange server on the dynamic port.

Figure 2-18. Flow of a typical MAPI connection



In high-security environments it is desirable to hard code the Exchange server port in the event that multiple Exchange servers need multiple static ports. With a firewall between the clients and the Exchange server, an issued dynamic port cannot always be interpreted. The default behavior of the SteelHead is to remap the port to a single dynamic port. This causes a problem for MAPI servers running on multiple defined ports and the operation fails.

To disable the remapping capabilities of the MAPI software

- On the client-side SteelHead, connect to the CLI and enter the following commands:

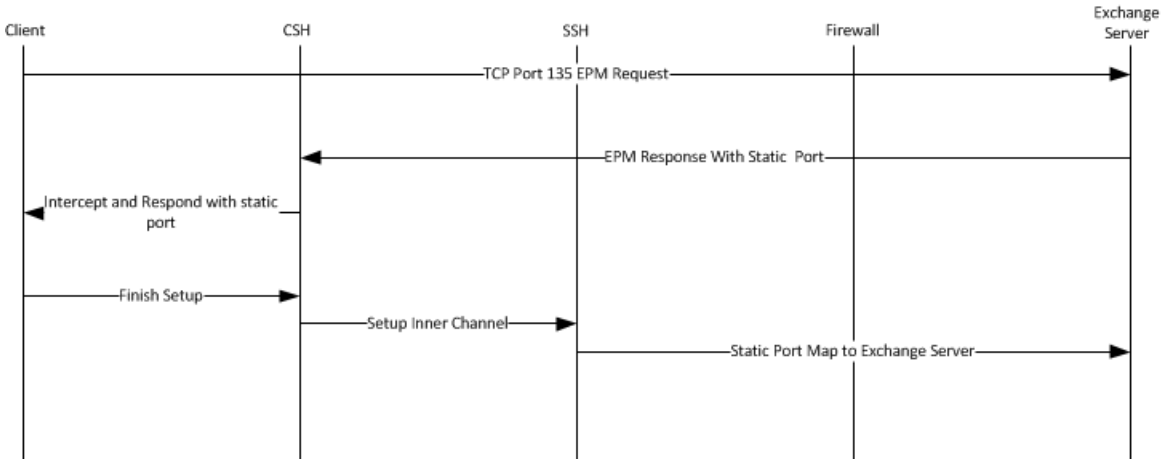
```
no protocol mapi port-remap enable
write memory
service restart
```

After you disable remapping capabilities of the MAPI software, MAPI has the following flow:

1. The client sends the EPM request to the Exchange server.

2. The Exchange server responds with a static MAPI port.
3. The client-side SteelHead intercepts the response.
4. The client-side SteelHead issues the client the Exchange server address and the assigned static port.
5. The client-side SteelHead finishes setting up the client side connection
6. The inner channel between the client-side SteelHead and the server-side SteelHead is completed.
7. The server-side SteelHead completes the connection to the Exchange server on the assigned static port.

Figure 2-19. MAPI flow with static ports



MAPI multiple context

Both MAPI and Outlook Anywhere enable multiple protocols to run over an individual TCP session and a TCP connection with the same TCP source and destination port. This feature minimizes the number of TCP connections consumed per client. *Multiple context* is when a client requests a new protocol over the same TCP connection. This technology has become increasingly prevalent with the adoption of Exchange 2013. RiOS 9.0 and later support multiple context. Prior to RiOS 9.0, multiple contexts caused the connection to enter pass-through mode and create errors in the log, similar to the following examples:

```
[rpch/mapi/csh/req.NOTICE] 1235184
{x.x.x.x:4149 x.x.x.x:4152} MSRPC Bind Request does not contain MAPI UUID, but f5cc5a18-4264-101a-
8c59-08002b2f8426 (NSPI)
```

and

```
[rpch/csh.NOTICE] 1235176
{x.x.x.x:4152 x.x.x.x:443} detected Exchange 365, but peer SteelHead does not have multi-context
support.
```

Note: You can identify multiple contexts in the header of the MAPI packet using a protocol decode tool, such as Wireshark.

RiOS 9.0 and later support multiple context. We recommend that you enable this feature in an Exchange 2013 environment. Enabling this feature does not have any adverse effect on nonmultiple context traffic.

To enable MAPI multiple context support

- On the SteelHead, connect to the CLI and enter the following commands:

```
(config) protocol mapi multi-context enable
(config) protocol mapi encrypted multi-context auth enable
(config) protocol mapi outlook-anywhr multi-context enable
```

Note: You need multiple SteelHeads in deployments in which both Office 365 and On-Premises Exchange traffic need optimization.

For more information about Outlook Anywhere, see “Outlook Anywhere optimization” on page 30.

3

Signed SMB and Encrypted MAPI Optimization

This chapter discusses high-level techniques and guidance for configuring signed Service Message Block (SMB) and encrypted MAPI traffic to ensure data integrity. This chapter includes the following sections:

- "Overview of SteelHeads optimizing with secure Windows traffic" on page 45
- "Windows security concepts" on page 46
- "Domain relationships" on page 47
- "Choosing an authentication mode for the server-side SteelHead" on page 49
- "Overview of configuring SMB signing and encrypted MAPI" on page 50
- "SMB3 optimization with Windows 8, Windows 2012 Server, and later operating systems" on page 51
- "Joining a SteelHead to a domain" on page 53
- "Kerberos" on page 56
- "Configuring the server-side SteelHead for Active Directory integrated (Windows 2003/2008)" on page 61
- "Best practices for the SteelHead in a secure Windows deployment" on page 62
- "Domain authentication scaling" on page 63
- "Domain health check and domain authentication automatic configuration" on page 65
- "Single domain example configuration" on page 72

Overview of SteelHeads optimizing with secure Windows traffic

Signed SMB and encrypted MAPI traffic use techniques to protect against unauthorized man-in-the-middle devices from making modifications to their exchanged data. Additionally, encrypted MAPI traffic and encrypted SMB3 traffic ensure data confidentiality by transmitting data with protection across the network. To securely optimize this traffic, a properly configured client-side and server-side SteelHead:

- decrypts and removes signatures on received LAN side data from the client or server.
- performs bandwidth and application layer optimization.
- uses the secure inner channel feature to maintain data integrity and confidentiality of data transmitted over the WAN.

- converts the received optimized data back to its native form.
- encrypts and applies signatures for LAN side transmission of data to the client or server.

To query the Windows domain controller for the necessary cryptographic information to optimize this traffic, the server-side SteelHead must join a Windows domain. The SteelHead can require other configurations, both on the SteelHead and in the Windows domain. This cryptographic information is only useful for the lifetime of an individual connection or session. The information is obtained at the beginning of a connection and transferred to the client-side SteelHead as needed, using the secure inner channel feature. You must configure the secure inner channel to ensure maximum security.

Only the server-side SteelHead is required to join the domain, and it does so using a machine account in the same way that a Windows device joins the domain using a machine account. The SteelHead joins the domain this way to obtain a client user session key (CUSK) or server user session key (SUSK), which allows the SteelHead to sign and/or decrypt MAPI on behalf of the Windows user that is establishing the relevant session.

The server-side SteelHead must join a domain that is either:

- the user domain. The domain must have a trust with the domains that include the application servers (file server, Exchange server, and so on) you want to optimize.
- a domain having a bidirectional trust with the user domain, or a one-way trust such that the user domain is the trusted domain.

The domain that the server-side SteelHead is joined to might include some or all of the Windows application servers (file server, Exchange server) for SteelHead optimization. The server-side SteelHead can only be a member of one domain, but where suitable domain trusts exist, it can also optimize connections for Windows application servers that reside in other domains. For more information, see “Domain relationships” on page 47.

Production deployments can have multiple combinations of client and server Windows operating system versions, and they can include different configuration settings for signed SMB and encrypted MAPI. Therefore it is possible that the security authentication between clients and servers can use NT LAN Manager (NTLM) or Kerberos, or a combination of the two. This chapter includes more information about authentication types and SteelHead configuration requirements.

Windows security concepts

The Windows security framework is based on a formal structure known as a *domain*. Inside the domain is a logical group of host resources (primarily clients and servers, but also printers and other peripherals) that share a central directory database. The database resides on one or more servers known as *domain controllers* and contains user accounts and security information for all the resources in the domain. Other domains can coexist alongside and are joined together through a *Trust Relationship* to allow resources to securely communicate between each other even though they are in different domains with their own domain controllers.

When users and clients are accessing server resources (like file servers and Exchange servers), their credentials are validated against the database on the domain controller. This validation ensures that the client and the user have the correct security privileges to be able to access resources that provide signed SMB traffic or encrypted MAPI traffic.

Several techniques and protocols are used to validate the credentials of the user and client. These techniques vary according to the Windows operating system version and application configuration (for example, Microsoft Outlook and Microsoft Exchange) on both the client and server. Example protocols include Kerberos, NTLMv1, and NTLMv2. The earliest suitable RiOS version required to perform optimization in a Windows security environment is based on the authentication protocol used. Because Riverbed has added more capabilities in newer versions of RiOS, we recommend that you use the most recent version that meets your needs.

Domain relationships

Some organizations might have more than one Windows domain in use in their environment. A SteelHead, like a Windows server, can only join a single domain. Therefore, the choice of which domain the SteelHead should join depends on the domain where the file or mail servers are located, the type of trust relationship between the SteelHeads potential domain, the file or mail servers domain, and the domain containing a user's credentials.

Figure 3-1 shows an example of a simple, single domain structure. All resources (such as clients and servers) that have joined the same domain are subject to the domain permissions and authentications, and they can access the other available resources. Only a single domain controller is required in this case, although you can have more than one domain controller for resilience. The SteelHead must join the one available domain as a precursor to secure Windows protocol optimization.

Figure 3-1. Single domain structure

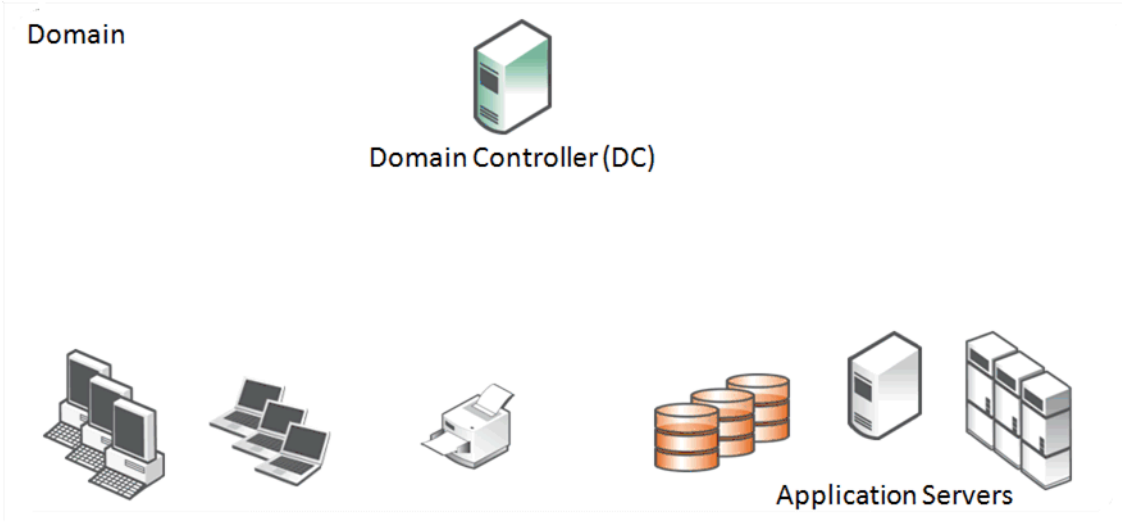


Figure 3-2 shows an example where clients are in one domain and servers are in a second domain. There is a trust relationship between the two domains that allow the clients and servers to access each other. The trust relationship is a *one-way trust*. The client domain is described as the *Trusted Domain* and the server domain is described as the *Trusting Domain*. The arrow that indicates the direction of trust is from the trusting domain to the trusted domain. Because of the one-way trust, only the resources in the client domain are allowed to access the resources in the server domain and not the other way around. Each of the two domains has its own domain controller, each with their own database for the resources in its domain.

Figure 3-2. Two domains and one-way trust structure

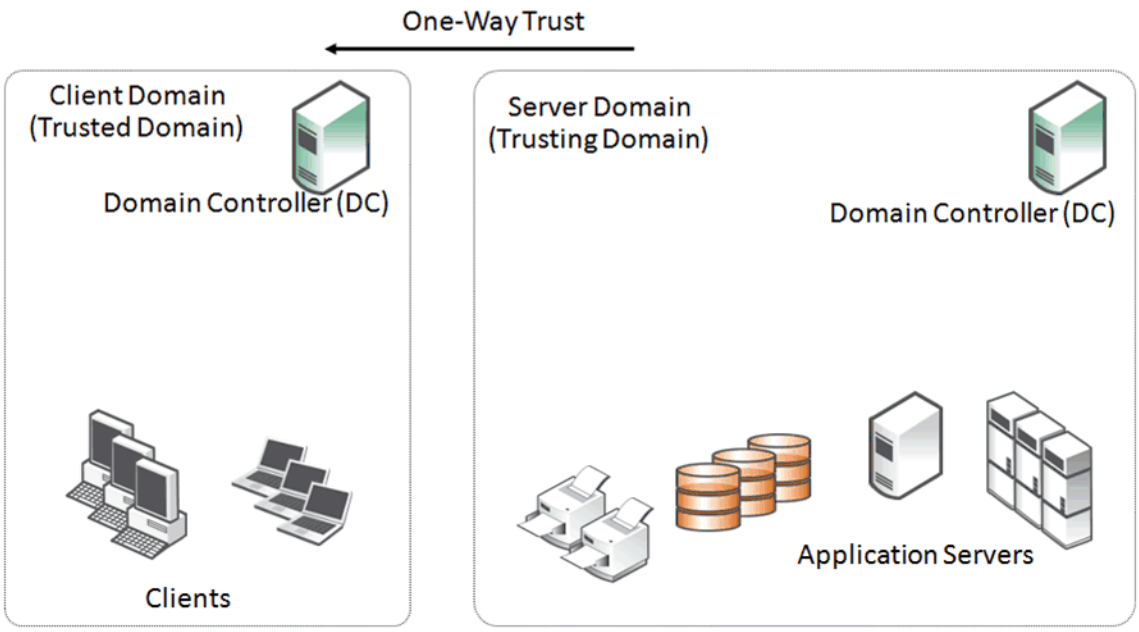
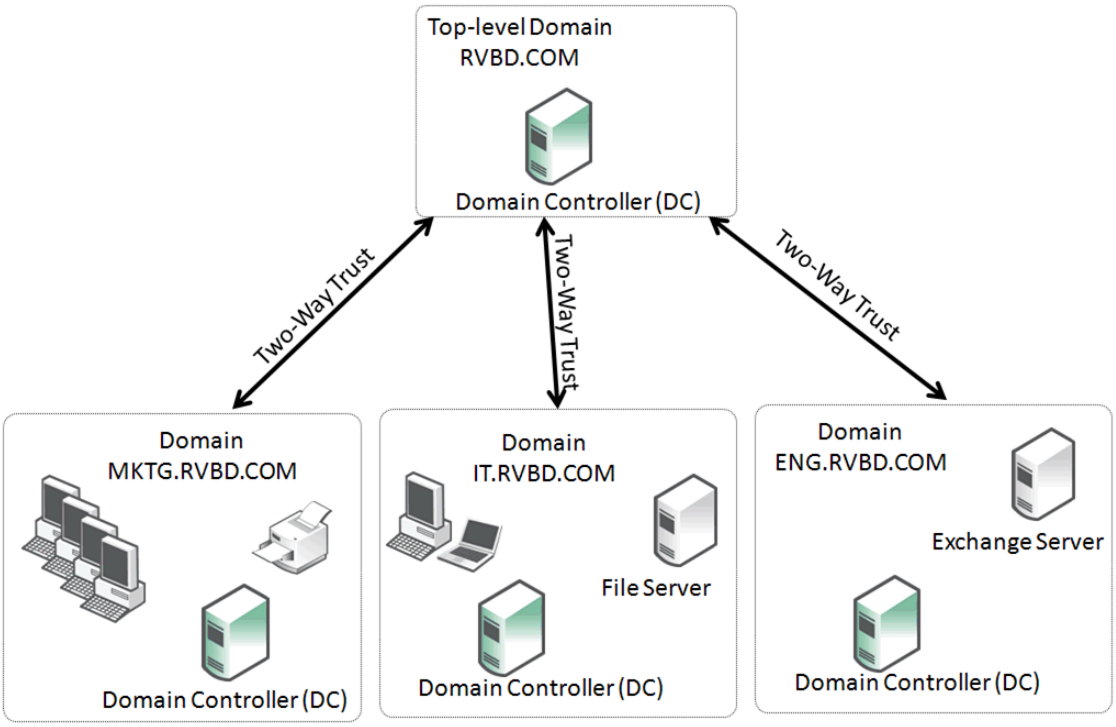


Figure 3-3 shows an example of a configuration where there are multiple domains. In a two-way trust, each of the resources in the child domains can access the other domains through the top-level (parent) domain as long as the correct permissions have been set up within the child domains.

Figure 3-3. Multiple domains with two-way trust structure



There can be a mixture of domains with domain controllers with different Windows operating system versions and a mixture of trust types. *Native mode* is when all domain controllers run the same version of operating system, and *mixed mode* is when there are different versions of the operating system.

Choosing an authentication mode for the server-side SteelHead

There are many combinations and settings for the Windows client operating system, Windows domain level, Windows authentication method, and RiOS version. You can choose from several different configuration options on the server-side SteelHead. In some cases, you can select multiple options to provide the best possible flexibility for your deployment. The options are as follows:

- **Transparent mode** - Authenticate using NTLM pass-through authentication to the Windows domain controller. This option is considered the easiest to deploy with the minimum of administrative overhead. To use this mode with RiOS 7.0 and later, you can join as Active Directory integrated (Windows 2008 and later) or Active Directory integrated (Windows 2003). We recommend that you use Active Directory integrated (Windows 2008 and later) because this mode enables the server-side SteelHead to securely communicate with domain controllers in other trusted domain in which there may be additional servers requiring optimized connections. For more information about join types, see "Joining a SteelHead to a domain" on page 53.

- **End-to-end Kerberos** - When you need the authentication between Windows clients and servers to be Kerberos from end to end, the server-side SteelHead can make use of a replication user account in the Windows domain. While this is the only choice available when the authentication is required to be end-to-end Kerberos, you can combine it with transparent mode to provide the most flexibility in the event that some clients still negotiate NTLM authentication.
- **Delegation mode** - This is a legacy configuration option within RiOS on the SteelHead. The feature is retained for reasons of backward compatibility. Riverbed does not recommend that you use the configuration for deployments using versions later than RiOS 7.0. For information, see "Legacy Delegate User Configuration" in Appendix A of the *SteelHead Deployment Guide - Protocols*, July 2015.

Overview of configuring SMB signing and encrypted MAPI

A deployment that successfully optimizes SMB signing and encrypted MAPI traffic requires gathering information about the Windows environment, configuring the SteelHeads with the appropriate RiOS version and configuration; when using delegation mode or end-to-end Kerberos, this same deployment potentially requires more configuration in the Windows domain.

The following is a general overview on how to configure a typical SMB signing and MAPI encrypted deployment (for complete details, see the *SteelHead User Guide*):

1. Determine if the domain containing users to optimize is different from the server domain.

   If the user domain is different from the server domain, determine the trust relationship between the domains.

2. Gather version information about the Windows environment:

   - Client operating system versions.
   - Domain functional level of all server domains involved in the configuration (for example, Native 2003, mixed mode, Native 2008-R2, and Native 2012).

   Typical environments allow the Windows client and server to negotiate either NTLM authentication or Kerberos authentication for SMB or MAPI traffic. In these environments, the SteelHead can force negotiation to NTLM authentication to provide latency and bandwidth optimization. Certain environments do not negotiate NTLM or Kerberos—instead they require Kerberos authentication. The Kerberos authentication is set either through settings on the client or through domain policy settings. RiOS 7.0 and later support this type of end-to-end Kerberos authentication. If your environment is Kerberos only, see "Kerberos" on page 56.

3. For the best compatibility with the current versions of Microsoft Windows Active Directory environments, we recommend using a minimum of RiOS 7.0 on both the client-side and server-side SteelHeads. Additionally, Riverbed has incorporated new features designed to simplify the configuration and management of SteelHeads that are integrated with Active Directory. These features are available in RiOS 8.6.x and 9.x, therefore consider upgrading to a suitable recent version.

   If you must use a version prior to RiOS 7.0, see Appendix A of the *SteelHead Deployment Guide - Protocols*, July 2015.

4. Upgrade the RiOS version on the client-side and server-side SteelHeads, if necessary.

5. If not already implemented, configure the secure inner channel feature on both the client-side and server-side SteelHead.

6. Join the server-side SteelHead to the domain. The domain can be the user domain or a domain that has a trust with the user domain: for example, the server domain. For details, see "Joining a SteelHead to a domain" on page 53.

   In environments where the server domain only has a one-way trust to the user domain, you need a special configuration when joining the server-side SteelHead to a domain. For details, see "One-way trust configuration" on page 55.

7. If delegation mode is required, set up the delegation accounts in the server domains and configure Delegation mode on the server-side SteelHead. For information, see "Configuring Constrained Delegation for Delegation Mode" in Appendix A of the *SteelHead Deployment Guide - Protocols*, July 2015.

8. Verify successful optimization of the SMB signed or encrypted MAPI traffic.

SMB3 optimization with Windows 8, Windows 2012 Server, and later operating systems

Microsoft created an update to the SMB protocol with the release of Microsoft Windows 8 and Server 2012 operating systems. This new version is officially known as SMB3, but in some early documentation it might be referred to as SMB 2.2.

The following new features included in SMB3 are relevant to your SteelHead deployment:

- Encryption between client and server
- New SMB signing algorithm
- Secure dialect negotiation

You can use all three features only if the client and server are Windows 8 and Windows 2012 servers respectively. If the client is Windows 7 or earlier, or the server is Server 2008-R2 or earlier, the client and server do not use the new features for reasons of backward compatibility—with the exception of secure dialect negotiation.

Secure dialect negotiation is enabled and used by default in Windows 8 and Server 2012 connections. You can optionally use secure dialect negotiation with SMB2 when you are setting up a connection to a server running Server 2008-R2, but this is not normally enabled for compatibility reasons.

With the release of Windows 8.1 and Windows Server 2012 R2, Microsoft has added support for SMB 3.02.

For SMB3 and SMB 3.02 specifications, go to http://msdn.microsoft.com/en-us/library/cc246482.aspx.

RiOS versions prior to 8.5 provide some limited optimization of SMB3, but in some cases there is no optimization at all. For more information about SMB3 and releases prior to RiOS 8.5, go to https://supportkb.riverbed.com/support/index?page=content&id=S16547.

In RiOS 8.5 and later, you can provide full optimization for signed SMB3 traffic between Windows 8 clients and Server 2012 servers, as well as other client-server combinations that result in the use of signed SMB3 traffic.

Additionally, in RiOS 8.5 and later, you can optimize SMB3 traffic that is encrypted at the file share or file server level.

To optimize SMB3 traffic, you must run RiOS 8.5 or later on both the client-side and server-side SteelHeads and join the server-side SteelHead to the domain. Additionally, you must enable SMB3 optimization on both the client-side and server-side SteelHeads.

Note: Even if you do not configure Windows 8 clients and Server 2012 servers to require signing and encryption, the SMB3 protocol continues to use secure dialect negotiation during the connection setup. Therefore, the configuration settings outlined in this section are mandatory to ensure SMB3 traffic is optimized.

If you correctly configure the SteelHeads to optimize signed SMB3 traffic, there is no additional configuration required to support the optimization of signed SMB3 traffic that is encrypted at the share or server level.

For more information about:

- the configuration settings required for SMB3, see the *SteelHead User Guide*.
- optimizing SMB traffic between combinations Windows 8.1 and Server 2012R2, go to https://supportkb.riverbed.com/support/index?page=content&id=S23689.

SMB 3.1.1 optimization

Further updates to the SMB protocol were included with Microsoft's release of Windows 10 and Windows Server 2016 operating systems. The default SMB protocol used between combinations of client and server running these operating system versions is SMB 3.1.1. Similar to SMB3 and SMB 3.02, encryption for SMB 3.1.1 can be enabled at the share or fileserver level. The default cipher is AES-128-GCM, but SMB 3.1.1 can negotiate to AES-128-CCM to support older configurations. The ability to negotiate the cipher (unlike SMB 3.02) means that Microsoft can chose to include additional crypto algorithms in the future.

Along with signing, encryption, and secure dialect negotiation included with previous SMB3 dialects, an additional mechanism was introduced with SMB 3.1.1 called *pre-authentication integrity*. This is a form of integrity check that occurs between two hosts when establishing an SMB 3.1.1 session. This check is a mandatory feature with SMB 3.1.1 and is designed to protect against any tampering of the client-server session setup by using SHA-512 cryptographic hashing. The hash is used as one of the inputs to the key derivation function from which the session's secret keys are obtained.

For more information on pre-authentication integrity, go to http://blogs.msdn.com/b/openspecification/archive/2015/08/11/smb-3-1-1-pre-authentication-integrity-in-windows-10.aspx. This page also discusses the use of WAN accelerators in the *Proxy and traffic accelerator* section.

To provide full Layer-7 latency optimization for SMB 3.1.1 connections, both the client-side and server-side SteelHeads must be running RiOS 9.2 or later and must have SMB2 and SMB3 latency optimization settings enabled. The server-side SteelHead must also be configured with the following items:

- SMB2 and SMB3 signing enabled.
- Joined to a suitable Windows domain, preferably using Active Directory integrated (Windows 2008 or later).
- NTLM Transparent Mode or Kerberos Authentication Support.
  - You can use NTLM Delegation Mode, but it is discouraged due to higher administration overheads compared with Transparent Mode or Kerberos.
- Suitable replication or delegation service accounts to match the authentication modes in use.

With the exception of the service accounts, you can complete all of the above settings on the server-side SteelHead by using the Configure Domain Auth Easy Config.

When you have the client-side and server-side SteelHeads running RiOS 9.2 or later, and have problems with the preauthentication integrity hash or negotiated encryption algorithm, then the connection between the client and server is blacklisted and subsequent sessions between the two hosts are passed through with no optimization.

Note: You must use RiOS 9.2 or later for complete Layer-7 latency optimization of SMB 3.1.1. Previous versions of RiOS might provide some degree of bandwidth optimization. For more guidance on the use of earlier RiOS versions and optimizing SMB 3.1.1 traffic between combinations of Windows 10 and Server 2016, go to https://supportkb.riverbed.com/support/index?page=content&id=S25594.

Joining a SteelHead to a domain

SteelHeads support joining domains that are operating in native or mixed modes for the following domain functional levels:

- Windows 2000
- Windows 2003 R2
- Windows 2008
- Windows 2008 R2
- Windows 2012
- Windows 2012 R2
- Windows 2016

For more information about maximum latency between a SteelHead and a domain controller, go to http://supportkb.riverbed.com/support/index?page=content&id=S22110.

Before you join a SteelHead to a domain, perform the following prerequisite verifications:

- The primary interface on a SteelHead must have IP reachability to any domain controllers, DNS servers, and other resources that a Windows server or workstation typically has in the domain that the SteelHead is joining. By default, the SteelHead uses the primary interface as the source interface for the join operation. Other interfaces, such as the auxiliary and in-path management interfaces, might work if enabled, but they are not supported for authentication traffic in general.

- The source interface for the SteelHead must not have its communication to the Windows domain controller blocked by a firewall or other security mechanism—for example, the RiOS Management ACL mechanism—either during the join or while optimizing connections. For information about the ports and protocols required for operation, go to http://support.microsoft.com/?id=179442.
- The SteelHead must be configured to use the same DNS server that other members of the domain use. Active Directory uses the DNS protocol to discover domain resources. When a SteelHead is joining a domain, it needs to make DNS based requests to a DNS server that understands these Active Directory-specific operations.
- The source interface for the join must have an A record in the DNS domain associating the SteelHead hostname to its IP address.
- The SteelHead must have its clock synchronized to within a few seconds of the domain controller it communicates with. The best practice is to have the SteelHead use the NTP protocol to synchronize with the NTP server that is used for clock synchronization within the Windows domain.

Note: A clock that is not synchronized on the SteelHead is the most common cause of domain join errors.

- Credentials for a domain account with sufficient privileges to join a machine to the domain must be temporarily available. Many organizations have specific accounts that are used only for the purpose of joining machines to the domain and have no other privileges. Enter the credentials on the SteelHead to perform the join operation. The credentials are not stored on the SteelHead. If you have multiple SteelHeads in the data center (for example, in a high-availability configuration), then you can add the same replication user account to the configuration of all the server-side SteelHeads. You can also create an account solely for the purpose of joining the domain, and can be used once. For more details, see https://supportkb.riverbed.com/support/index?page=content&id=S18097.

If the server-side SteelHead is running a version of RiOS between 6.1 and 6.5, it can join the domain only to appear as a *Workstation*. In RiOS 7.0 and later, the SteelHead can join the domain and appear as one of three different roles: Workstation, Active Directory integrated (Windows 2003), or Active Directory integrated (Windows 2008 and later).

Regardless of the domain functional level, as long as there is at least one domain controller running Windows 2008 or later, we recommend joining with the role of Active Directory integrated (Windows 2008 and later). As well as enabling the server-side SteelHead to communicate with domain controllers running Windows 2008 or later, this join type provides the broadest coverage of NTLM dialects used by clients and the ability for the SteelHead to operate more easily across multi-domain environments.

For more details, see "Configuring the server-side SteelHead for Active Directory integrated (Windows 2003/2008)" on page 61.

During the join operation, the SteelHead communicates with domain controllers and other entities related to the Windows domain, including the DNS server. The exact operations performed depend on the domain functional level, but are similar to the operations made when a new Windows server joins the domain.

The following process occurs during the domain join:

1. The SteelHead performs a lookup for Domain Controllers. It uses the DNS server that is specified in the Host Settings page of the SteelHead Management Console.
2. The SteelHead picks a domain controller from the list—usually the one that responds quickest.

3. The SteelHead establishes a short-lived SMBv1 and well-known *named pipe* session to the chosen domain controller.

4. Using these two short-lived sessions, and with the authority of the temporary user account entered into the Domain Join page of the SteelHead Management Console (or CLI), the SteelHead requests the flags for the UserAccountControl attribute for its machine account.

5. After the machine account is successfully created, the SteelHead establishes a long-lived secure channel to the NetLogon service of the chosen domain controller.

By default, the SteelHead appears in the Computers Organizational Unit (OU) in the domain. This is the default for Windows member servers joining a domain. You can use the CLI for a domain join and specify a different OU.

The following example shows a SteelHead joining the domain RVBD.COM and placing the SteelHead in the WAN-opt OU:

```
domain join domain-name RVBD.COM login join-account password join-password org-unit WAN-opt
```

One-way trust configuration

You need RiOS 6.1 or later when the user is in a different domain from the servers, and when the server domain has a one-way trust relationship with the user domain. In this scenario, the user domain is the *trusted domain*, and the server domain is the *trusting domain*. This setup is occasionally deployed in organizations that have a number of subsidiary companies or between two companies that have recently merged but wish to keep their IT separate for a period of time.

More typically this configuration is found when the user organization has file or mail services hosted by an external service provider. For example, with Microsoft Office 365 Dedicated service, this setup enables the SteelHead to use authentication of the Exchange servers that provide Microsoft Exchange online services.

In environments in which there is a one-way trust between server domains and user domains, the server-side SteelHead is joined to the user domain, or any domain that trusts the domain of the client user. Because of the one-way trust, the server-side SteelHead cannot automatically build a list of domains on the other side of the trust. Therefore, you must explicitly configure the server domains that trust the user domain on the server-side SteelHead, using the **protocol domain-auth oneway-trust** command.

For example, if the users in the RVBD.COM domain use file or mail servers in the SERVERS.PROVIDER.COM domain (whose shortened NetBIOS domain name is SERVERS), you first ensure that the server-side SteelHead is joined to the RVBD.COM domain and then use the following commands on the server-side SteelHead:

```
protocol domain-auth oneway-trust dns-name SERVERS.PROVIDER.COM netbios-name SERVERS
```

You can view the list of configured one-way trust domains with the **show protocol domain-auth oneway-trust** command and, if needed, you can remove domains with the **no protocol domain-auth oneway-trust dns-name** command.

In RiOS 7.0 and later, the support for one-way trusts is further enhanced to include Windows 7 clients without requiring a registry change on the client. You must join the server-side SteelHead to the domain using the Active Directory integrated (Windows 2003/2008) mode and then execute the CLI command as indicated previously.

Support for Kerberos authentication through a one-way trust is included in RiOS 8.5 and later. Versions of RiOS prior to 8.5 support only NTLM authentication in one-way trust configurations.

Enabling Kerberos in a restricted trust environment

When you use Kerberos authentication, there are some additional one-way trust restrictions if you use external managed services such as Microsoft Office 365 Dedicated. These restricted trust models are deliberately designed with split resource and management Active Directory domains.

Within a hosted services deployment, or any other deployment in which there is a restricted one-way trust, the server-side SteelHead cannot use the replication user account to contact the resource domain controller for the session key.

The server-side SteelHead joins the user account domain and uses the replication user account to communicate with the domain controller in the user domain.

In addition, the client-side and server-side SteelHeads use a Kerberos feature in RiOS 8.5 to intercept traffic over TCP port 88, the port most used by Kerberos.

By intercepting the Kerberos exchanges (which are initiated by the client when setting up an authenticated connection to a server), the server-side SteelHead can obtain a copy of the session key used between the client and server. This action enables the SteelHeads to optimize the signed SMB or encrypted MAPI session in a restricted trust deployment.

Note: The Kerberos feature must use TCP.

Windows XP clients, by default, use UDP for Kerberos authentication. You must reconfigure Windows XP clients to use TCP, if necessary.

For more information about configuring SteelHeads for restricted one-way trusts, see “Configuring the server-side SteelHead for Active Directory integrated (Windows 2003/2008)” on page 61 and the *SteelHead User Guide*.

Kerberos

Kerberos is a network authentication protocol. In a Microsoft Windows environment, the Active Directory domain controller maintains user account and login information to support the Kerberos service. From a corporate perspective, you can think of Kerberos as guarding against unauthorized access to your IT assets.

This section includes the following topics:

- “Overview of Kerberos” on page 57
- “Optimization in a native Kerberos environment” on page 59
- “Domain user with replication privileges” on page 60

- “Configuring traffic optimization for HTTP (SharePoint), encrypted MAPI, and signed SMB, SMB2, and SMB3” on page 61

Note: Kerberos authentication is not unique to a Microsoft Windows environment. This guide contains explanations and details limited to a Windows configuration.

Overview of Kerberos

The three components of Kerberos are the key distribution center (KDC), the client user, and the server resource with the service that the client user wants to access. The KDC is a part of the Windows domain controller and performs two service functions: the authentication service (AS) and the ticket-granting service (TGS).

To enable the client user to access a server resource (for example, a Windows file server), three exchanges are involved:

- The AS exchange
- The TGS exchange
- The AP (application protocol) exchange

When users initially log in to a Microsoft Windows network, they must provide a login name and password for access. These credentials are verified in the AS exchange of a KDC within the domain of the user. The KDC has access to Active Directory user account information. When a user is successfully authenticated, the authentication server grants the user a ticket to get tickets (TGT) that is valid for the local domain—you are first approved to get tickets, and next you are approved for each ticket as you ask for them. The TGT (sometimes referred to as the *AS reply*, although the AS reply actually contains more parts) has a default lifetime of 10 hours and can be renewed during the user's session without requiring the user to reenter a password.

The AS reply includes two components: the TGT itself, which is encrypted with a key that only the KDC (TGS) can decrypt, and the first of two session keys encrypted with the user's password hash. Any future communication with the KDC uses this initial session key (SK1). The client uses the TGT, which is cached in volatile memory space on the client machine, to request sessions with services throughout the network.

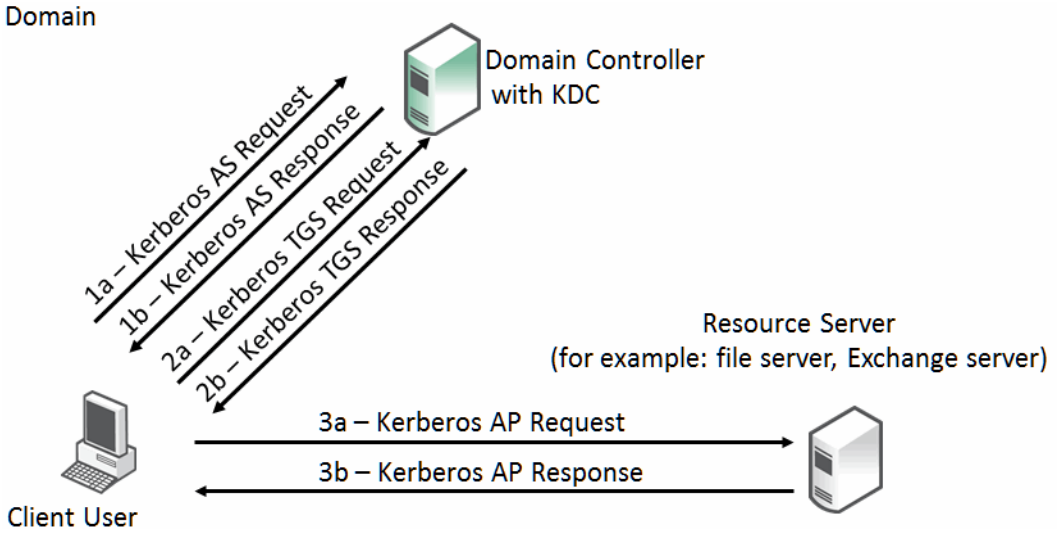
The client sends a request to the TGS and asks for credentials of the desired server. The client first creates an *authenticator* using the SK1. The authenticator contains the username, IP address, and a time stamp—all encrypted using SK1. The client next sends a copy of the TGT and authenticator to the TGS. The TGS responds with the server credentials in the form of a service ticket that also has two components: one encrypted in the server's key, which the client cannot read, and the other encrypted with the client's key, which is temporary. The encryption key is the second of the two session keys (SK2) issued during the sequence of exchanges. The SK2 is valid only for communicating with the desired server and lasts for the lifetime of the session with the desired server. A second or subsequent session with the same server needs a completely new SK2 session key requested by the client from the TGS.

Next, the client creates another authenticator, this time using SK2 for encryption. The client transmits the authenticator with the service ticket, which contains the client identity and a copy of the session key—SK2—all encrypted with the server's key, to the server. As a result of this exchange, the SK2 is now held by the client and server. SK2 authenticates the client and can optionally authenticate the server. The SK2 can also be used to encrypt further communication between the two hosts or to exchange a separate subsession key used for encrypting further communication.

The AP exchange is also known as the client-server exchange because the exchange is often completed with whatever application protocol is required between the client and server: for example, SMB and SMB2 for file sharing, MAPI for email, or HTTP for SharePoint.

The Kerberos exchange process is shown in Figure 3-4.

Figure 3-4. Kerberos exchanges



Multiple domain environments and referral tickets

The Kerberos exchange occurs when the client user, KDS, and resource server are in the same Windows domain. Where there are multiple domains, there are steps that include referral tickets.

The AS and TGS functions are separate within the key distribution center. This enables the user to use the TGT obtained from the AS in the domain to obtain service tickets from a TGS in other domains. This is accomplished through referral tickets. A referral ticket is similar to the service ticket that is granted during the TGS exchange described earlier. The difference between the two is that the referral ticket contains a ticket for the KDC in the adjoining domain rather than the server.

If there is a trust established between two domains, referral tickets are granted to clients that request authorization for services in other domains. Because of the trust relationship between the two domains, an interdomain key, based on the trust password, is available for authenticating KDC functions.

For example, consider a client user in the local domain who seeks services in a foreign domain. The two domains are directly linked through a two-way trust. The basic steps are as follows:

1. The client performs a TGS exchange with its local domain KDC. The KDC recognizes a request for a session with a foreign domain server and responds by returning a referral ticket for the KDC in the foreign domain.

2. The client contacts the KDC of the foreign domain with the referral ticket. This ticket is encrypted with the interdomain key. Given that the decryption works, the TGS service for the foreign domain returns a service ticket for the server service in the foreign domain.

3. The client performs the client-server exchange with the server in the foreign domain and begins the user session with the service.

In a multiple domain configuration where the client is in one domain and the server is in a foreign domain, communication between the two can only be through an intermediate domain. Again, assume that there are transitive two-way trusts between the local domain and the intermediate domain, and the intermediate domain and the foreign domain.

The process is similar to the two-domain example. The basic steps are as follows:

1. The client performs a TGS exchange with its local domain KDC. The KDC recognizes a request for a session with a foreign domain server and responds by returning a referral ticket for the KDC in the intermediate domain.
2. The client contacts the KDC of the intermediate domain with the referral ticket. This ticket is encrypted with the interdomain key. The TGS service for the intermediate domain returns a referral ticket for the foreign domain.
3. The client contacts the KDC of the foreign domain with the intermediate domain referral ticket. The TGS service for the foreign domain returns a service ticket for the server service in the foreign domain.
4. The client performs the client-server exchange with the server in the foreign domain and begins the user session with the service.

For more details and reference material on Kerberos that are beyond the scope of this guide, go to:

- http://technet.microsoft.com/en-us/library/bb742516.aspx
- http://web.mit.edu/kerberos/
- http://www.ietf.org/rfc/rfc4120.txt

Optimization in a native Kerberos environment

From a Riverbed perspective, the optimization of secure protocols in a Microsoft Windows environment includes the following protocols:

- Signed SMB
- Signed SMB2
- Signed SMB3
- Encrypted MAPI
- Encrypted Outlook-Anywhere
- SharePoint

You must understand how RiOS performs optimization on these protocols when the authentication is end-to-end Kerberos before you proceed to more specific details. Remember the following points:

- Secure protocols use a session key to encrypt or digitally sign protocol packets. This session key is exchanged during the authentication phase between the client and server.
- In the case of Kerberos, the session key is extracted from the service ticket by the target server.
- Decrypting the service ticket requires the server machine account password, which is set up during the domain join process and might be periodically refreshed.

- The servers machine account password is only known to the target server and domain controller.
- Active Directory supports account replication between domain controllers for high availability.

Therefore, the only way for RiOS to legitimately optimize these encrypted, or digitally signed, secure protocols is to obtain the machine credentials for any servers that require optimization. RiOS uses Active Directory replication by using the same protocols and tools as the Windows domain controllers do and by employing a domain user that has replication privileges.

When the server-side SteelHead uses this replication capability, it can transparently participate in the key exchange between the client and server, maintain security, and optimize relevant traffic.

Domain user with replication privileges

Although a user with generic domain administrator privileges meets the requirements for replication, these privileges could have some security implications.

Instead, you can assign a restricted set of privileges to a user, known as a *replication user*. You can configure the replication user on a per forest basis so that the user assigned to it can retrieve machine credentials from any domain controller in any trusted domain within the forest. Remember that a forest can comprise of multiple domains with trusts between them.

The replication user begins with a generic user account created by a Windows domain administrator, which might or might not be you. After this is done, you can apply the privileges and restrictions for the account to become a replication user on the SteelHead. If you have multiple SteelHeads in the data center (for example, in a high-availability configuration), then you can add the same replication user account to the configuration for all the server-side SteelHeads. For information about how to configure the replication user, see the *Riverbed Command-Line Interface Reference Manual* and the *SteelHead User Guide*.

The replication user has enough rights to calculate the S1 and S2 Kerberos session keys for the SteelHead to properly terminate the session and perform optimization. These rights are given by granting Replicate Changes and Replicate Changes All.

After the configuration is complete, the SteelHead retains the replication user account details by encrypting them in its secure vault.

Note: In RiOS 7.0 and later, the secure vault is locked on a new SteelHead. The secure vault is also locked on a SteelHead that is upgraded to RiOS 7.0 or later. You must unlock the secure vault to view, add, remove, or edit any replication or delegate user configuration details that are stored on the SteelHeads.

In Windows 2008 domains, you can apply further restrictions to the replication user by using a password replication policy (PRP). A PRP is a set of rules that describe a list of accounts that you can replicate between domain controllers. When you enable PRP in a Windows domain, the SteelHead is restricted, specifically to using the replication user to replicate accounts as determined by the PRP settings. Therefore, you can use the PRP to limit the accounts known to the replication user. This configuration can create additional administrative overhead in managing the PRP; however, it might be worth the additional security.

PRP is not available in Windows domains with a functional level lower than Windows 2008. For information about how to configure PRP, see the *SteelHead User Guide*.

Configuring traffic optimization for HTTP (SharePoint), encrypted MAPI, and signed SMB, SMB2, and SMB3

In RiOS 7.0 and later, you can enable Kerberos authentication support for HTTP for SharePoint, encrypted MAPI, signed SMB, and signed SMB2. To fully optimize signed SMB3 traffic, you must use RiOS 8.5 or later. When you enable the Kerberos setting on the server-side SteelHead, you enable the SteelHead to use the relevant domain user privileges and to access to the session key used by the client and server.

You must have RiOS 7.0 or later on both the server-side and client-side SteelHeads. You must use RiOS 8.5 or later for SMB3.

For information about how to configure each application, see the *SteelHead User Guide* and "HTTP optimization for SharePoint" on page 95.

Configuring the server-side SteelHead for Active Directory integrated (Windows 2003/2008)

Note: Depending on the authentication mechanism used by the Windows clients and servers, an alternative to creating and using a specific replication user can be used. For more information, see "Domain user with replication privileges" on page 60.

Regardless of the client-server authentication type, the server-side SteelHead must join the Windows domain. But depending on the authentication type, you may not need to use the replication user to optimize secure Windows traffic. If you are using NTLM authentication, so long as the server-side SteelHead join type is Active Directory integrated Windows 2003 or, preferably, Active Directory integrated Windows 2008, the SteelHead can use NTLM Transparent Mode to ensure the secure Windows traffic is fully optimized.

For the server-side SteelHead to integrate into Active Directory, you must configure the role when the appliance joins the Windows domain. Irrespective of the domain functional level (2003, mixed, 2008, and so on), select the SteelHead join type based on the operating system version running on the domain controllers. For example, if you only have domain controllers running 2003 or 2003-R2, then select the join type of Active Directory integrated Windows 2003. If you have domain controllers with a mixture of 2003, 2003-R2, 2008, or later, then select Active Directory integrated Windows 2008 as the join type.

For information about how to configure Active Directory integration, see the *Riverbed Command-Line Interface Reference Manual* and the *SteelHead User Guide*.

Be aware, that when you integrate the server-side SteelHead in this way, it does not provide any Windows domain controller functionality to any other machines in the domain and does not advertise itself as a domain controller or register any SRV records (service records). In addition, the SteelHead does not participate in an active replication nor does it write any Active Directory objects to persistent storage. The server-side SteelHead has just enough privileges so that it can read necessary attributes of servers you want to optimize and then use transparent interception.

This scenario is successful only for servers and clients that can make use of NTLM authentication. If you configure any clients and servers exclusively to use Kerberos authentication, they cannot use NTLM authentication. Therefore, the only way to optimize secure Windows traffic between such hosts is to configure the server-side SteelHead for end-to-end Kerberos support with a replication user account. However, in the event of a Kerberos authentication failure, unless explicitly configured, the session falls back to NTLM. Also, any service requested by IP address or a name that cannot be validated by Active Directory falls back to NTLM authentication.

For information about replication users, see "Domain user with replication privileges" on page 60.

The following table shows the different combinations of Windows clients and authentication methods with the earliest required version of RiOS and Windows configuration (delegation, Kerberos, Active Directory integrated) for the server-side SteelHead.

Client OS	Authentication method	RiOS 7.0 Active Directory integrated mode	RiOS 7.0 Kerberos	RiOS 9.x Active Directory integrated mode
Windows 7	Negotiate authentication/ SPNEGO	Optimized using NTLM	Optimized using Kerberos	Optimized transparent
Any client up to Windows 7	Kerberos	Optimized	Optimized	Optimized
Windows 8 and 8.1	NTLM	Optimized	Optimized	Optimized
Windows 8 and 8.1	Kerberos	Optimized (fallback)	Optimized	Optimized

For Windows 8 clients behavior, use Windows 7 information in the above table, and RiOS 8.5 or later. For Windows 8.1 and Windows 10 clients, use RiOS 9.0 or later.

Remember, if you configure the server-side SteelHead to support end-to-end Kerberos authentication, you can also join it to the domain in Active Directory integrated mode to support other clients that might be using NTLM authentication. This configuration can provide flexible and broad support for multiple combinations of Windows authentication types in use within the Active Directory environment.

For advice and suggestions on configuration best practices, see "Best practices for the SteelHead in a secure Windows deployment" on page 62.

Best practices for the SteelHead in a secure Windows deployment

There are many possible ways to configure the server-side SteelHead to support the different secure Windows application options. The following is a list of best practices to ensure that the majority, if not all, of the secure Windows traffic in your environment is fully optimized:

- Make sure that the server-side SteelHead has a DNS entry. You need only an *A record*.

- Make sure that the server-side SteelHead time-of-day is synchronized through NTP. We recommend that you synchronize to the same NTP service as the domain controllers.
- Join the server-side SteelHead to the Windows domain of choice using the role of Active Directory integrated Windows 2008. If possible, use the domain of the user domain or preferably, the same domain as the majority of the Windows application servers on which you want optimized traffic.
- Optionally, when joining the domain, consider specifying one or more domain controllers within the same domain that are geographically closest to the server-side SteelHead. If a list is not specified, the SteelHead automatically discovers domain controllers and builds its own list.
- Enable native Kerberos authentication support for the relevant Layer-7 RiOS features (for example, signed CIFS, signed SMB2, signed SMB3, encrypted MAPI, and HTTP) on the server-side SteelHead.

Note: For encrypted MAPI and/or signed SMB3 optimization, the client-side SteelHead must also have encrypted MAPI and/or SMB3 optimization enabled. You do not need to enable SMB3 signing on the client-side SteelHead.

- Configure a replication user within the Active Directory forest and enter the account details into the server-side SteelHead.
- Configure a PRP within Active Directory to further restrict the replication user account (optional).
- If the server-side SteelHead interacts with other domains through a one-way trust, use the CLI or the Management Console to enable this setting.

  If there are no one-way trusts, then this step is not required.
- Depending on the RiOS version you have on the server-side SteelHead, you can perform the Windows Active Directory configuration steps (such as domain join and user account creation) by using the domain authentication automatic configuration feature in the Management Console or through the CLI.

  For information about the domain health and domain authentication automatic configuration features, see “Domain health check and domain authentication automatic configuration” on page 65.

Domain authentication scaling

This section describes the ability of the SteelHeads to scale the authentication communications that occur between the server-side SteelHead and the Windows domain controller. This section includes the following topics:

- “When to use domain authentication scaling” on page 64
- “General improvements in RiOS 8.6” on page 64
- “Domain controller load balancing” on page 64

When to use domain authentication scaling

In releases previous to RiOS 8.6, the server-side SteelHead communicates only with a single domain controller at a time. Even if the domain that the SteelHead is part of contains additional domain controllers, the SteelHead can choose to communicate only with one. If the domain controller fails or is unavailable for some reason, then the SteelHead automatically discovers an alternative domain controller.

While this process works well in most deployments, the following are examples of environments in which an individual SteelHead can be limited by this single-instance approach and can benefit from domain authentication scaling. Domain authentication scaling is available in RiOS 8.6 and later.

A large-scale deployment (with tens of thousands of users) typically involves a larger number of authentication requests between clients and servers. Because the server-side SteelHead is involved in so many requests when optimizing secure Windows traffic, lightening its communication to an individual domain controller is important.

In another scenario, there might not be a large number of requests but there might be a high degree of latency between the server-side SteelHead and the single domain controller it is communicating with. For example, with a deployment to Microsoft Office 365, there might be *non-LAN* latency between the domain controller and the server-side SteelHead.

In general, we recommend deployment in which the latency between domain controllers and the server-side SteelHead is no more than a few milliseconds.

For more information about maximum latency between a SteelHead and a domain controller, go to http://supportkb.riverbed.com/support/index?page=content&id=S22110.

General improvements in RiOS 8.6

With RiOS 8.6 and later, the server-side SteelHead can manage higher quantities of communication with the Windows Active Directory (AD) environment it is requesting authentication details from. This has no direct benefit to the optimization of the secure Windows traffic (such as signed SMB and encrypted MAPI), but it can improve the authentication workload that the server-side SteelHead is part of. These general improvements include multithreading the RiOS code associated with AD interaction, which enables the server-side SteelHead to respond to more requests in a shorter time. Because these improvements are embedded within the RiOS code, you do not need to configure any additional settings other than to have the server-side SteelHead running RiOS 8.6 or later.

Domain controller load balancing

Windows domains often contain more than one domain controller for redundancy and scalability. With RiOS 8.6 and later, you can configure the server-side SteelHead to communicate simultaneously with multiple domain controllers in the same domain. This enables the server-side SteelHead to:

- sustain higher quantities of authentication requests.
- tolerate the loss of a domain controller much more efficiently.
- reduce the load it places on an individual domain controller.

To enable the server-side SteelHead to load balance domain controllers, enter this command on the server-side SteelHead:

```
protocol domain-auth configure load-balancing
```

You can configure this setting only on the CLI. For more information about this command, see the *Riverbed Command-Line Interface Reference Manual*.

When you enable domain controller load balancing, the server-side SteelHead balances the traffic load across up to six domain controllers within the same domain. You can configure load balancing across more domain controllers by specifying the desired number as part of the command. The following example shows how to configure the SteelHead to load balance traffic across six domain controllers:

```
protocol domain-auth configure load-balancing 6
```

You can balance traffic across a maximum of eight domain controllers. Use the **show protocol domain-auth load-balancing** command to display the current status.

You do not need to specify the individual domain controllers to which the SteelHead load balances traffic. After the SteelHead is joined to a domain it automatically discovers any domain controllers that exist by performing a DNS lookup. The results from the DNS lookup enable the server-side SteelHead to automatically build its own list of domain controllers to balance traffic across. However, if you have already configured the SteelHead with a static list of domain controllers, then the static list takes precedence over an automatically discovered list. You can create a static list either by specifying one or more domain controllers on the server-side SteelHead during the join domain procedure or by using the **domain settings dc-list *** command.

Domain health check and domain authentication automatic configuration

This section describes domain health check and domain authentication automatic configuration and includes the following topics:

- “Domain health check” on page 66
- “Domain authentication automatic configuration” on page 70

To optimize secure Windows traffic using SteelHeads, you:

- need to configure NTP-related and DNS-related settings.
- must join the server-side SteelHead to the Windows domain.
- must configure the necessary RiOS features based on the type of protocols you want optimize: for example, encrypted MAPI, signed SMB, signed SMB2, and signed SMB3.
- must deploy one or more service accounts configured for delegation (if using constrained delegation) or replication (if using end-to-end Kerberos).

The domain health check comprises of a series of tests related to Active Directory configuration settings. These tests and checks provide troubleshooting help for domain-related problems that might occur between the server-side SteelHead and the Active Directory environment.

The domain authentication automatic configuration includes a series of graphical widgets to assist you in performing the relevant configuration tasks associated with both the SteelHead and the Active Directory setup.

Domain health check

In RiOS 8.5 and later, the domain health check feature is available in the SteelHead Management Console and the CLI. Prior to RiOS 8.5, domain health check was only available in the CLI. You can use the domain health check feature to execute a variety of tests that provide diagnostic reports about the status of domain membership, end-to-end Kerberos replication, both manual and automatic constrained delegation, and DNS resolution. This information enables you to resolve issues quickly.

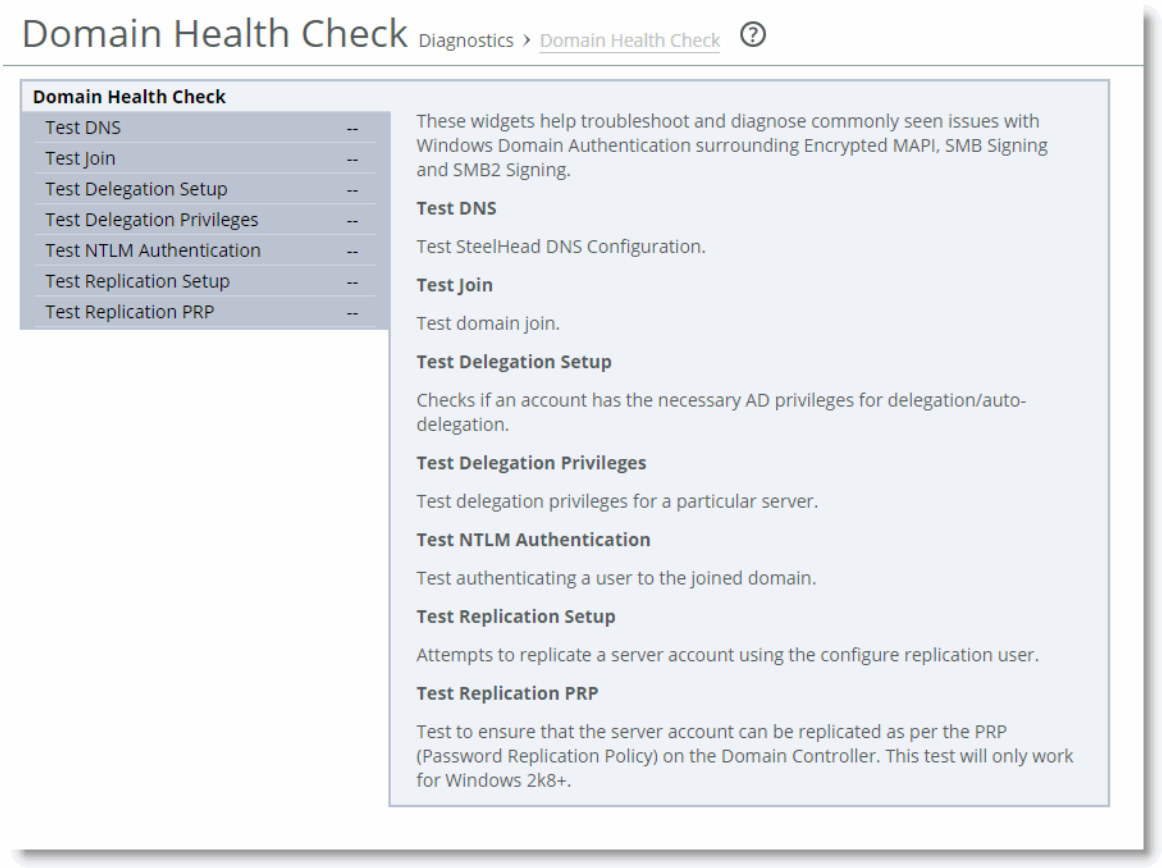
We recommend that you use domain health check from the SteelHead Management Console rather than the CLI.

For a full description of how to use domain health check, see the *SteelHead User Guide* and the *Riverbed Command-Line Interface Reference Manual*.

Using the SteelHead Management Console to test domain health check

Use the Domain Health Check page to run tests on domain health. You can create test parameters by entering specific information into certain fields. Click **Test** to run the relevant test. You receive feedback on whether the test succeeds or fails, along with the option to display a detailed log file of the test as it progresses. The output of the log file can aid in troubleshooting issues that might be found during testing.

Figure 3-5. Domain Health Check page



You can access the same tests by choosing Optimization > Active Directory: Domain Join and Optimization > Active Directory: Service Accounts pages of the SteelHead Management Console.

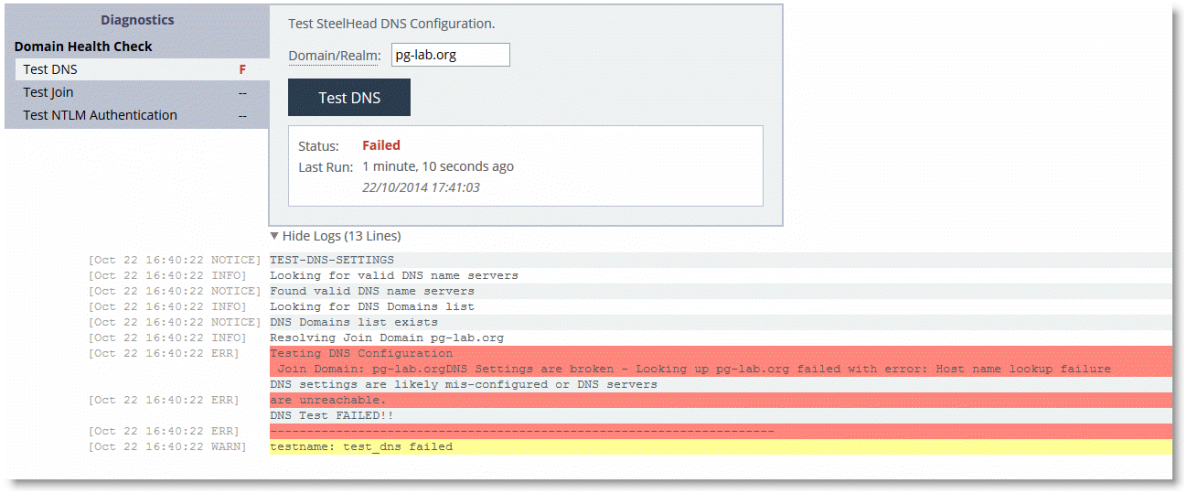
The following examples describe several ways to confirm that the domain health check feature is functioning correctly.

To test the DNS setting using the SteelHead Management Console

- From the Management Console, choose either Reports > Diagnostics: Domain Health Check or Optimization > Active Directory: Domain Join.

  Figure 3-6 shows the check for Test DNS. In this particular example, the test has failed. You can choose to display or hide the logs for the test.

Figure 3-6. Failed DNS test in the Management Console

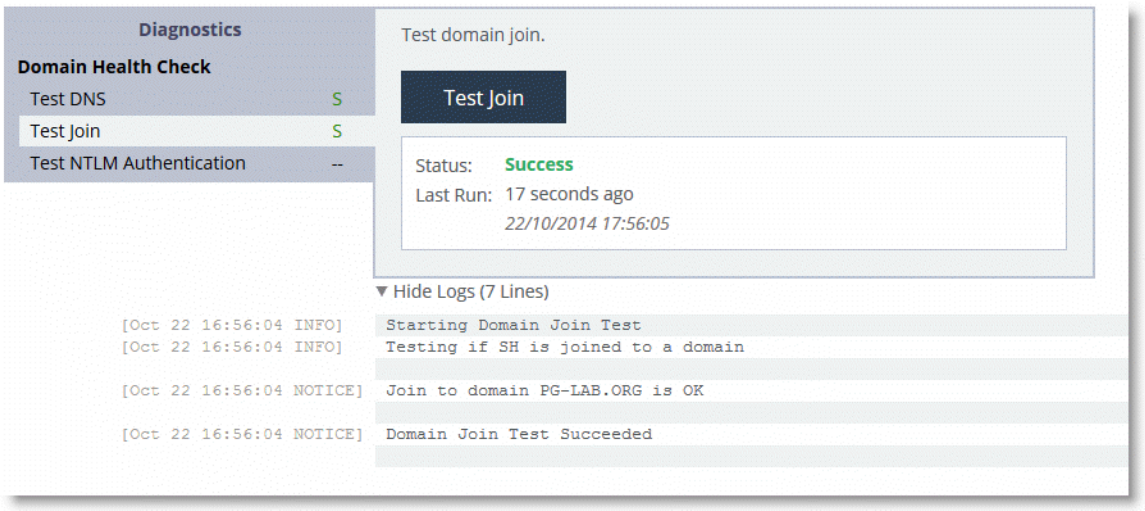


To test Domain Join using the SteelHead Management Console

- From the Management Console, choose either Reports > Diagnostics: Domain Health Check or Optimization > Active Directory: Domain Join.

Figure 3-7 shows the check for Test Join. In this particular example, the test has passed. You can choose to display or hide the logs for the test.

Figure 3-7. Successful test join in the Management Console



Using the RiOS CLI commands to test domain health check

To use the RiOS domain health check CLI commands, you must understand that each command performs a test or configuration task, but the result of the command is displayed only by executing a follow-on command. This second command is displayed as part of the output of the previous command and is usually a **show** command.

For example, the **protocol domain-auth test dns** command is followed by the **show protocol domain-auth test dns** command to display the results of the test.

The main reason for this two-stage process is that the tests themselves perform a request or look-up that is outside of the SteelHead: for example, a DNS query to a DNS server can take a few moments to complete. The two-stage process means the SteelHead CLI does not hang while waiting for the test to execute. As each test is executed, the results are saved to a temporary log file on the SteelHead. After a test is complete, the contents of the log file are displayed in a more user-friendly format when you use the relevant **show** command.

The following table lists the test or configuration tasks and the associated commands.

Task	CLI commands
Check DNS settings.	protocol domain-auth test dns show protocol domain-auth test dns
Confirm that the SteelHead is correctly joined to the Windows domain.	protocol domain-auth test join show protocol domain-auth test join
Ensure that the SteelHead can authenticate client connections.	protocol domain-auth test authentication username * password * show protocol domain-auth test authentication

Task	CLI commands
Auto-configure a previously created account in Active Directory with replication privileges over the entire domain.	protocol domain-auth auto-conf replication adminuser * adminpass * domain * dc * show protocol domain-auth auto-conf replication
Determine if end-to-end Kerberos replication is correctly configured.	protocol domain-auth test replication try-repl domain * shortdom * rserver * show protocol domain-auth test replication try-repl

The following example describes how to confirm that the domain health check feature is functioning correctly.

To check the DNS settings using the CLI

- Connect to the SteelHead CLI in and enter the following commands:

```
protocol domain-auth test dns
show protocol domain-auth test dns
```

Figure 3-8 shows a successful DNS test and Figure 3-9 shows a failed DNS test.

Figure 3-8. Successful DNS test in the CLI

```
bravo-sh93 # protocol domain-auth test dns
Test DNS Status : STARTED.
Check result using : 'show protocol domain-auth test dns'
bravo-sh93 # show protocol domain-auth test dns

Action                          STATUS                    LAST RUN
------------------------------------------------------------------------------

Test DNS                        SUCCESS              Tue Aug  9 00:17:04 2011

------------------------------------------------------------------------------


RESULT : Testing DNS Configuration
Joined Domain : VCS246.GEN-VCS78DOM.COM


DNS Test Passed
```

Figure 3-9. Failed DNS test in the CLI

```
bravo-sh93 # show protocol domain-auth test dns

Action                          STATUS                    LAST RUN
------------------------------------------------------------------------------

Test DNS                        FAILED               Tue Aug  9 00:14:37 2011

------------------------------------------------------------------------------


RESULT : Testing DNS Configuration
Joined Domain : VCS246.GEN-VCS78DOM.COM

DNS Settings are broken - Looking up VCS246.GEN-VCS78DOM.COM failed with error : Host name lookup failure
DNS settings are likely mis-configured or DNS servers are unreachable.
```

Domain authentication automatic configuration

Domain authentication automatic configuration is available in the SteelHead Management Console in RiOS 8.5 and later. Domain authentication automatic configuration is a powerful set of widgets designed to help you easily configure the server-side SteelHead and Active Directory. In RiOS versions prior to 8.5, you can configure the server-side SteelHead and Active Directory using only the CLI.

For example, Figure 3-10 shows how, in RiOS 8.5 and later, the domain authentication automatic configuration guides you through the steps to join the SteelHead to the domain and enable the relevant Windows features (encrypted MAPI, signed SMB, signed SMB2, and signed SMB3).

Figure 3-10. Domain authentication automatic configuration



You can use the domain authentication automatic configuration to configure a Windows user account that you can use for delegation or replication purposes. The domain authentication automatic configuration on the SteelHead does not create the delegate or replication user; the Windows domain administrator must create the account in advance, using the preferred standard Active Directory procedures.

After you create the basic user account in Active Directory, you can complete the remaining configuration steps using domain authentication automatic configuration on the SteelHead.

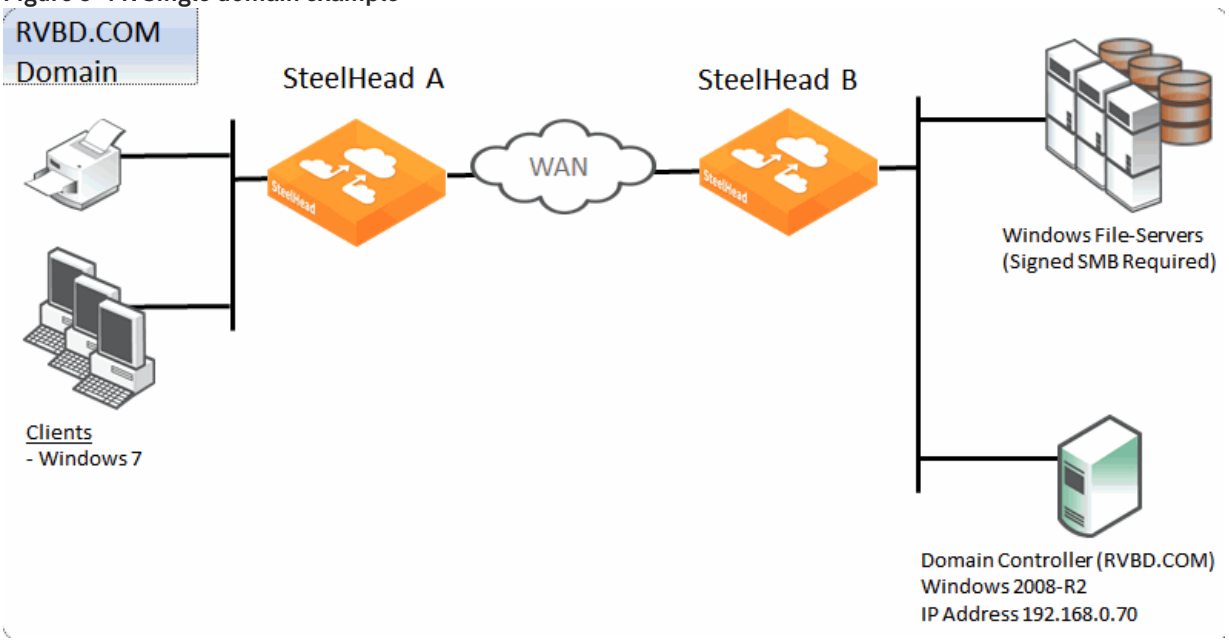
Along with configuring the delegation and replication user accounts, domain authentication automatic configuration enables you to add and remove entries in the lists of delegation servers.

For information about how to use domain authentication automatic configuration, see the *SteelHead User Guide*.

Single domain example configuration

Figure 3-11 shows a data center and a branch office that are both in the RVBD.COM domain hosted by a domain controller running Windows 2008-R2. The Windows 7 clients in the remote office want optimized CIFS access to the file servers in the Data Center. The file servers are configured with *Signing Required*. For ease of management, transparent mode is preferred by the customer.

Figure 3-11. Single domain example



The following is true for this configuration:

- We recommend that all the SteelHeads are time synchronized for SteelHead deployments that involve some form of Windows authentication. This is especially true where Kerberos is involved in the authentication process. Consider using NTP to make the time synchronization task easier to maintain.
- SteelHead A on the client side must have RiOS 7.0 or later running to support signed SMB, but no further client-side configuration is required.
- SteelHead B on the server side needs to join the RVBD.COM domain.
- SteelHead B needs to have RiOS 7.0 or later and transparent mode configured.
- Secure inner channel is optional for this configuration.

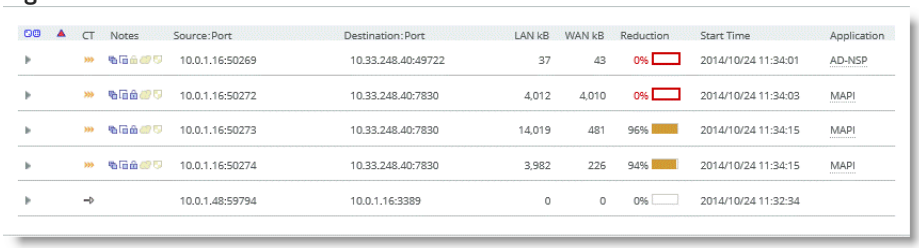
To configure a data center and branch office in the same domain

1. SteelHead A does not need configuration.
2. On SteelHead B, connect to the CLI in configuration mode and enter the following commands:

```
domain join domain-name RVBD.COM login Administrator password ******** dc-list 192.168.0.70
short-name RVBD
protocol cifs smbv1-mode enable
protocol cifs smb signing enable
protocol cifs smb signing mode-type transparent
write memory
restart
```

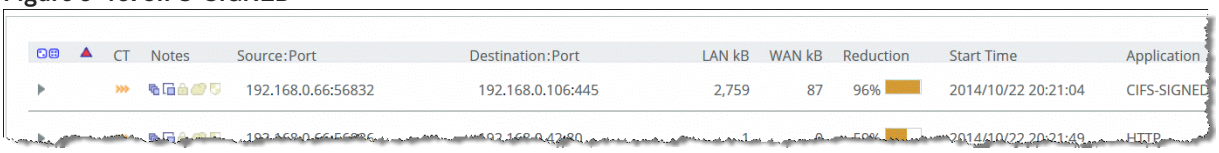
Confirm the correct configuration settings display in the Current Connections report, shown in Figure 3-12 and Figure 3-13. For information about the Current Connection Report, see the *SteelHead User Guide*.

Figure 3-12. MAPI-ENCRYPTS



CT	Notes	Source:Port	Destination:Port	LAN kB	WAN kB	Reduction	Start Time	Application
		10.0.1.16:50269	10.33.248.40:49722	37	43	0%	2014/10/24 11:34:01	AD-NSP
		10.0.1.16:50272	10.33.248.40:7830	4,012	4,010	0%	2014/10/24 11:34:03	MAPI
		10.0.1.16:50273	10.33.248.40:7830	14,019	481	96%	2014/10/24 11:34:15	MAPI
		10.0.1.16:50274	10.33.248.40:7830	3,982	226	94%	2014/10/24 11:34:15	MAPI
		10.0.1.48:59794	10.0.1.16:3389	0	0	0%	2014/10/24 11:32:34	

Figure 3-13. CIFS-SIGNED



CT	Notes	Source:Port	Destination:Port	LAN kB	WAN kB	Reduction	Start Time	Application
		192.168.0.66:56832	192.168.0.106:445	2,759	87	96%	2014/10/22 20:21:04	CIFS-SIGNED

4

HTTP Optimization

This chapter includes examples of techniques used by the Hypertext Transfer Protocol (HTTP) optimization module on the SteelHead to improve optimization. This chapter includes the following sections:

- "HTTP and browser behavior" on page 76
- "RiOS HTTP optimization techniques" on page 85
- "HTTP authentication optimization" on page 90
- "HTTP automatic configuration" on page 91
- "HTTP Settings for common applications" on page 94
- "HTTP optimization for SharePoint" on page 95
- "HTTP optimization module and internet-bound traffic" on page 96
- "HTTP and IPv6" on page 97
- "Overview of the web proxy feature" on page 97
- "Tuning Microsoft IIS server" on page 98
- "Info-level logging" on page 103
- "Use case" on page 103

The HTTP protocol has become the accepted standard mechanism for transferring documents on the internet. The original version, HTTP 1.0, is documented in RFC 1945 but has since been superseded by HTTP 1.1 and documented in RFC 2068. TCP protocol is the most common underlying transport protocol for HTTP.

With RiOS 9.8 and later, SteelHead appliances support optimization of HTTP/2 cached traffic.

By default, a web server listens for HTTP traffic on TCP port 80, although you can reconfigure the web server to listen on a different port number. HTTP uses a very simple client-server model—after the client has established the TCP connection with the server, it sends a request for information and the server replies back. A web server can support many different types of requests but the two most common types of requests are as follows:

- **GET request** - asks for information from the server.
- **POST request** - submits information to the server.

A typical web page is not one file that downloads all at once. web pages are composed of dozens of separate objects including JPG and GIF images, JavaScript code, cascading style sheets—each of which is requested and retrieved separately, one after the other. Given the presence of latency, this behavior is highly detrimental to the performance of web-based applications over the WAN. The higher the latency, the longer it takes to fetch each individual object and, ultimately, to display the entire page. Furthermore, the server might be protected and require authentication before delivering the objects. The authentication can be once per connection or it can be once per request.

The HTTP optimization module addresses these challenges by using several techniques. For example:

- The HTTP optimization module learns about the objects within a web page and prefetch those objects in bulk before the client requests them. When the client requests those objects, the local SteelHead serves them out locally without creating extra round trips across the WAN.
- The HTTP optimization module learns the authentication scheme that is configured on the server. It can inform the client that it needs to authenticate against the server without the client incurring an extra round trip to discover the authentication scheme on the server.

HTTP and browser behavior

The HTTP protocol uses a simple client-server model where the client transmits a message (also referred to as a *request*), to the server and the server replies with a response. Apart from the actual request, which is to download the information from the server, the request can contain additional embedded information for the server. For example, the request might contain a *Referer* header. The Referer header indicates the URL of the page that contains the hyperlink to the currently requested object. Having this information is very useful for the web server to trace where the requests originated from, for example:

```
GET /wiki/List_of_HTTP_headers HTTP/1.1
Host: en.wikipedia.org
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.7)
Gecko/20100713 Firefox/3.6.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

SOURCE: http://en.wikipedia.org/wiki/HTTP_referer

The HTTP protocol is a stateless protocol. A stateless protocol means that the HTTP protocol does not have a mechanism to store the state of the browser at any given moment in time. Being able to store the state of the browser is useful particularly in situations where a user who is online shopping has added items to the shopping cart but was then interrupted and closed the browser. With a stateless protocol, when the user revisits the website, the user has to add all the items back into the shopping cart. To address this issue, the server can issue a *cookie* to the browser. These cookies are the typical types:

- **Persistent cookie** - remains on the user's PC after the user closes the browser.
- **Nonpersistent cookie** - discarded after the user closes the browser.

In the example earlier, assuming that the server issues a persistent cookie, when the user closes the browser and revisits the website, all the items from the previous visit are populated in the shopping cart. Cookies are also used to store user preferences. Each time the user visits the website, the website is personalized to the user's liking.

This example shows where to find the cookie in the client's GET request.

```
GET /cx/scripts/externalJavaScripts.js HTTP/1.1
Host: www.acme.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.6)
Gecko/20100625 Firefox/3.6.6
Accept: */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://www.acme.com/cpa/en_INTL/homepage
Cookie: TRACKER_CX=64.211.145.174.103141277175924680;
CoreID6=94603422552512771759299&ci=90291278,90298215,90295525,90296369,902941
55; _csoot=1277222719765; _csuid=4c20df4039584fb9; country_lang_cookie=SG-en
```

Web browser performance has improved considerably over the years. In the HTTP 1.0 specification, the client opened a connection to the server, downloaded the object, and then closed the connection. Although this was easy to understand and implement, it was very inefficient because each download triggered a separate TCP connection setup (TCP 3-way handshake). In the HTTP 1.1 specification, the browser has the ability to use the same connection to download multiple objects.

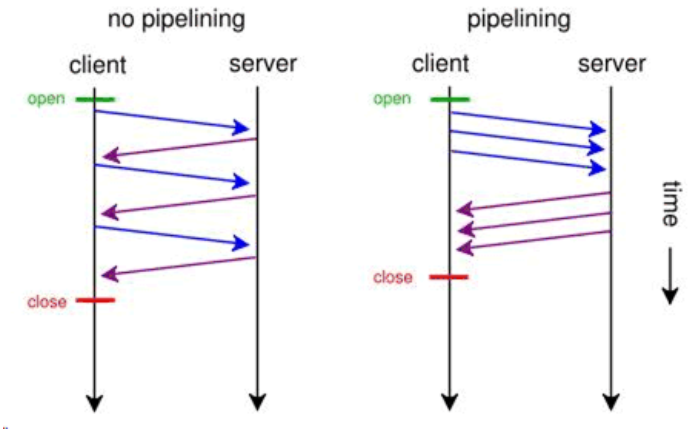
The next section discusses how the browser uses multiple connections and a technique known as *pipelining* to improve the browser's performance.

Multiple TCP connections and pipelining

Most modern browsers establish two or more TCP connections to the server for parallel downloads. The concept is simple—as the browser parses the web page, it knows what objects it needs to download. Instead of sending the requests serially over a single connection, the requests are sent over parallel connections resulting in a faster download of the web page.

Another technique used by browsers to improve the performance is *pipelining*. Without pipelining, the client first sends a request to the server and waits for a reply before sending the next request. If there are two parallel connections, then a maximum of two requests are sent to the server concurrently. With pipelining, the browser sends the requests in batches instead of waiting for the server to respond to each individual object before sending the next request. Pipelining is used with a single HTTP connection or with multiple HTTP connections. Although most servers support multiple HTTP connections, some servers do not support pipelining. Figure 4-1 illustrates the effect of pipelining.

Figure 4-1. Pipelining



HTTP authentication

To provide access control to the contents on the web server, authentication is enabled on the server. There are many ways to perform authentication—for example, certificates and smartcards. The most common schemes are NT LAN Manager (NTLM) and Kerberos. Although the two authentication schemes are not specific to HTTP and are used by other protocols such as CIFS and MAPI, there are certain aspects that are specific to HTTP.

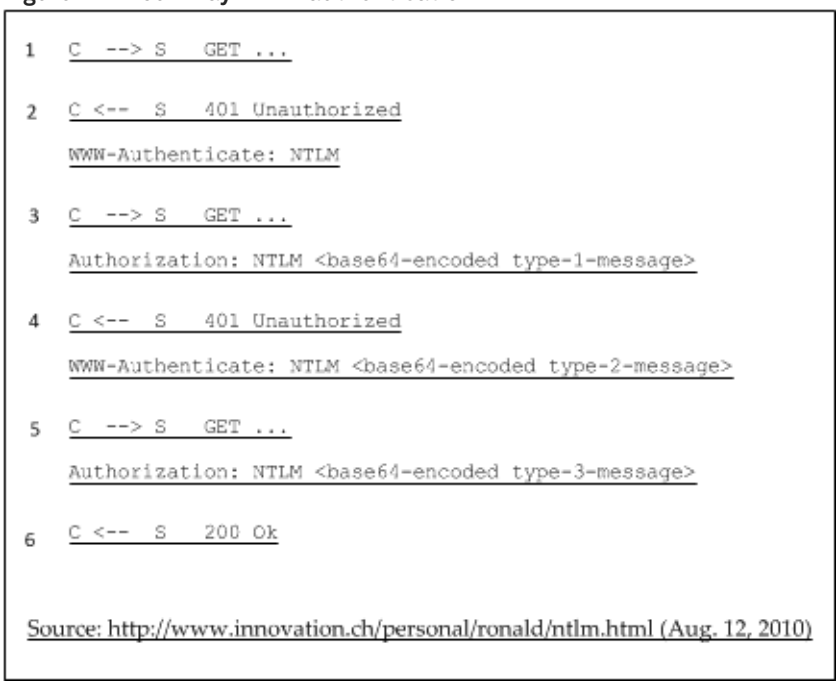
Note: For more information about Kerberos, see "CIFS Optimization" on page 13, "Microsoft Exchange Email Optimization" on page 21, and "Kerberos" on page 56.

The following steps describe four-way NTLM authentication shown in Figure 4-2:

1. The client sends a GET request to the server.
2. The server, configured with NTLM authentication, sends back a *401 Unauthorized* message to inform the client that it needs to authenticate. Embedded in the response is the authentication scheme supported by the server. This is indicated by the WWW-Authenticate line.
3. The client sends another GET request and attaches an NTLM Type-1 message to the request. The NTLM Type-1 message provides the set the capability flags of the client (for example, encryption key size).
4. The server responds with another 401 Unauthorized message, but this time it includes an NTLM Type-2 message in the response. The NTLM Type-2 message contains the server's NTLM challenge.
5. The client computes the response to the challenge and once again attaches this to another GET request. Assuming the server accepts the response, the server delivers the object to the client.

6. Assuming a network on 200 ms of round-trip latency, it would take at least 600 ms for the browser begins to download the object.

Figure 4-2. Four-way NTLM authentication



For a more in-depth discussion on NTLM authentication, go to:

- http://www.innovation.ch/personal/ronald/ntlm.html

A detailed explanation of the Kerberos protocol is beyond the scope of this guide. For details, go to:

- http://simple.wikipedia.org/wiki/Kerberos
- http://technet.microsoft.com/en-us/library/cc772815(WS.10).aspx

There is the concept of per-request and per-connection authentication with HTTP authentication. A server configured with per-request authentication requires the client to authenticate every single request before the server delivers the object to the client. If there are 100 objects (for example, .jpg images), it performs authentication 100 times with per-request authentication. With per-connection authentication, if the client only opens a single connection to the server, then the client only needs to authenticate with the server once. No further authentication is necessary. Using the same example, only a single authentication is required for the 100 objects. Whether the web server does per-request or per-connection authentication varies depending on the software. For Microsoft's Internet Information Services (IIS) server, the default is per-connection authentication when using NTLM authentication, the default is per-request authentication when using Kerberos authentication.

When the browser first connects to the server, it does not know whether the server has authentication enabled. If the server requires authentication, the server responds with a 401 Unauthorized message. Within the body of the message, the server indicates what kind of authentication scheme it supports in the WWW-Authenticate line. If the server supports more than one authentication schemes, then there are multiple WWW-Authenticate lines in the body of the message. For example, if a server supports both Kerberos and NTLM, the following appears in the message body:

```
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM
```

When the browser receives the Negotiate keyword in the WWW-Authenticate line, it first tries Kerberos authentication. If Kerberos authentication fails, it falls back to NTLM authentication.

Assuming that there are multiple TCP connections in-use, after the authentication succeeds on a first connection, the browser downloads and parses the base page as they authenticate. To improve performance, most browsers parse the web page and start to download the objects over parallel connections (for details, see "Multiple TCP connections and pipelining" on page 77).

Connection jumping

When the browser establishes the second, or subsequent, TCP connections for the parallel or pipelining downloads, it does not remember if the server requires authentication. Therefore, the browser sends multiple GET requests over the second, or additional, TCP connections, without the authentication header. The server rejects the requests with the 401 Unauthorized messages. When the browser receives the 401 Unauthorized message on the second connection, it is aware that the server requires authentication. The browser initiates the authentication process. Yet, instead of keeping the authentication requests for all the previously requested objects on the second connection, some of the requests jump over the first and are authenticated. This is known as *connection jumping*.

Note: Connection jumping is specific behavior to Internet Explorer, unless you are using Internet Explorer 8 with Windows 7.

Figure 4-3 shows the client authenticated with the server and the client request for the index.html web page. It parses the web page and initiates a second connection for parallel download.

Figure 4-3. Client authenticates with the server

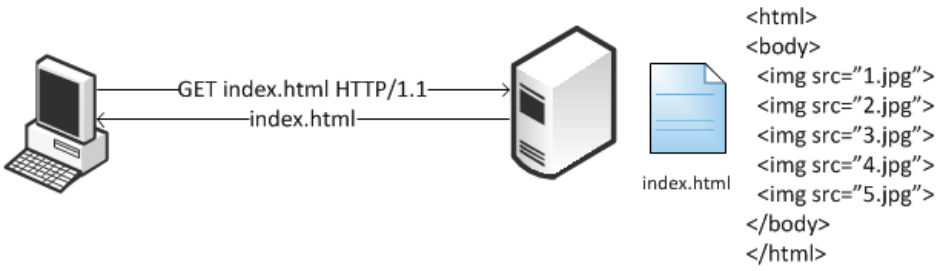


Figure 4-4 shows the client pipelines the requests to download objects 1.jpg, 2.jpg, and 3.jpg on the second connection. However, because the server requires authentication, the requests for those objects are all rejected.

Figure 4-4. Rejected second connection

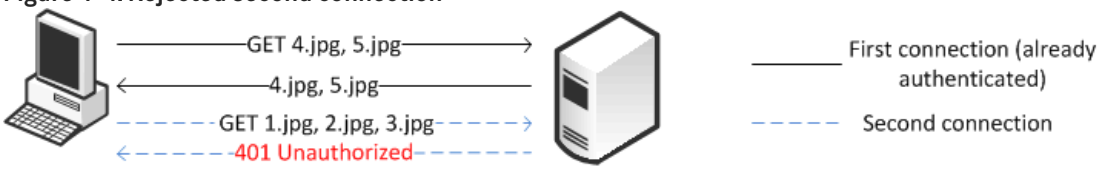
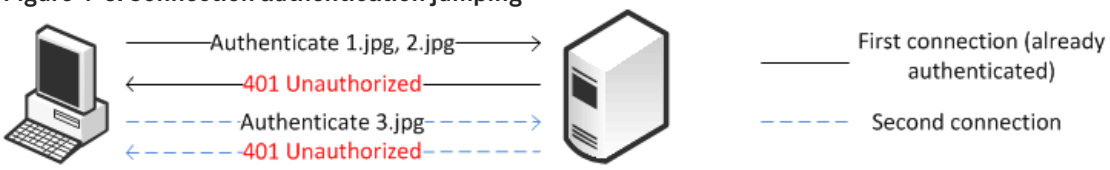


Figure 4-5 shows the client sends the authentication request to the server. Instead of keeping the requests on the second connection, some of the requests have jumped over to the first, and an already authenticated, connection.

Figure 4-5. Connection authentication jumping



The following problems, that result in performance impact, arise as a result of this connection jumping behavior:

- If an authentication request appears on an already authenticated connection, the server can reset the state of the connection and force it to go through the entire authentication process again.
- The browser has effectively turned this into a per-request authentication even though the server can support per-connection authentication.

Note: For more details, see Strip auth header in “HTTP authentication optimization” on page 90.

HTTP proxy servers

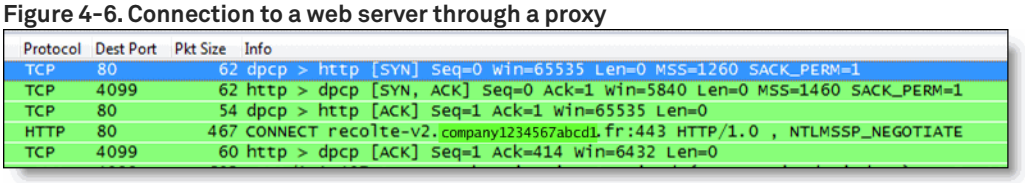
Most enterprises have proxy servers deployed in the network. Proxy servers serve the following purposes:

- Access control
- Performance improvement

Many companies have some compliance policies that restrict what the users can or cannot access from their corporate network. Enterprises meet this requirement by deploying proxy servers. The proxy servers act as a single point through which all web traffic entering or exiting the network must traverse so the administrator can enforce the necessary policies. In addition to controlling access, the proxy server might act as a cache engine caching frequently accessed data. By doing this, it can eliminate the need to fetch the same content for different users.

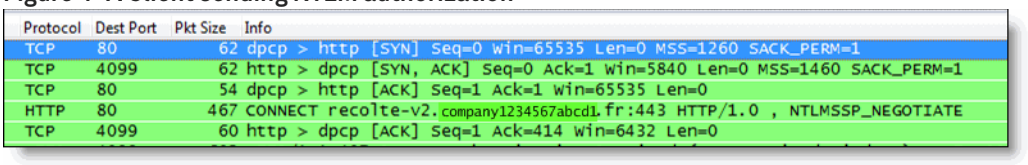
Communication with an HTTP proxy server differs from the browser to the requested HTTP server. When you use a defined proxy server, the browser initiates a TCP connection with that defined proxy server. The initiation starts with a standard TCP three-way handshake. Next, the browser requests a web page and issues a CONNECT statement to the proxy server, with instructions to which web server it wants to connect.

Figure 4-6. Connection to a web server through a proxy



In a standard HTTP request with an open proxy server, the proxy next opens a connection to the requested web server and returns the requested objects. However, most corporate environments use proxy servers as outbound firewall devices and require authentication by issuing a code 407 Proxy Authentication Required. After a successful authorization is complete, the proxy returns the originally requested objects. Successful authorization by the proxy server can be verifying username and password, and if the destination website is on the approved list.

Figure 4-7. Client sending NTLM authorization



SSL connections are different. In a standard HTTP request, the SteelHead optimizes proxy traffic without any issues. Standard HTTP optimized traffic is leveraged, and data reduction and latency reductions are prevalent. Due to the nature of proxy server connections, an additional step is required to set up SSL connections. Prior to RiOS 7.0, SSL connections were set up as pass-through connections and were not optimized because the SSL sessions were negotiated after the initial TCP setup. In RiOS 7.0 and later, RiOS can negotiate SSL after the setup of the session.

Figure 4-8. Setting up an SSL connection through a proxy server

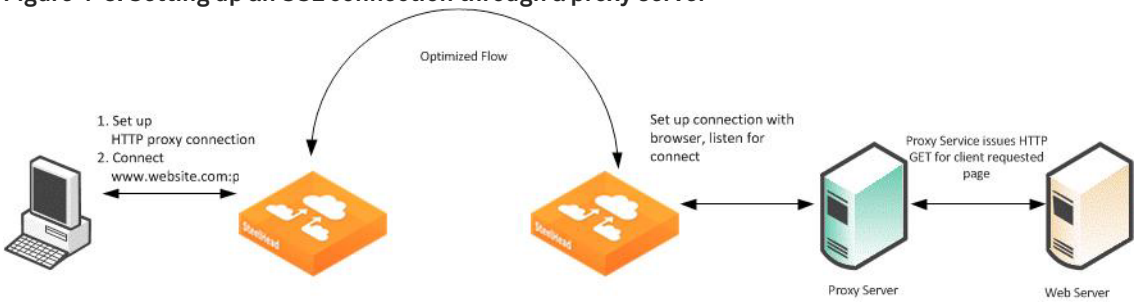


Figure 4-8 shows an SSL connection through a proxy server. The steps are as follows:

1. A client sends a TCP three-way handshake to the proxy server. The proxy HTTP is made on this connection.
2. The client issues a CONNECT statement to the host it wants to connect with, for example:

```
CONNECT www.riverbed.com:443 HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;.NET CLR 1.0.3705;.NET CLR
1.1.4322;.NET CLR 2.0.50727;.NET CLR 3.0.4506.2152;.NET CLR 3.5.30729)
Host: www.riverbed.com
Content-Length: 0
Proxy-Connection: Keep-Alive
Pragma: no-cache
```

3. The proxy server forwards the connect request to the remote web server.
4. The remote HTTP server accepts and send back an ACK.
5. The client sends an SSL Client Hello Message.

6. The server-side SteelHead intercepts the Client Hello Message and begins to set up an SSL inner-channel connection with the client. The SteelHead also begins to set up the SSL conversation with the original web server.

Note: The private key and certificate of the requested HTTP server must exist on the server-side SteelHead, along with all servers targeted for SSL optimization.

7. An entry is added to a hosts table on the server-side SteelHead.

   The host table is how a SteelHead discerns which key is associated with each SSL connection, because each SSL session is to the same IP and port pair. The hostname becomes the key field for managing this connection.

8. The server-side SteelHead passes the session to the client-side SteelHead using SSL optimization. For information about SSL optimization, see "SSL Deployments" on page 169.

Configuring HTTP SSL proxy interception

In some HTTP proxy implementations, the proxy server terminates the SSL session to the web server to inspect the web payload for policy enforcement or surveillance. In this scenario, the SSL server that is defined in the SSL Main Settings page on the server-side SteelHead is the SSL proxy device, and not the server being requested by the browser. If you use self-signed certificates from the proxy server, you must add the CA from the proxy server to the SteelHead trusted certificate authority (CA) list.

Note: For more information about SSL, see "Configuring SSL optimization on SteelHeads" on page 174.

Figure 4-9. SSL Session Setup between the Proxy Server and the client

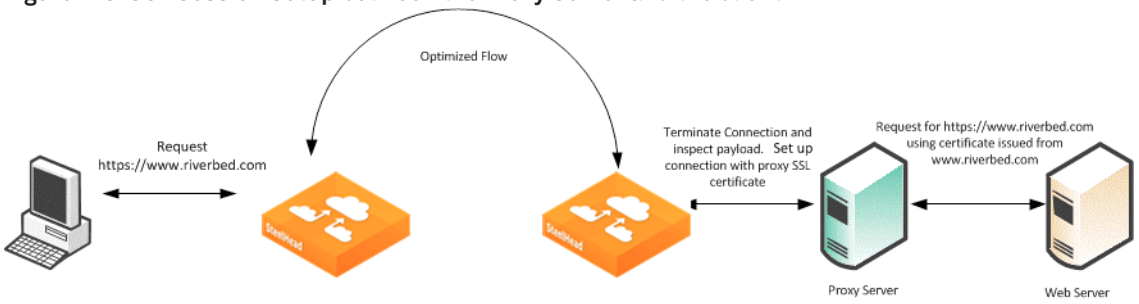
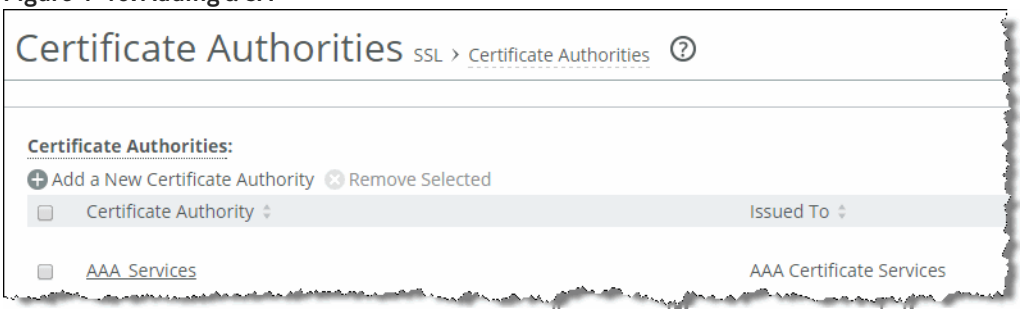


Figure 4-9 shows that the client requests a page from http://www.riverbed.com/. This request is sent to the proxy server as a CONNECT statement. Because this is an intercepted proxy device, the SSL session is set up between the proxy server and the client, typically with an internally trusted certificate. You must use this certificate on the server-side SteelHead for SSL optimization to function correctly.

Figure 4-10. Adding a CA



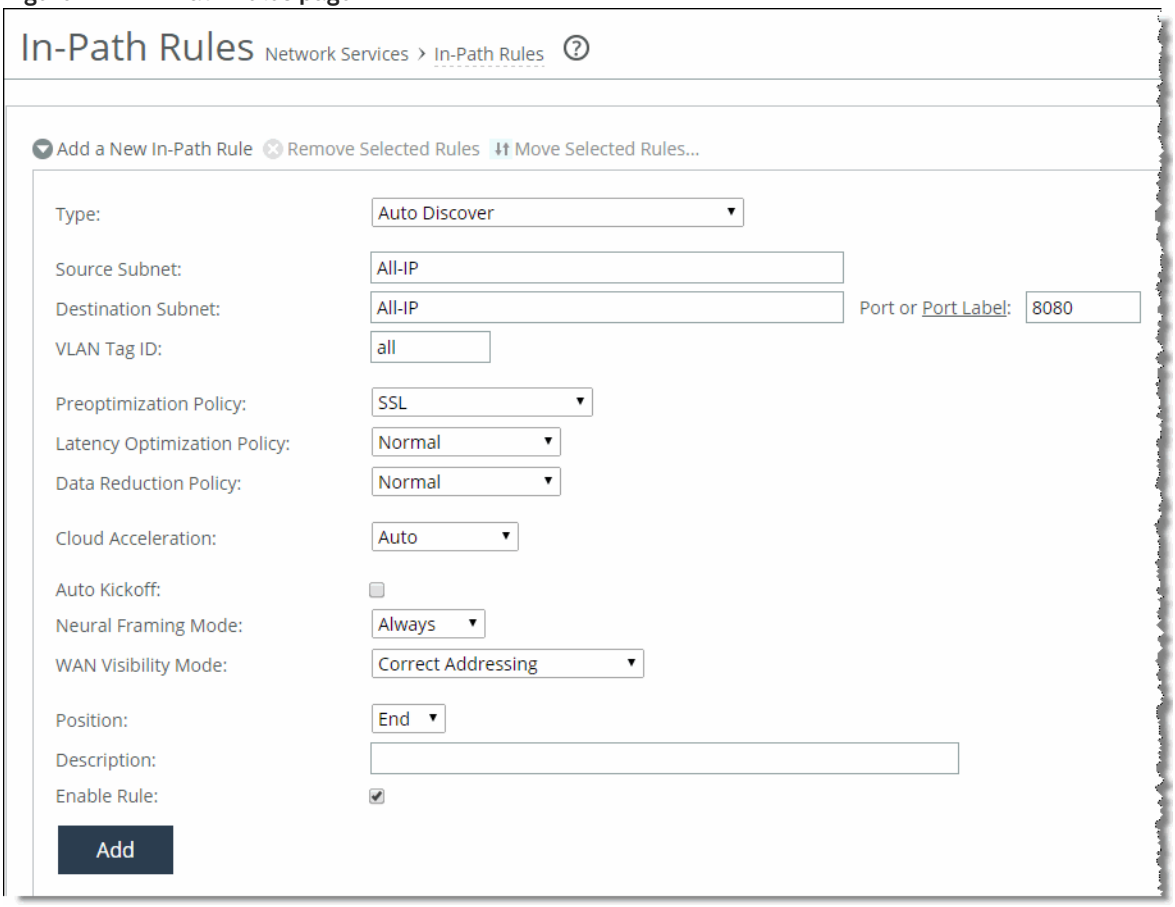
To configure RiOS SSL traffic interception

1. Before you begin:
   - You must have a valid SSL license.
   - The server-side and client-side SteelHeads must be running RiOS 7.0 or later.
2. Choose Optimization > SSL: SSL Main Settings and select Enable SSL Optimization.

   The private key and the certificate of the requested web server, and all the servers targeted for SSL optimization, must exist on the server-side SteelHead.
3. Choose Optimization > SSL: Advanced Settings and select Enable SSL Proxy Support.

4. Create an in-path rule on the client-side SteelHead (shown in Figure 4-11) to identify the proxy server IP address and port number with the SSL preoptimization policy for the rule.

Figure 4-11. In-Path Rules page



RiOS HTTP optimization techniques

The HTTP optimization module was first introduced in RiOS 4.x. At the time, the only feature available was URL learning. Subsequently, in RiOS 5.x, the parse-and-prefetch and metadata response features were added to the family. In RiOS 6.0, the metadata response feature was replaced with the object prefetch table. In RiOS 6.1, major enhancements were made specifically to optimize HTTP authentication.

RiOS 7.0 introduces HTTP automatic configuration. With HTTP automatic configuration you can:

- enable HTTP automatic configuration to profile HTTP applications. In other words, HTTP automatic configuration collects statistics on applications and dynamically generates a configuration entry with the proper optimization settings and self-tunes its HTTP parameter settings.

  You can also store all objects permitted by the servers—you do not need to specify the extension type of specific objects to prefetch.

- expand the HTTP authentication options by including end-to-end Kerberos authentication. All previous configurations are still available for configuration.

You must run RiOS 7.0 or later on both the client-side SteelHead and server-side SteelHead to use automatic HTTP configuration and end-to-end Kerberos. For more details, see “HTTP automatic configuration” on page 91 and “HTTP authentication optimization” on page 90.

Primary content optimization methods

The following list explains the primary content optimization methods:

- **Strip compression** - To conserve bandwidth, nearly all browsers support compression. The browser specifies what encoding schemes it supports in the *Accept-Encoding* line in the request. Before the server responds with a reply, it compresses the data with the encoding scheme that the client supports. To maximize the benefit of SDR, data coming from the server must decompress to allow SDR to de-duplicate this data. When Strip Compression is enabled, the HTTP optimization module removes the Accept-Encoding line from the request header before sending the request to the server. Because the modified request does not contain any supported compression scheme, the server responds to the request without any compression. This allows SDR to deduplicate the data. With this option enabled, the amount of LAN-side traffic increases as the server no longer sends the traffic in a compressed format.
- **Insert cookie** - The HTTP optimization module relies on cookies to distinguish between different users. If the server does not support cookies, the HTTP optimization module inserts its own cookie so that it can distinguish between different users. Cookies that are inserted by the HTTP optimization module start with *rbt-http=* and are followed by a random number. Enable this option only if the server does not issue its own cookies.

  The SteelHead removes the Riverbed cookie when it forwards the request to the server.
- **Insert keep-alive** - Keep-alive, or persistent connection, is required for the HTTP optimization module to perform prefetches. If the client does not support keep-alive and the server does, the client-side SteelHead inserts a Connection: keep-alive to the HTTP/1.0 response, unless the server explicitly instructed to close the connection by adding Connection: close.

  This option does not apply to situations where the client supports keep-alive but the server does not. If the server does not support keep-alive, then prefetching is not possible and changes must be made on the server to support keep-alive.
- **URL learning** - The SteelHead learns associations between a base request and a follow-on request. This feature is most effective for web applications with large amounts of static-content images and style sheets for example. Instead of saving each object transaction, the SteelHead saves only the request URL of object transactions in a knowledge base, and then it generates related transactions from the list. URL learning uses the Referer header field to generate relationships between object requests and the base HTML page that referenced them and to group embedded objects. This information is stored in an internal HTTP database. The benefit of URL learning is faster page downloads for subsequent references to the same page, from the original browser of the requester or from other clients in the same location. When the SteelHead finds a URL in its database, it immediately sends requests for all of the objects referenced by that URL, saving round trips for the client browser. You can think of URL learning as an aggressive form of prefetching that benefits all users in a location, instead of just a single user, and that remembers what was fetched for subsequent accesses.

- **Parse-and-prefetch** - The SteelHead includes a specialized algorithm that determines which objects are going to be requested for a given web page, and it prefetches them so that they are available when the client makes a request. This feature complements URL learning by handling dynamically generated pages and URLs that include state information.

  Parse-and-prefetch reads a page, finds HTML tags that it recognizes as containing a prefetchable object, and sends out prefetch requests for those objects. Typically, a client needs to request the base page, parse it, and then send out requests for each of these objects. This still occurs, but with parse-and-prefetch the SteelHead has quietly prefetched the page before the client receives it and has already sent out the requests. This allows the SteelHead to serve the objects as soon as the client requests them, rather than forcing the client to wait on a slow WAN link.

  For example, when an HTML page contains the tag <img src="my_picture.gif">, the SteelHead prefetches the image my_picture.gif

  Like URL learning, parse-and-prefetch benefits many users in a location rather than just a single user. Unlike URL learning, parse-and-prefetch does not remember which objects were referenced in a base request, so it does not waste space on dynamic content that changes each request for the same URL, or on dynamic URLs, which is not an efficient use of space.

- **Object prefetch table** - The SteelHead stores objects per the metadata information contained in the HTTP response. The object prefetch table option helps the client-side SteelHead respond to If-Modified-Since (IMS) requests from the client, cutting back on round trips across the WAN. This feature is useful for applications that use a lot of cacheable content.

  Although URL learning and parse-and-prefetch can request information from the HTTP server much sooner than the client would have requested the same information, the object prefetch table stores information to completely eliminate some requests on the WAN. The client browsers receive these objects much faster than they would if the object had to be fetched from the server.

  In releases previous to RiOS 8.6, all HTTP responses above 1 Mb triggered the RiOS HTTP traffic optimization to bypass the remainder of established HTTP sessions. In RiOS 8.6 and later, environments running both client-side and server-side SteelHeads remove the 1-Mb response limit. However, single HTTP objects returned as part of the response, which are greater than 1 Mb, are not stored in the HTTP object prefetch table by default.

Although features such as URL learning, parse-and-prefetch, and the object prefetch table (the replacement to metadata response) can help speed up applications, other factors, such as authentication, can negatively impact the performance of applications.

HTTP vary headers

RFC 2616 provides HTTP 1.1 the ability for client and server to determine if they can retrieve HTTP responses from local cache or if they must retrieve the responses from the origin server. The vary header consists of HTTP field names that the client evaluates. In releases previous to RiOS 8.6, responses that contained an HTTP vary header were excluded from SteelHead HTTP optimization.

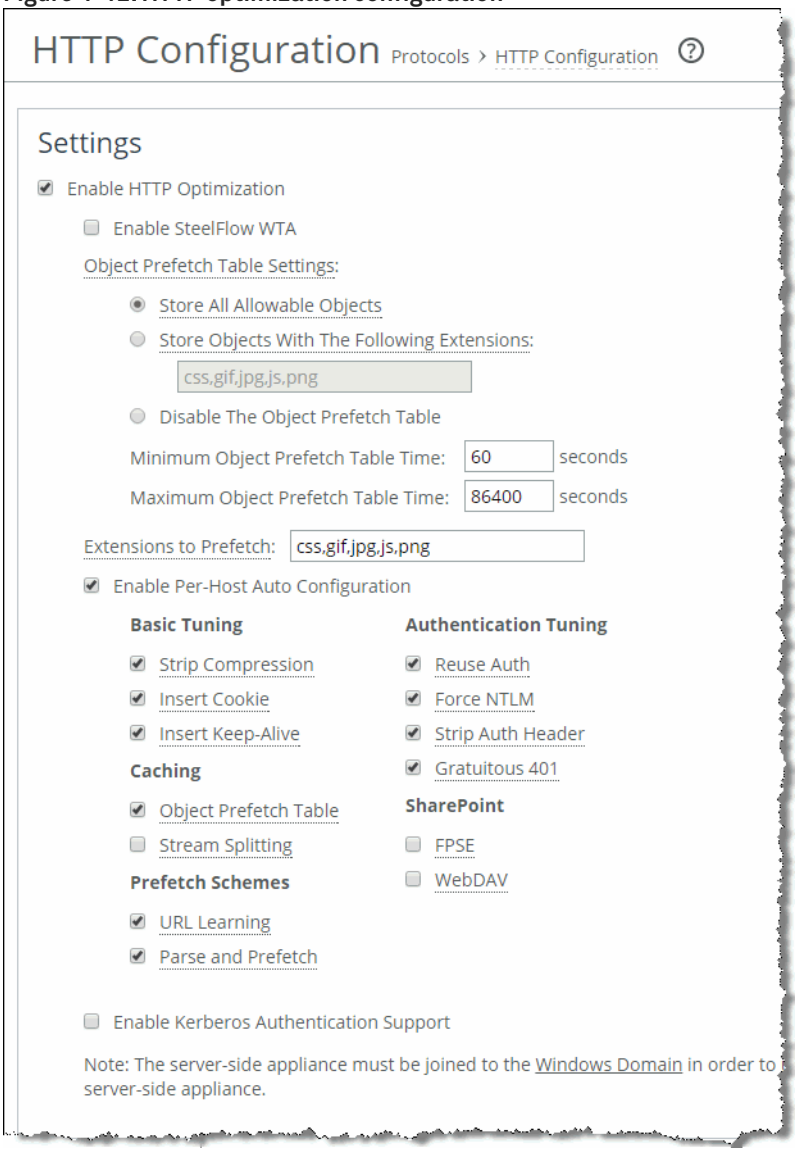
In RiOS 8.6 and later, if HTTP responses contain a vary HTTP header and are not compressed, RiOS HTTP optimization caches the response for client-side consumption. However, if RiOS receives a response with a vary HTTP header and the response is content-encoded, the local RiOS HTTP object cache does not store the HTTP object.

The following example shows an HTTP response with an HTTP vary header set without content encoding enabled:

```
HTTP/1.1 200 OK
Date: Sat, 01 Mar 2014 23:07:28 GMT
Server: Apache
Last-Modified: Fri, 28 Feb 2014 10:23:10 GMT
ETag: "183-4f374d3157f80-gzip"
Accept-Ranges: bytes
Vary: User-Agent
Content-Encoding: gzip
Content-Length: 265
Keep-Alive: timeout=15, max=98
Connection: Keep-Alive
Content-Type: text/css
```

As a best practice, enable the HTTP Strip Compression option on the HTTP page. You can apply this as part of the HTTP automatic configuration settings or manually per HTTP server. For more information about HTTP automatic configuration, see “HTTP automatic configuration” on page 91.

Figure 4-12. HTTP optimization configuration



Connection pooling

Connection pooling preestablishes 20 inner channel connections between each pair of SteelHeads. HTTP traffic benefits the most from connection pooling, although connection pooling is not specific to HTTP. When the SteelHead requires an inner channel, it picks one from the pool and therefore eliminates the time for the TCP three-way handshake. The reason HTTP traffic benefits the most is because those connections are typically short-lived.

Note: Connection pooling is available only when using the Correct Addressing as the WAN visibility mode. For details, see the *SteelHead Deployment Guide*.

HTTP authentication optimization

RiOS 6.1 and later has specific HTTP authentication optimization for handling the various inefficient browser authentication behaviors. The HTTP authentication optimization attempts to modify the client-to-server behavior so that it maximizes the benefit of the HTTP optimization module.

RiOS 7.0 and later support Kerberos as an authentication mechanism, in addition to the NTLM-based authentication supported by previous RiOS versions. Kerberos authentication support is beneficial for access to SharePoint, Exchange, IIS, and other Microsoft applications that use Active Directory and Kerberos for authentication. With this feature, your system is capable of prefetching resources when the web server employs per-request Kerberos authentication.

Prior to RiOS 7.0, servers that required Kerberos authentication did not take advantage of the parse-and-prefetch optimization feature. RiOS 7.0 and later can decrypt the Kerberos service ticket and generate session keys to authenticate, on a per-request basis, with the web server. For more information about Kerberos, see “Kerberos” on page 56.

The following is a list of HTTP authentication optimization methods:

- **Reuse auth** - URL learning and parse-and-prefetch sends a particular request from the browser and triggers the HTTP optimization module to prefetch the objects of a web page. The browser typically opens parallel TCP connections to the server to download the objects (for details, see “Multiple TCP connections and pipelining” on page 77). If the Reuse Auth feature is not enabled, the HTTP optimization module does not distribute the objects to the client through an unauthenticated connection, even though it can already have the objects in its database. With reuse auth enabled, the HTTP optimization module requires that the session has already been authenticated and that it is safe (in other words, without violating any permissions) to deliver the prefetched objects to the client regardless of whether the connection is authenticated or not. Reuse Auth is as if the client only used a single connection to download all the objects in serial.
- **Force NTLM** - The default authentication behavior on Microsoft's IIS server is per-request authentication for Kerberos and per-connection authentication for NTLM. Thus, the HTTP optimization module is configured to change the client-to-server negotiation so that the client chooses an authentication that maximizes the benefit of the HTTP optimization module. When enabled, Force NTLM removes the WWW-Authenticate: Negotiate line from the 401 Unauthorized message from the server. When the 401 Unauthorized message arrives at the client, the only authentication option available is NTLM. The client has no choice but to use NTLM for authentication.

  Do not use this feature if Kerberos authentication is required.
- **Strip auth header** - When you enable strip auth header, the HTTP optimization module detects an authentication request on an already authenticated connection. It removes the authentication header before sending the request to the server. Because the connection is already authenticated, the server delivers the object to the client without having to go through the entire authentication process again. If you require Kerberos authentication, do not use strip auth header.

  Strip Auth Header specifically addresses the issue described in “Connection jumping” on page 80.

- **Gratuitous 401** - You can use this feature with both per-request and per-connection authentication but it is most effective when used with per-request authentication. With per-request authentication, every request must be authenticated against the server before the server would serve the object to the client. Most browsers do not cache the server response requiring authentication, which wastes one round-trip for every GET request. With gratuitous 401, the client-side SteelHead caches the server response. When the client sends the GET request without any authentication headers, it responds locally with a 401 unauthorized message and saves a round trip.

  The HTTP optimization module does not participate in the actual authentication. The HTTP optimization module informs the client that the server requires authentication without requiring it to waste one round trip.

To enable HTTP Kerberos authentication

1. Install RiOS 7.0 or later on the client-side and server-side SteelHead.
2. Join the server-side SteelHead to the active domain directory.
3. Enable Kerberos key replication on the server-side SteelHead.
4. On the HTTP page, select Enable Kerberos Authentication Support.
5. Click **Apply**.

HTTP automatic configuration

RiOS 7.0 and later reduces the complexity of configuring the HTTP optimization features by introducing an automatic configuration method. HTTP optimization can automatically detect new HTTP web hosts and initiate an evaluation of the traffic statistic to determine the most optimal setting. Rule sets are built automatically. The analysis of each HTTP application is completed and configured independently per client-side SteelHead.

We recommend that both the client-side and server-side SteelHeads are running RiOS 7.0 or later for full statistics gathering and optimization benefits. For more details, go to https://supportkb.riverbed.com/support/index?page=content&id=S17039.

HTTP automatic configuration is enabled by default—the SteelHead starts to profile HTTP applications upon startup.

A detailed description of the HTTP automatic configuration process phases are as follows:

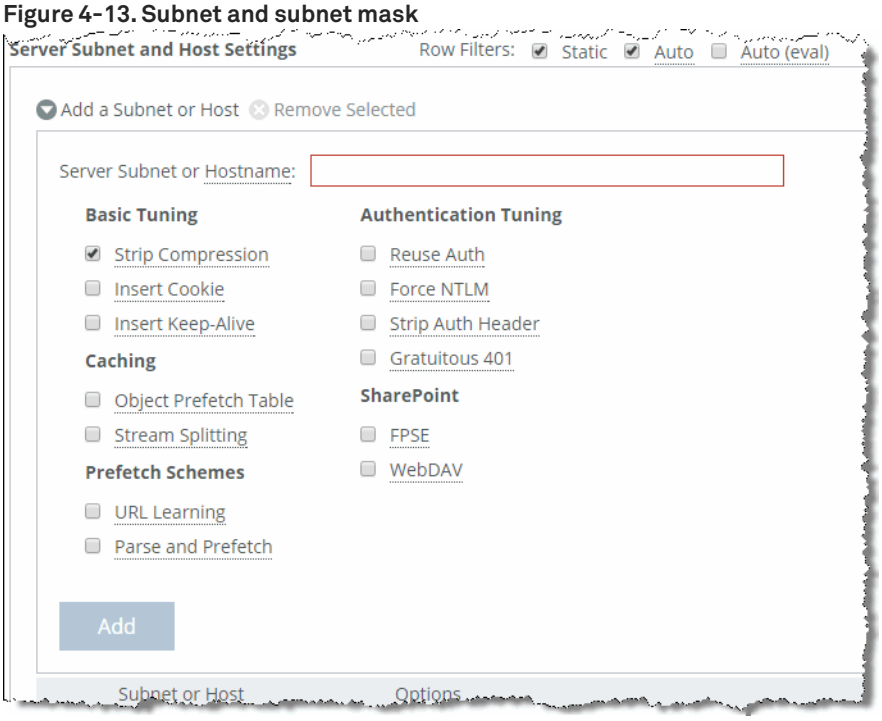
1. **Identification phase** - HTTP applications are identified by a hostname derived from the HTTP request header: for example, host: sharepoint.example.com or www.riverbed.com. This makes applications more easily identifiable when an HTTP application is represented by multiple servers. For example, http://sharepoint.example.com is collectively resolved to multiple IP addresses, each belonging to multiple servers for the purpose of load balancing. A single entry encompasses the multiple servers.
2. **Evaluation phase** - The SteelHead reads the metadata and builds latency and throughput-related statistics on a per-HTTP application basis. After the SteelHead detects a predetermined number of transactions, it moves to the next phase. During the evaluation phase, the web server host has an

object prefetch table, insert keep alive, reuse authentication, strip authentication and gratuitous 401 enabled by default to provide optimization. Strip authentication might be disabled after the evaluation stage if the SteelHead determines that the HTTP server requires authentication.

3. **Automatic phase** - The HTTP application profiling is complete and the HTTP application is configured. At this phase, prefetch (URL learning and parse-and-prefetch), strip-compression, and insert-cookie optimization features are declared as viable configuration options for this application, in addition to the options from the earlier phase. Evaluation of prefetching is based on the time difference between the server-side SteelHead to the server and the server-side SteelHead to the client-side SteelHead. Prefetch is enabled if the time difference is significantly greater. Stripping the compression is enabled if the server-side SteelHead LAN bandwidth is significantly greater than the WAN of the client-side SteelHead. Insert-cookie optimization is automatically enabled only when the server does not use cookies.

4. **Static phase** - You can insert custom settings for specific hosts or subnets. You can also select an automatically configured rule and override the settings. In RiOS 7.0 and later, you can insert hostnames (for example, www.riverbed.com) along with specific IP hosts/subnets (for example, 10.1.1.1/32). If you use a subnet instead of a hostname, you must specify the subnet mask.

Figure 4-13. Subnet and subnet mask



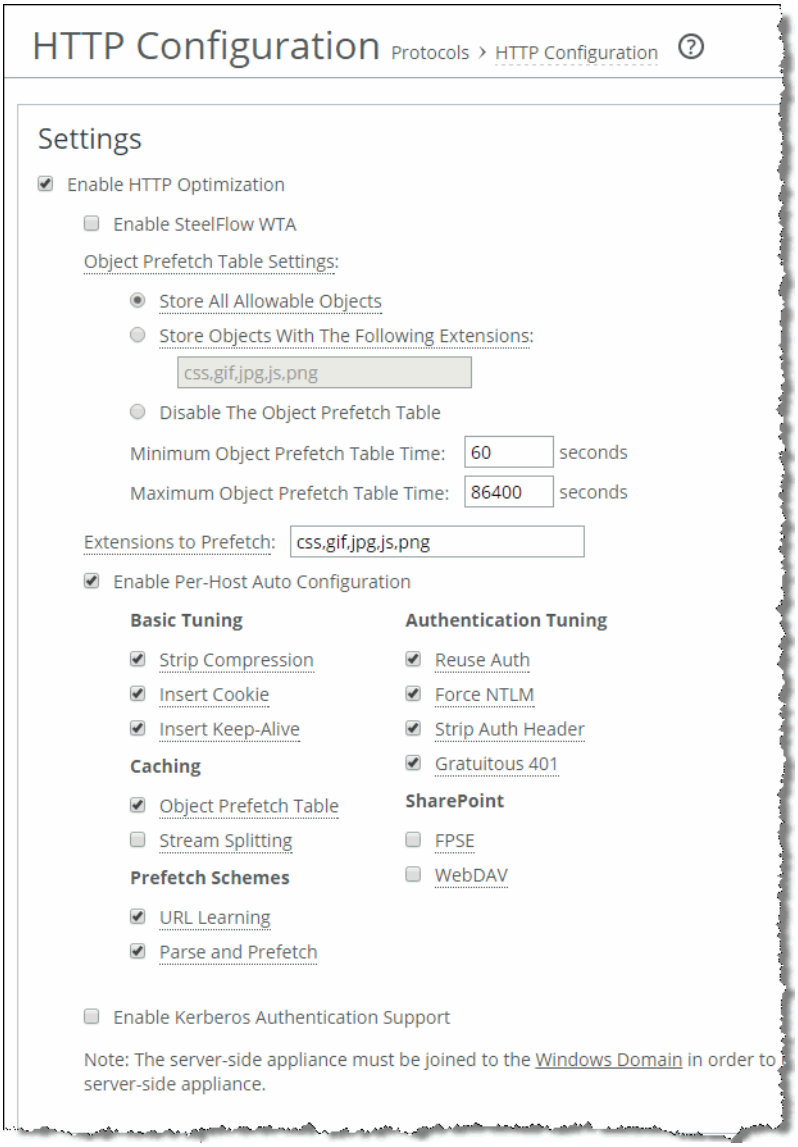
RiOS 7.0 and later stores all cacheable objects, as long as the objects are allowed by the HTTP server and not prohibited by the headers in the HTTP request and response. The restriction to cache objects based on their extensions is removed. You can choose to cache all allowable objects or change to pre-RiOS 7.0 settings and explicitly point out the extension of certain object you wish to store. You can also choose to not store any objects.

To enable HTTP automatic configuration

1. In the Management Console, choose Optimization > Protocols: HTTP.
2. Select Enable Per-Host Auto Configuration.
3. Select the appropriate options. All are enabled by default.

   For more information about HTTP options, see the *SteelHead User Guide*.
4. Click **Apply**.

Figure 4-14. HTTP configuration page



You can filter between the discovered and statically added hosts. When you enable HTTP automatic configuration, you can select the row filters to see auto-configured hosts and evaluated hosts. To delete the entry, you must edit an automatically configured host. The host automatically changes its configuration to become static. Only then can you remove the entry.

Note: Removal of an entry allows the SteelHead to relearn the HTTP application and begin the evaluation phase again.

You can select an automatically configured rule and edit the optimization settings. Upon completion, the rule set is now considered a static rule. This process is similar to previous RiOS versions in which manually adding subnets to optimize hosts was needed.

When you disable the automatic configuration option, only static and the default optimization rule (0.0.0.0/0) apply. The automatic and evaluation rules no longer populate the table.

Note: If you restart the optimization process, you remove all automatic and evaluation configured entries. The SteelHead profiles the applications again.

To configure HTTP automatic configuration with the command line

- On the client-side SteelHead, connect to the CLI in configuration mode and enter the following command:

```
protocol http auto-config enable
```

To clear the HTTP automatic configuration statics with the command line

- On the client-side SteelHead, connect to the CLI in configuration mode and enter one of the following commands:

```
protocol http auto-config clear-stats all
```

or

```
protocol http auto-config clear-stats hostname <http-hostname>
```

For more information about how to configure HTTP automatic configuration, see the *SteelHead User Guide* and the *Riverbed Command-Line Interface Reference Manual*.

HTTP Settings for common applications

We recommend that you test the HTTP optimization module in a real production environment. To manually configure HTTP applications, the following table shows the recommended settings for common enterprise applications.

Application	URL learning	Parse-and-prefetch	OPT
SAP/Netweaver	No	Yes	Yes
Microsoft CRM	Yes	No	Yes
Agile	No	No	Yes
Pivotal CRM	No	Yes	Yes
SharePoint	No	Yes	Yes

As discussed in “Primary content optimization methods” on page 86, URL learning accelerates environments with static URLs and content, parse-and-prefetch accelerates environments with dynamic content, and OPT saves round trips across the WAN, and reduces WAN bandwidth consumption.

HTTP optimization for SharePoint

You can configure Microsoft SharePoint-specific HTTP optimization options in RiOS 8.5 and later. The SharePoint optimization options are available on the Optimization > Protocols: HTTP page (Figure 4-14):

- **Microsoft Front Page Server Extensions (FPSE) protocol** - The FPSE protocol enables the client application to display the contents of a website as a file system. FPSE supports uploading and downloading files, directory creation and listing, basic file locking, and file movement in the web server. The FPSE HTTP subprotocol is used by SharePoint 2007 and 2010 when opening documents in Microsoft Office. To increase performance, the following FPSE requests are cached:
  - URL-to-Web URL request
  - Server-version request
  - Open-service request

  One of the inherent issues with SharePoint communication is that after each request is complete, the web server closes the connection. Thus, each new request requires a new TCP handshake. If you are using SSL, additional round trips are required. Request caching eliminates the round trips and speeds up the connection.

  FPSE supports SharePoint Office clients 2007 and 2010, installed on Windows 7 (SP1) and Windows 8. SharePoint 2013 does not support FPSE.
- **Microsoft Web Distributed Authoring and Versioning (WebDAV)** - The WebDAVE represents a set of standardized extensions (RFC 4918) to the HTTP/1.1 protocol that enables users to collaborate, edit, and manage files on remote web servers. Specific to SharePoint, WebDAV is a protocol for manipulating the contents of the document management system. You can use WebDAV protocol to map a SharePoint site to a drive on a Windows client machine. To properly set up as a WebDAV drive, you must create it as a mapped drive and not as a new network folder location.

  The WebDAV optimization option in the RiOS cache often repeats metadata and replies locally from the client-side SteelHead. WebDAV protocol is used in both SharePoint 2010 and 2013.
- **File Synchronization via SOAP over HTTP (FSSHTTP) protocol** - FSSHTTP is a protocol you can use for file data transfer in Microsoft Office and SharePoint. FSSHTTP enables one or more protocol clients to synchronize changes performed on shared files stored on a SharePoint server. FSSHTTP allows coauthoring and is supported with Microsoft Office 2010 and 2013 and SharePoint 2010 and 2013.

  Currently, RiOS does not offer any latency optimization for FSSHTTP. Clients can continue to benefit from HTTP optimization, TCP streamlining, and data reduction.

- **OneDrive for Business** - OneDrive for Business is a personal library intended for storing and organizing work documents. As an integral part of Office 365 or SharePoint 2013, OneDrive for Business enables end users to work within the context of your organization, with features such as direct access to your company address book. OneDrive for Business is usually paired with Sync OneDrive for Business (sync client). This synchronization application enables you to synchronize OneDrive for Business library or other SharePoint site libraries to a local computer. OneDrive for Business is available with Office 2013 or with Office 365 subscriptions that include Microsoft Office 2013 applications.

  RiOS 8.6 and later support optimization of traffic between OneDrive for Business and Sync OneDrive for Business. Office 365 OneDrive for Business requires a SteelHead SaaS. SteelHead SaaS runs in the Akamai SRIP network, and it optimizes both SharePoint and Exchange.

The following table shows a summary of SharePoint application visibility reporting.

SharePoint Version	HTTP	FPSE	WebDAV
SharePoint 2007	Yes	Yes	Yes
SharePoint 2010	Yes	Yes	Yes
SharePoint 2013	Yes	No	No

For information about how to configure SharePoint optimization and more information about the SharePoint optimization options, see the *SteelHead User Guide* and the *Riverbed Command-Line Interface Reference Manual*. For information about AFE, see the *SteelHead Deployment Guide* and the *SteelHead User Guide*.

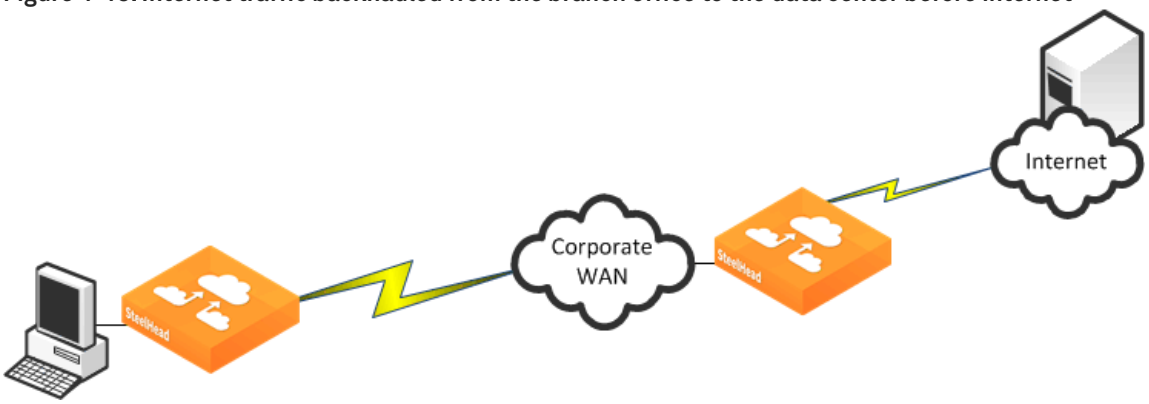
HTTP optimization module and internet-bound traffic

Sometimes internet access from the branch offices is back-hauled to the data center first, before going out to the internet. Consequently, a significant portion of the WAN traffic is internet-bound traffic and you might be tempted to optimize that traffic.

For internet-bound traffic, we recommend that you enable HTTP automatic configuration.

Similar to other Layer-7 optimization modules, the HTTP optimization module expects the latency between the server and the server-side SteelHead to be LAN latency. With internet-bound traffic, the latency between the server-side SteelHead and the server is going to be far greater than typical LAN latency. In this case, the URL learning and parse-and-prefetch features are not very effective, and in certain cases they are slower than without optimization. we recommend that you do not disable HTTP automatic configuration—the SteelHead adjusts as necessary.

Figure 4-15. Internet traffic backhauled from the branch office to the data center before internet



HTTP and IPv6

HTTP enables you to specify settings for a particular server or subnet using an IPv6 address similar to how you configure HTTP for IPv4 addresses. The following shows an example from the CLI:

```
protocol http server-table subnet 2001:dddd::100/128 obj-pref-table no parse-prefetch no url-learning no reuse-auth no strip-auth-hdr no gratuitous-401 no force-nego-ntlm no strip-compress no insert-cookie yes insrt-keep-aliv no FPSE no WebDAV no FSSHTTP no
```

For more information about IPv6, see the *SteelHead Deployment Guide*.

Overview of the web proxy feature

RiOS 9.1 and later include the web proxy feature. Web proxy uses the traditional single-ended internet HTTP-caching method enhanced by Riverbed for web browsing methodologies of today. The web proxy feature enables SteelHeads to provide a localized cache of web objects and files. The localized cache alleviates the cost of repeated downloads of the same data. Using the web proxy feature in the branch office provides a significant overall performance increase due to the localized serving of this traffic and content from the cache. Furthermore, multiple users accessing the same resources receive content at LAN speeds while freeing up valuable bandwidth.

Because this is a client-side feature, it is controlled and managed from an SCC. You can configure the in-path rule on the client-side SteelHead running the web proxy or on the SCC. You must also enable the web proxy globally on the SCC and add domains to the global HTTPs white list. For more information, see the *SteelCentral Controller for SteelHead Deployment Guide*, the *SteelCentral Controller for SteelHead User's Guide*, and the *SteelHead User Guide*.

Note: SteelHead-v models VCX-10 through VCX-110 running RiOS 9.6 software and later support the web proxy feature. xx50 models, xx60 models, and other SteelHead-v models do not support this feature.

Tuning Microsoft IIS server

This section describes the steps required to modify the IIS server so that it can maximize the performance of the HTTP Authentication optimization module. In some instances of the HTTP Authentication optimization requires the IIS server to behave in a certain way. For example, if NTLM authentication is in use, then the HTTP optimization module expects the NTLM authentication to use per-connection authentication.

Determining the current authentication scheme on IIS

IIS 6 uses *metabase* to store its authentication settings. Although you can manually modify the metabase, the supported method in querying or modifying the metabase is to use the built-in Visual Basic (VB) script adsutil.vbs. The adsutil.vbs script is located in the C:\inetpub\adminscripts directory on the IIS server.

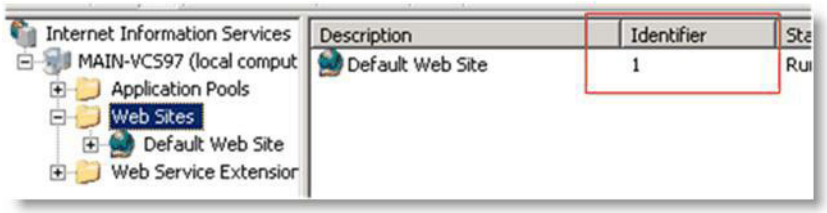
To determine the current setting on the IIS server

- From the command prompt on the Windows server, enter the following command in the c:\inetpub\adminscripts directory:

```
cscript adsutil.vbs get w3svc/$WebsiteID$/root/NTAuthenticationProviders
```

The $WebsiteID$ is the ID of the website hosted by the web server. The $WebsiteID$ can be found in the IIS Manager and it's the number under the Identifier column.

Figure 4-16. Web Site ID in the Identifier column



If the output of the command is one of the following, it indicates that the server prefers Kerberos authentication and if that fails, it falls back to NTLM:

```
The parameter "NTAuthenticationProviders" is not set at this node.
NTAuthenticationProviders       : (STRING) "Negotiate,NTLM"
```

If the output of the command is the following, it indicates that the server only supports NTLM authentication:

```
NTAuthenticationProviders       : (STRING) "NTLM"
```

Determining the current authentication mode on IIS

Both NTLM and Kerberos authentication can support either per-connection or per-request authentication. To determine the current authentication mode on IIS, use the same VB script, adsutil.vbs, but query a different node:

```
cscript adsutil.vbs get w3svc/$WebsiteID$/root/AuthPersistSingleRequest
```

The output of the command should be similar to the following text:

```
The parameter "AuthPersistSingleRequest" is not set at this node; or
```

```
AuthPersistSingleRequest        : (BOOLEAN) False; or

AuthPersistSingleRequest        : (BOOLEAN) True
```

Per-connection or per-request NTLM authentication

If the authentication scheme is NTLM and the output of the authentication mode is either *not set* or *false*, then the server is configured with per-connection NTLM authentication. If the output of the authentication mode is *true*, then the server is configured with per-request NTLM authentication.

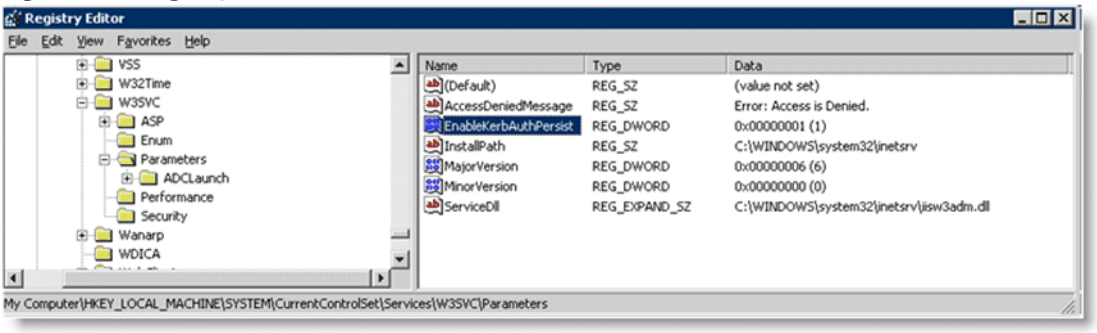
Per-connection or per-request Kerberos authentication

If the authentication scheme is Kerberos, you must perform extra steps you can determine whether the server is using per-connection or per-request authentication.

To determine whether Kerberos is using per-connection or per-request (applies to IIS 6.0)

1. Check the registry for key named EnableKerbAuthPersist under HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters. If the key does not exist, or if the key does exist but has a value of zero, then Kerberos is using per-request authentication.

Figure 4-17. Registry Editor window



2. If the key exists but has a nonzero value, and the output from authentication mode is false, then Kerberos might be using per-connection Kerberos. Kerberos is using per-connection authentication if the server running is running IIS 6.0 and has the patch installed per Microsoft's knowledge base article http://support.microsoft.com/kb/917557.

For information about IIS 7.0, go to http://support.microsoft.com/kb/954873.

The following table shows the different combinations and whether Kerberos would perform per-connection or per-request authentication.

		EnableKerbAuthPersist/AuthPersistNonNTLM*		
		Not-set	Nonzero	Zero
AuthPersistSingleRequest	Not-set	Per-request	Per-connection**	Per-request
	TRUE	Per-request	Per-request	Per-request
	FALSE	Per-request	Per-connection	Per-request

* AuthPersistNonNTLM of IIS 7 replaces EnableKerbAuthPersist of IIS 6

** Requires IIS patch

Changing the authentication scheme

You can change the authentication by using the adsutil.vbs script.

To modify the server so that it only supports NTLM authentication

- From the command prompt on the Windows server, enter the following command:

```
cscript adsutil.vbs set w3svc/$WebsiteID$/root/NTAuthenticationProviders "NTLM"
```

You can configure the server to attempt Kerberos authentication first before NTLM by changing the last parameter to *Negotiate, NTLM*.

Remember to restart the IIS server after the changes have been made.

Changing the per-connection/per-request NTLM authentication mode

By default, NTLM uses per-connection authentication.

To change per-connection NTLM to per-request NTLM

- From the command prompt on the Windows server, enter the following command:

```
cscript adsutil.vbs set w3svc/$WebsiteID$/root/AuthPersistSingleRequest TRUE
```

To change NTLM back to its default, replace the word TRUE with FALSE and restart the IIS server.

Note: When you use NTLM, the HTTP optimization module works best when per-connection authentication is set.

Changing the per-connection/per-request Kerberos authentication mode

By default, Kerberos uses per-request authentication. There are several ways to configure Kerberos to use per-connection authentication. For details, see “Per-connection or per-request Kerberos authentication” on page 99.

HTTP authentication settings

The following table shows some of the recommended configurations for the HTTP Authentication optimization. In this instance, assume that it is not possible to make any modifications on the IIS server.

For example, if the authentication setting on the IIS server is per-request Kerberos, enabling Force NTLM forces the client to use NTLM and in turn, the HTTP optimization module can provide better optimization by using the URL learning and parse-and-prefetch features. If NTLM authentication is not an option, then the only possibility is to enable gratuitous 401.

If you modify the settings on the IIS server, then Force NTLM might not be necessary. In this case, only the first row of the following table is applicable.

The HTTP optimization module expects the default authentication behavior for both NTLM and Kerberos (in other words, per-connection authentication for NTLM and per-request authentication for Kerberos). Continuing with the example earlier, if the IIS server is configured to use the nonstandard authentication scheme by using per-request Kerberos and per-request NTLM, then using Force NTLM does not help as it changes from per-request Kerberos to per-request NTLM. In this case, URL learning and parse-and-prefetch is not effective.

IIS authentication	Recommended configurations	Notes
per-connection NTLM	Reuse Auth. + Strip Auth. Header + Grat. 401	
	Reuse Auth. + Strip Auth. Header	
per-request Kerberos	Reuse Auth. +Force NTLM + Strip Auth. Header + Grat. 401	N/A if NTLM is not an option
	Reuse Auth. +Force NTLM + Strip Auth. Header + Grat. 401	N/A if NTLM is not an option
	Grat. 401	
per-request NTLM	Grat. 401	
per -connection Kerberos	Reuse Auth. + Force NTLM + Strip Auth. Header	N/A if NTLM is not an option
	Reuse Auth.	
	Change to per-request Kerberos and turn on	
	Grat. 401	
Don't know	Reuse Auth. + Strip Auth. Header + Grat. 401	
	Reuse Auth. + Strip Auth. Header	

HTTP optimization module and proxy servers

Some network deployments might involve HTTP proxy servers to speed up resource retrieval, apply access control policies, or audit and filter contents. In general, the SteelHeads provide full optimization benefits even with proxy servers in place. Optimized connections can be limited or even hindered in the following circumstances:

- **Proxy authentication** - If the proxy server uses authentication, prefetching performance can be affected. Similar to web server authentication, proxy authentication can limit the prefetching capability of the connection where prefetched resources are served when the proxy server uses per-connection authentication. In the case of the per-request mode, prefetching is not possible as every single request requires user authentication. HTTP authentication optimization features apply to both web server and proxy authentication.
- **Proxy caching** - A proxy server can maintain its own local cache to accelerate access to resources. Generally, proxy caching does not thwart HTTP optimization. If it caches objects for an excessively long time, object prefetch table (OPT) might not provide full benefits as the retrieved resource might not be sufficiently fresh.
- **Selective proxying** - If the same web server is accessed directly by a client and through a proxy server, URL learning might not work properly. URL learning builds a prefetch tree by observing ongoing requests. This is based on the assumption that the same URL is requested by the client if the same base page is requested again.

  When a prefetch tree is created without a proxy, the observed partial URLs comprise the tree. If another client asks for the same page but through a proxy, the proxy fails to forward the prefetched responses because it expects full URLs to be able to send the responses back to the client. Thus URL learning might not work well when proxy is selectively employed for the same host. This only happens with URL learning. Other prefetching schemes, such as parse-and-prefetch and object prefetch table, are not subjected to this issue.
- **Fat Client** - Not all applications accessed through a web browser use the HTTP protocol. This is especially true for fat clients that run inside a web browser that might use proprietary protocols to communicate with a server. HTTP optimization does not improve performance in such cases.

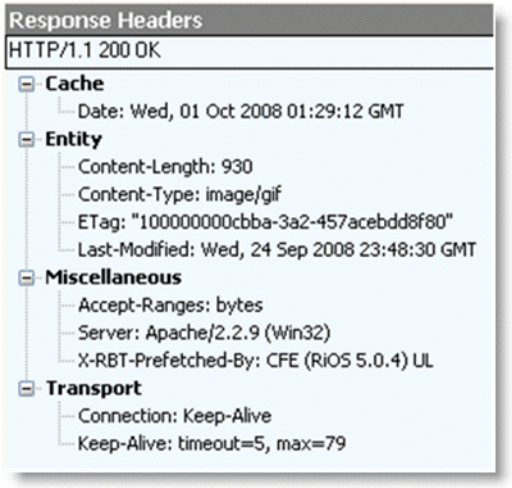
Determining the effectiveness of the HTTP optimization module

Figure 4-18 shows that when an object is optimized by the HTTP optimization module, the response header contains the line X-RBT-Prefetched-By. The X-RBT-Prefetched-By line also contains the name of the SteelHead, the version of system, and the method of optimization. The different methods of optimization are:

- **UL** - URL learning.
- **PP** - parse-and-prefetch.
- **MC** - metadata response (pre-RiOS 6.0)
- **PT** - object prefetch table (RiOS 6.0 and later)

- **AC** - gratuitous 401

Figure 4-18. Response header contained in the line X-RBT-Prefetched-By



The response header can be captured by using tcpdump, HTTPWatch, Fiddler, or similar tools. Because the HTTP optimization module is a client-side driven feature, the capture must be done on the client itself.

Info-level logging

You can look at the client-side SteelHead log messages to determine whether or not the HTTP optimization module is functioning. The logging level on the SteelHead must be set at info level logging for the messages to appear in the log. If the HTTP optimization module is prefetching objects, a message similar to the following appears in the logs:

```
Sep 25 12:28:57 CSH sport[29969]: [http/client.INFO] 2354 {10.32.74.144:1051 10.32.74.143:80}
Starting 40 Prefetches for Key->abs_path="/" host="10.32.74.143" port="65535" cookie="rbt-
http=2354"
```

This only applies to the URL learning and parse-and-prefetch. No log messages are displayed for metadata response or object prefetch table.

The key specified in the log message is not necessarily the object that triggered the prefetch operation. In RiOS 6.0 and later, the log message includes the object that triggered the prefetch operation.

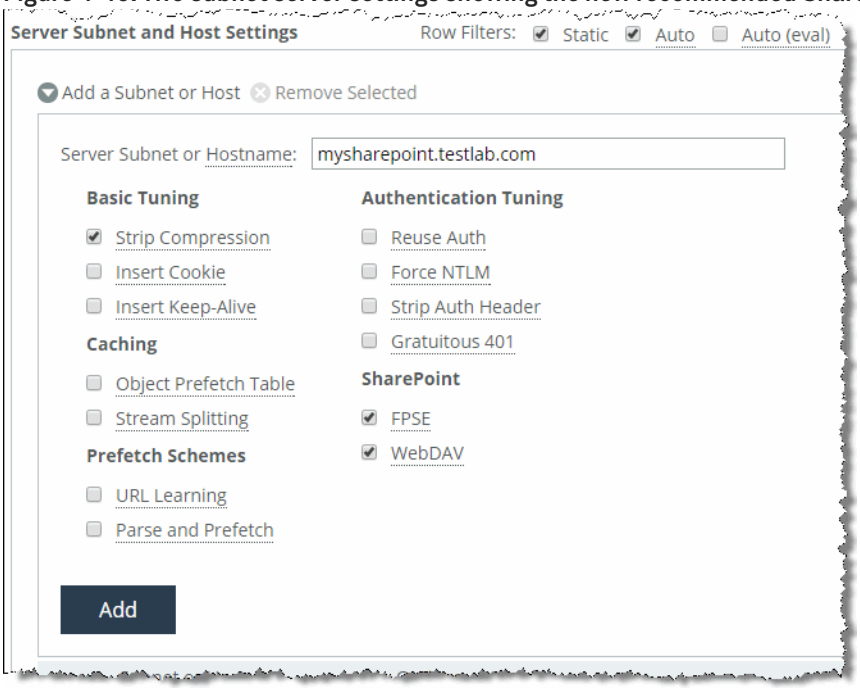
Use case

While automatic configuration is typically the preferred method of configuration, you have the option of manual configuration. The following use case shows a manual configuration.

A customer has a 1.5 Mbps link with 100 ms latency between the branch office and the data center. The PCs in the remote office are running Microsoft Windows XP with Internet Explorer 7. Users in the remote offices are complaining of slow access for SAP Netweaver and Microsoft SharePoint. The SAP Netweaver server has an IP address of 172.30.1.10 and the Microsoft SharePoint server has an IP address of 172.16.2.20.

Because both SAP Netweaver and Microsoft SharePoint are well-known applications, the customer configured the following on the client-side SteelHead.

Figure 4-19. Two subnet server settings showing the new recommended SharePoint settings



After configuring the settings above, the customer noticed a significant improvement in response time for SAP Netweaver but no changes for Microsoft SharePoint—even though the connections are optimized with good data reduction. One of the users mentioned that the Microsoft SharePoint portal required authentication, which might be the reason why parse-and-prefetch did not work. Unfortunately, the system administrator in charge of the SharePoint portal cannot be reached at this moment and you cannot check the authentication setting on the server.

Instead of checking the authentication on the server, you can capture tcpdump traces and check for the authentication scheme in use. Figure 4-20 shows the server has Kerberos enabled and hence the client attempts to authenticate using Kerberos first.

Figure 4-20. TCP dump trace confirming Kerberos enabled

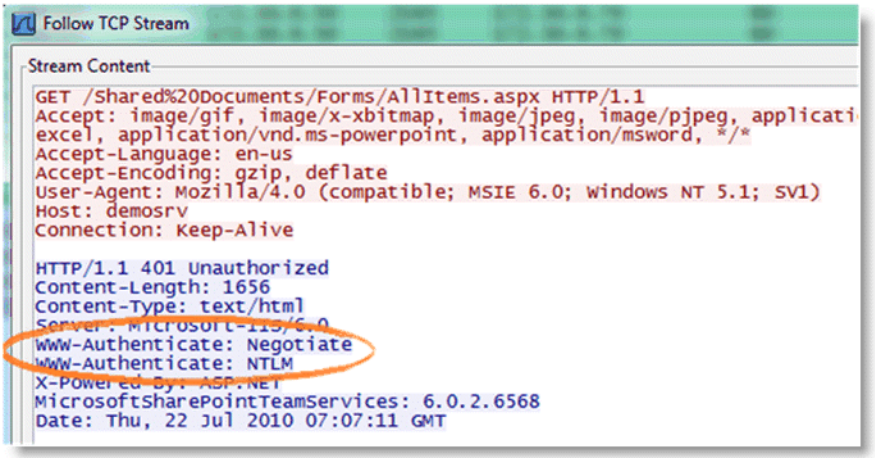


Figure 4-21 confirms that by scrolling through the trace, the per-request Kerberos is configured on the server.

Figure 4-21. TCP dump trace showing per-request Kerberos

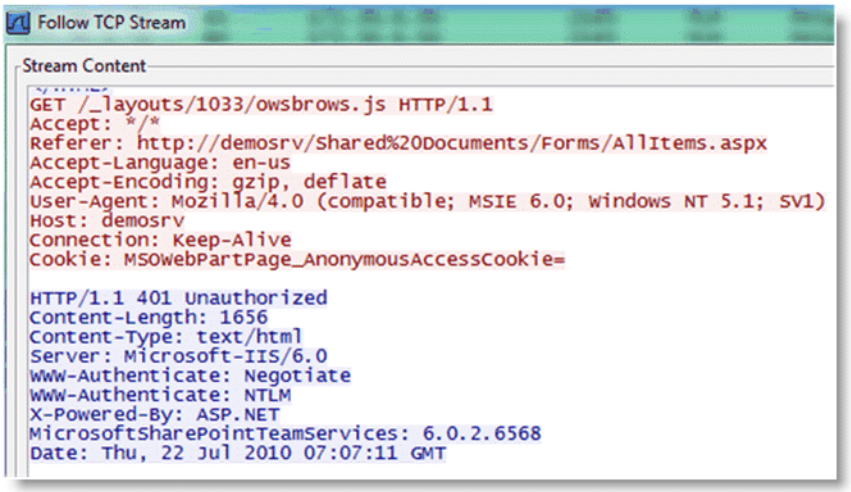
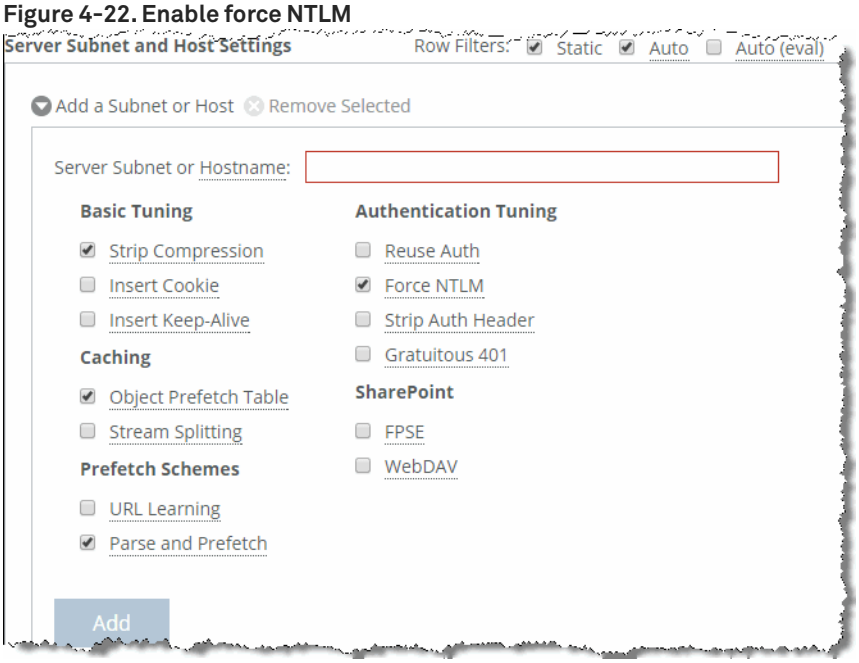


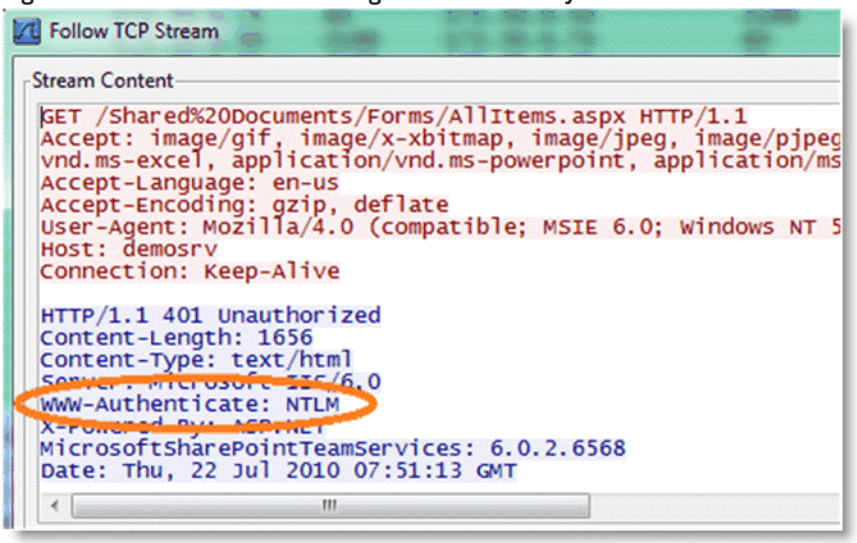
Figure 4-22 shows that given this information, the best option is to enable **Force NTLM** for the SharePoint server.

Figure 4-22. Enable force NTLM



Taking another trace on the client-side SteelHead confirms that the only authentication option available is NTLM. Because there is no other authentication option but NTLM, the client is forced to authenticate via NTLM and parse-and-prefetch and prefetches can once again function as before.

Figure 4-23. Trace stream showing NTLM as the only authentication available



In this instance, it is not necessary to enable the other features, as the entire transaction took place over a single connection. If the client uses multiple TCP connections, then it might be necessary to enable reuse auth, strip auth header, and gratuitous 401. Enabling the rest of the features does not provide any benefit in this instance, but it does not cause any problems either.

5

Citrix ICA Optimization

To consolidate operations, some organizations deploy desktop and application virtualization solutions such as Citrix XenDesktop and XenApp in the data center. Citrix uses a proprietary protocol called Independent Computing Architecture (ICA) to provide connectivity between its clients (called *receivers*) and its published applications and desktops.

This chapter includes the following sections:

- "Overview of Citrix ICA" on page 107
- "Citrix ICA traffic optimization with SteelHeads" on page 108
- "Citrix SecureICA encryption" on page 110
- "Citrix drive-mapping optimizations" on page 110
- "Citrix multi-stream ICA traffic optimization with SteelHeads" on page 112
- "QoS classification for Citrix traffic" on page 117
- "Automatic negotiation of multi-stream ICA traffic for QoS enforcement" on page 121
- "Reduction for Citrix small packet real-time traffic" on page 123
- "Citrix ICA optimization over SSL" on page 123

Overview of Citrix ICA

RiOS 6.0 and later provide these optimizations:

- Classification and shaping of Citrix ICA traffic using Riverbed QoS to improve the end-user desktop experience
- Bandwidth reduction of compressed and encrypted Citrix ICA traffic using SteelHead Citrix optimization

RiOS 7.0 and later provide these optimizations:

- Latency optimization for client drive mapping in the Citrix ICA session
- Optimization of Citrix sessions over SSL using Citrix Access Gateway (CAG)
- SteelHead Citrix Optimization for multi-port ICA traffic

RiOS 7.0.4 and later provide traffic optimization for enhanced data reduction for small Citrix packets.

Citrix version support

RiOS 6.0 and later provide support for the following Citrix software components.

Citrix Receiver or ICA client versions:

- Online plug-in version 9.x
- Online plug-in version 10.x
- Online plug-in version 11.x
- Online plug-in version 12.x
- Online plug-in version13.x (Receiver version 3.x)
- Receiver for Windows version 4.x

Citrix XenDesktop:

- XenDesktop 4
- XenDesktop 5
- XenDesktop 5.5
- XenDesktop 5.6
- XenDesktop 7.x

Citrix XenApp:

- Presentation Server 4.5
- XenApp 5
- XenApp 6
- XenApp 6.5
- XenApp 7.x

In addition, RiOS supports encrypted and compressed Citrix ICA traffic optimization.

For information about configuring Citrix optimization, see the *SteelHead User Guide*, the *Riverbed Command-Line Interface Reference Manual*, and the white paper *Optimizing Citrix ICA Traffic with RiOS 8.0 (June 2013)*.

Citrix ICA traffic optimization with SteelHeads

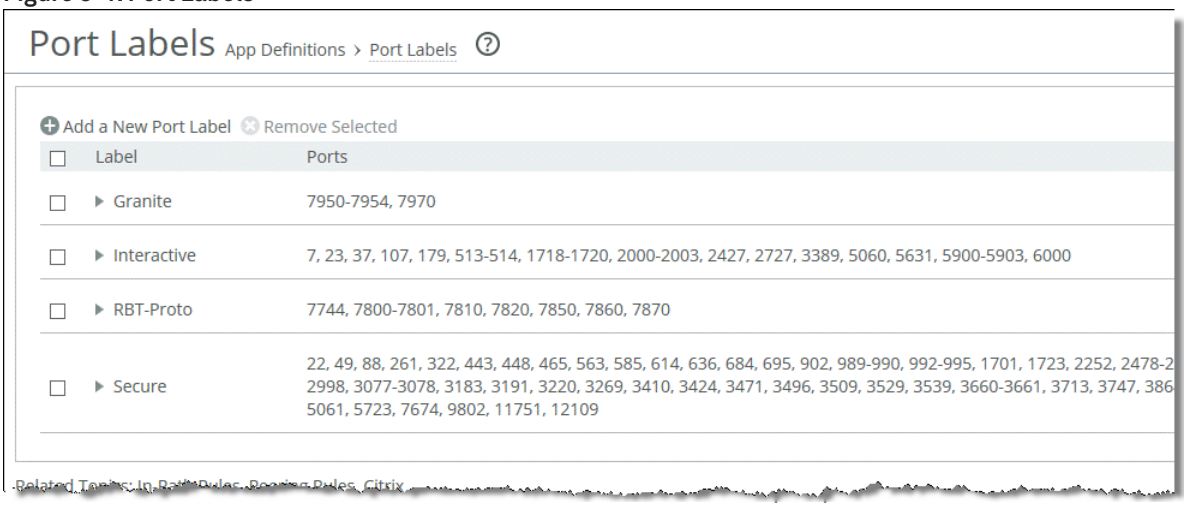
Citrix ICA traffic is usually transported on TCP port 1494. Citrix ICA traffic is transported on TCP port 2598 if Citrix session reliability is enabled. When you enable Citrix session reliability, the client tunnels its ICA traffic inside the Common Gateway Protocol (CGP) port 2598.

For more information about Citrix session reliability, go to http://support.citrix.com/article/CTX104147.

To configure Citrix ICA traffic optimization with a SteelHead

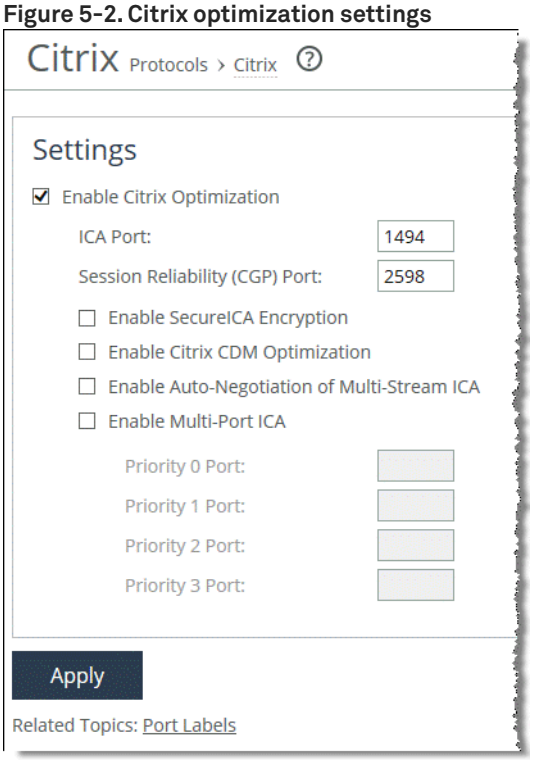
1. From the SteelHead Management Console, choose Networking > App Definitions: Port Labels and remove ports 1494 and 2598 from the Interactive port label.

Figure 5-1. Port Labels



2. Choose Optimization > Protocols: Citrix.

Figure 5-2. Citrix optimization settings



3. Select Enable Citrix Optimization.

4. Specify the ICA port and session reliability port.

5. Click **Apply** to apply the settings to the running configuration.
6. Restart the optimization service.

You must perform this configuration on both the client-side and server-side SteelHeads.

Citrix SecureICA encryption

SecureICA is a Citrix ICA protocol feature that encrypts the session data transmitted between Citrix clients and servers. By default, all Citrix ICA sessions have SecureICA set to basic ICA protocol encryption. The basic ICA protocol encryption setting obfuscates data but does not provide industry standard encryption. Other settings are available to increase the level of encryption using the RC5 algorithm.

For more information about SecureICA, go to http://docs.citrix.com/en-us/xenapp-and-xendesktop/xenapp-6-5/xenapp65-w2k8-wrapper/xenapp65-admin-wrapper/ps-securing-wrapper-v2/ps-securing-incr-lev-cltsrvr-security.html.

RiOS supports optimization of Citrix ICA sessions with SecureICA set to RC5 40-, 56-, and 128-bit encryption. By default, RiOS can optimize Citrix ICA traffic with SecureICA set to basic ICA protocol encryption. You must enable SecureICA encryption to allow RiOS to optimize ICA sessions with SecureICA encryption set to RC5 on both the client-side and the server-side SteelHeads.

To enable RiOS to optimize Citrix Sessions with SecureICA encryption set to RC5 SecureICA

1. From the SteelHead Management Console, choose Optimization > Protocols: Citrix.
2. Select Enable Citrix Optimization.
3. Specify the ICA port and session reliability port.
4. Select Enable SecureICA Encryption.
5. Click **Apply** to apply the settings to the running configuration.
6. Restart the optimization service.

You must perform this configuration on both the client-side and server-side SteelHeads.

Citrix drive-mapping optimizations

Client drive mapping (CDM) is a Citrix ICA protocol feature that enables users to access their local drives (such as floppy disk drives, network drives, USB drives, CD-ROM drives, and hard disk drives) from within an ICA session. When you use CDM to access a mapped drive within an ICA session, the end-user desktop experience can be negatively affected by the underlying network performance.

RiOS 7.0 and later support CDM latency optimization. CDM latency optimization improves the end-user desktop experience when reading and writing files on a mapped drive in the following ways:

- **Reading a file (client-to-server transfer)** - The SteelHeads monitor the CDM virtual channel for files being read from the client drive. When the client-side SteelHead detects a chunk of file data being requested, it begins to read ahead and sends the file data in larger chunks to the server-side SteelHead. The server-side SteelHead buffers the file data until it is requested. This effectively eliminates many round trips across the WAN and improves the user desktop experience.
- **Writing a file (server-to-client transfer)** -The SteelHeads monitor the CDM virtual channel for files written to the client drive. Whenever the server sends file data to the client, the server-side SteelHead immediately sends an acknowledgment to the server so that the server continues to send file data quickly. Without this local acknowledgment, the server does not send any more data until it receives the acknowledgment from the client. This optimization eliminates round trips across the WAN, which improves the user desktop experience.

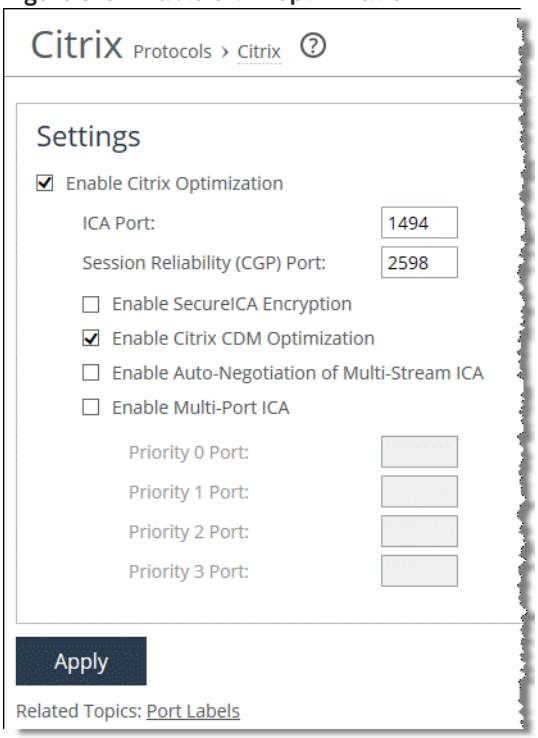
CDM latency optimization is available when both the client-side and server-side SteelHeads are running RiOS 7.0 or later. You cannot apply CDM latency optimization to Citrix sessions that use Citrix session reliability.

Note: File transfers greater than 1 GB do not perform as well as smaller file transfers.

To configure Citrix CDM latency optimization

1. Choose Optimization > Protocols: Citrix.
2. Select Enable Citrix Optimization.
3. Select Enable Citrix CDM Optimization (Figure 5-3).

Figure 5-3. Enable Citrix optimization



4. Click **Apply** to apply the settings to the running configuration.

5. Restart the optimization service.

To configure CDM with the CLI, use the following commands:

```
(config)# protocol citrix cdm enable
(config)# service restart
```

You must perform this configuration on both the client-side and server-side SteelHeads.

Citrix multi-stream ICA traffic optimization with SteelHeads

This section requires that you be familiar with the Citrix ICA protocol and how to configure your Citrix server. This section includes the following topics:

- "Citrix virtual channels and traffic priorities" on page 112
- "Single-stream and multi-stream ICA" on page 112

Citrix virtual channels and traffic priorities

The ICA traffic within a Citrix session comprises of many categories of traffic called *virtual channels*. A virtual channel provides a specific function of Citrix ICA remote computing architecture, such as print, CDM, audio and video. The ICA traffic within a Citrix session is also categorized by priority, in which virtual channels carrying real-time traffic, such as audio and video, are flagged with higher priority than virtual channels carrying bulk transfer traffic such as print and CDM. The ICA priority groups are as follows:

- Very High (priority 0)
- High (priority 1)
- Medium (priority 2)
- Low (priority 3)

For more information about Citrix ICA virtual channel, go to the Citrix Knowledge Base article CTX116890 at http://support.citrix.com/article/CTX116890.

Single-stream and multi-stream ICA

By default, a Citrix session uses a single TCP connection to carry traffic for all virtual channels and all priority groups. This is referred to as *single-stream ICA*.

XenApp 6.5 and XenDesktop 5.5 introduced a new feature, called *multi-stream ICA*, that enables use of multiple TCP connections to carry traffic for a Citrix session. Multi-stream ICA is available only when you enable Citrix session reliability. Multi-stream ICA carries traffic on port 2598 and three other user-configurable ports as defined in the multi-port Citrix computer policy. Each port represents an ICA priority group and enables you to apply true network-based QoS policies to the priority groups for the virtual channel traffic that they carry.

For more information about enhanced QoS with multi-stream ICA, go to http://blogs.citrix.com/2011/08/25/enhanced-qos-via-multi-stream-ica/.

This table shows the association of Citrix ICA priority groups to the various ICA virtual channels for multi-stream ICA.

Value	Priority	Description
0	Very high	Audio
1	High	Thin Wire/DX command remoting, seamless, MSFT TS licensing; SmartCard redirection; control virtual channel; and end-user experience monitoring
2	Medium	MediaStream (Windows media and Flash), USB redirection, clipboard, and CDM
3	Low	Printing, client COM port mapping, LPT port mapping, and legacy OEM virtual channels

You must use these software components to enable and optimize multi-stream and multi-port ICA:

- RiOS 7.0 and later
- Citrix XenApp 6.5 and later
- Citrix XenDesktop 5.5 and later
- Citrix Receiver 3.x (online plug-in 13.0) and later

In addition, you must enable Citrix session reliability**.** Citrix session reliability is enabled by default in XenApp 6.5 and XenDesktop 5.5.

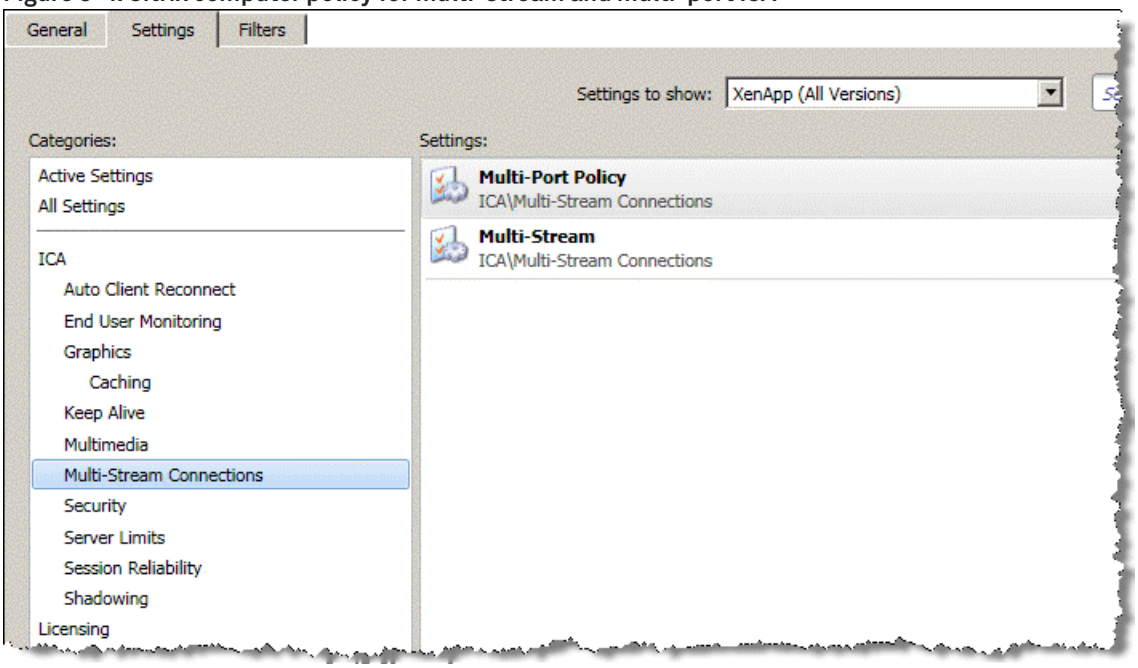
For more information about Citrix session reliability, go to http://support.citrix.com/article/CTX104147.

The following example shows how to configure multi-stream and multi-port ICA in XenApp 6.5.

To configure XenApp 6.5 multi-stream and multi-port ICA traffic optimization with a SteelHead

1. From the Citrix AppCenter or Windows Group Policy Editor, enable multi-stream ICA in the Citrix computer policy.

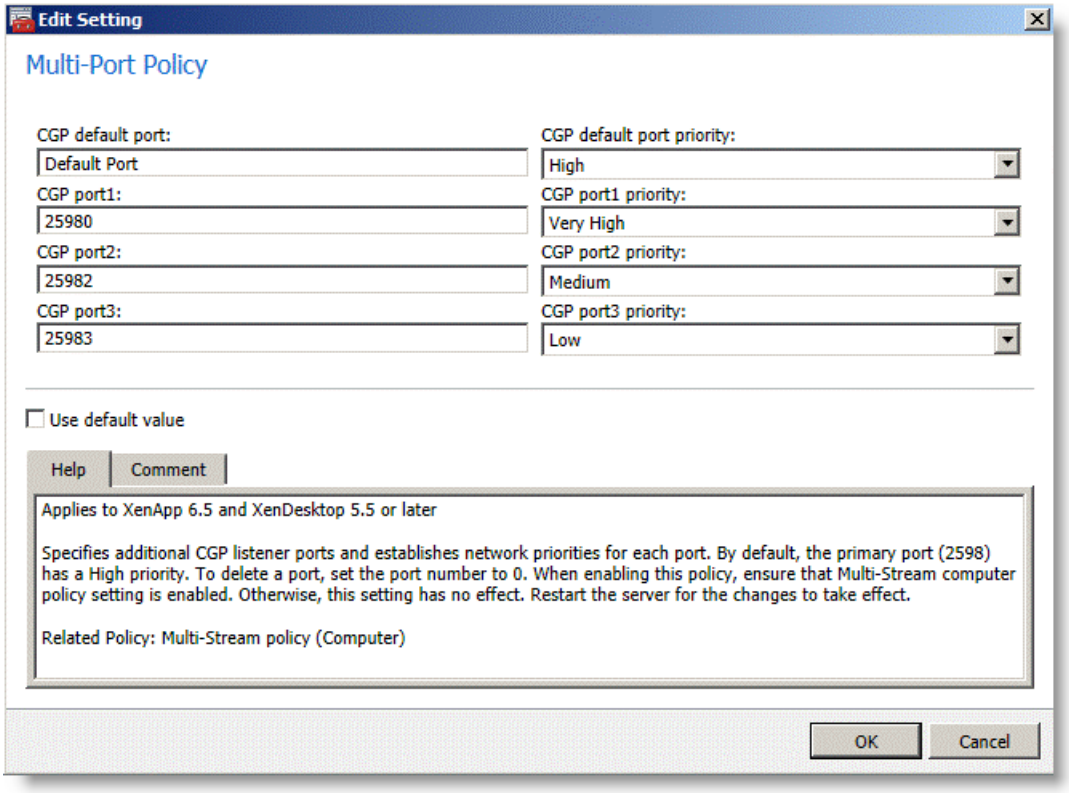
Figure 5-4. Citrix computer policy for multi-stream and multi-port ICA



2. Configure the multi-port policy in the Citrix computer policy.

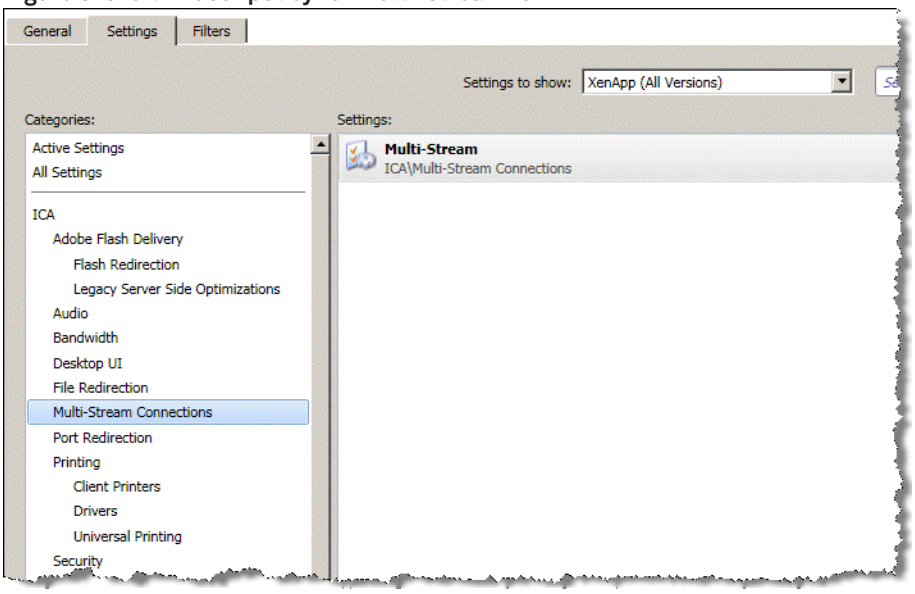
You cannot change the default port assigned to high priority. The default is mapped to the Citrix CGP port 2598 for session reliability.

Figure 5-5. Citrix computer policy for multi-port ICA



3. Enable multi-stream ICA in the Citrix user policy.

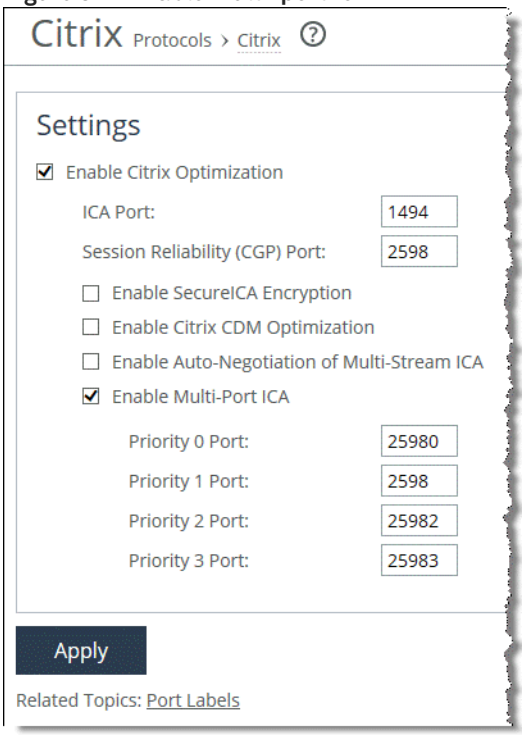
Figure 5-6. Citrix user policy for multi-stream ICA



4. Restart the XenApp Server to apply the Citrix computer policy. Log in the user once more to apply the Citrix user policy.

5. From the SteelHead Management Console, choose Optimization > Protocols: Citrix and specify the ports according to what is configured in the multi-port policy.

Figure 5-7. Enable multi-port ICA



6. Click **Apply** to apply the settings to the running configuration.

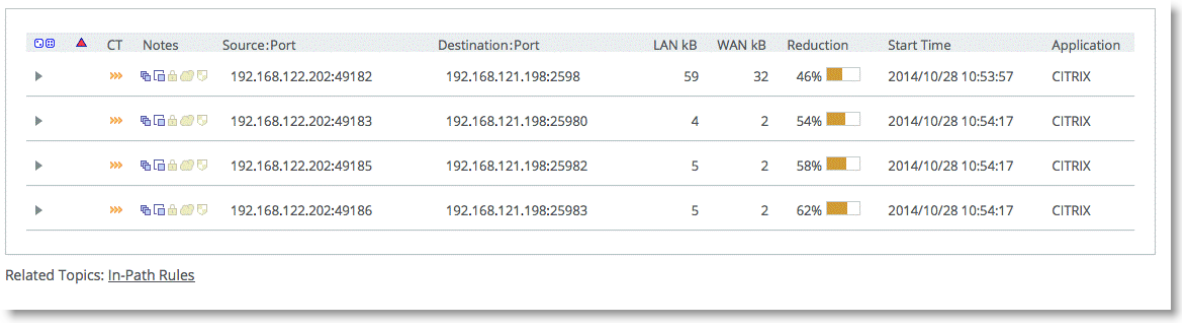
7. Restart the optimization service.

   You must repeat this configuration on both the client-side and the server-side SteelHead.

8. Log in to the XenApp server and launch a published application.

From the Current Connections page in the SteelHead Management Console, you see four unique TCP connections for the Citrix session, according to the TCP ports that you have configured in the multi-port policy.

Figure 5-8. Current Connections report



You do not need to change your current QoS configuration if you already have rules to prioritize Citrix ICA traffic on port 2598. However, you must configure additional QoS rules if you want to prioritize Citrix ICA traffic on the user-configurable ports that you have configured in the multi-port Citrix computer policy.

QoS classification for Citrix traffic

RiOS 6.0 and later enable you to classify Citrix traffic using QoS to prioritize the delivery of Citrix ICA traffic according to the ICA priority groups. QoS classification for Citrix traffic is beneficial in mixed-use environments in which Citrix users perform printing and use client drive-mapping features. Using QoS to classify Citrix traffic in a mixed-use environment improves the desktop computing experience for end users.

These RiOS QoS capabilities ensure optimal delivery of Citrix traffic over the network:

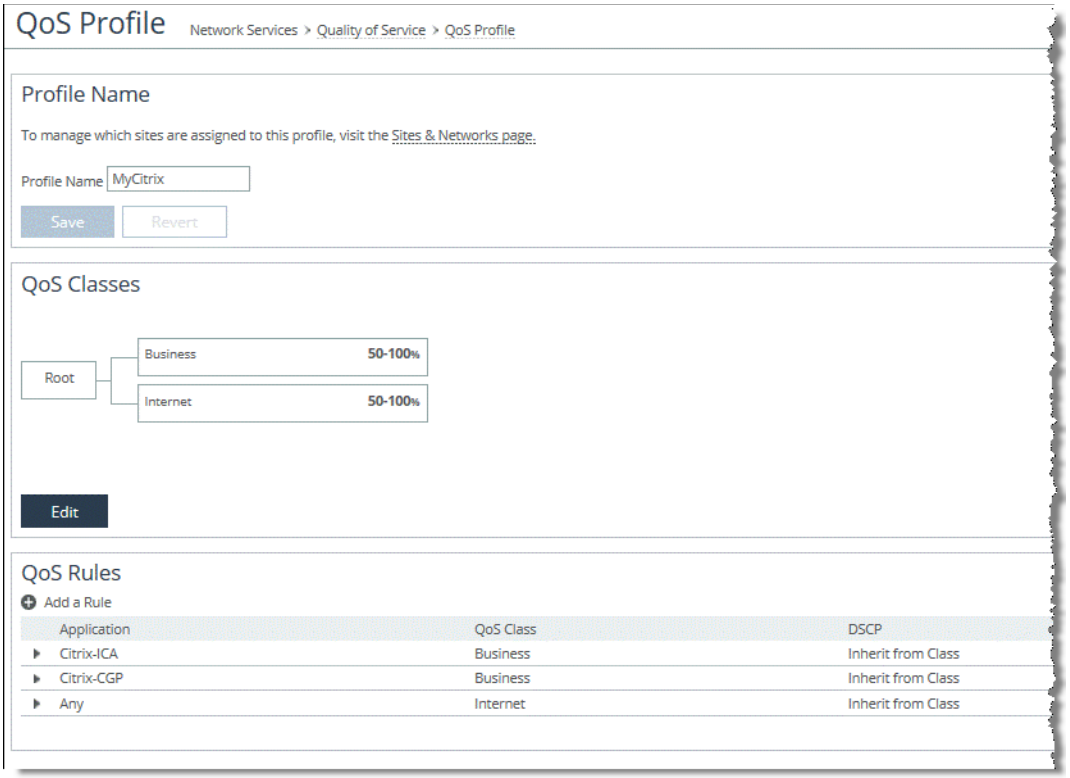
- **Latency priority** - Latency priority enables you to assign Citrix interactive traffic a higher priority than print or CDM traffic. A typical priority setting for interactive Citrix sessions, such as screen updates, is real-time or interactive. Remember that priority is relative to other classes in your QoS configuration.
- **Bandwidth allocation for traffic shaping** - When configuring QoS, you must allocate an appropriate amount of bandwidth for each QoS traffic class. The amount you specify is divided equally among all traffic flows with that class of traffic. Bandwidth allocation is important to ensure that a given traffic class has a minimum amount of bandwidth to perform on the network. While at the same time, the amount of bandwidth assigned to that traffic must not overrun the network and starve out other applications competing for network bandwidth.

For more information about configuring QoS, see the *SteelHead Deployment Guide*.

Note: RiOS 9.0 and later do not support Packet-Order Queue. We recommend that you use SFQ when you configure QoS for Citrix ICA traffic in RiOS 9.0 and later.

You can use the Riverbed Application Flow Engine (AFE) to classify Citrix ICA traffic. The AFE recognizes Citrix traffic on TCP port 1494 (Citrix-ICA) and port 2598 (Citrix-CGP). Use AFE to classify Citrix ICA traffic into a QoS class that is assigned with a higher priority than the QoS classes for other network traffic (Figure 5-9).

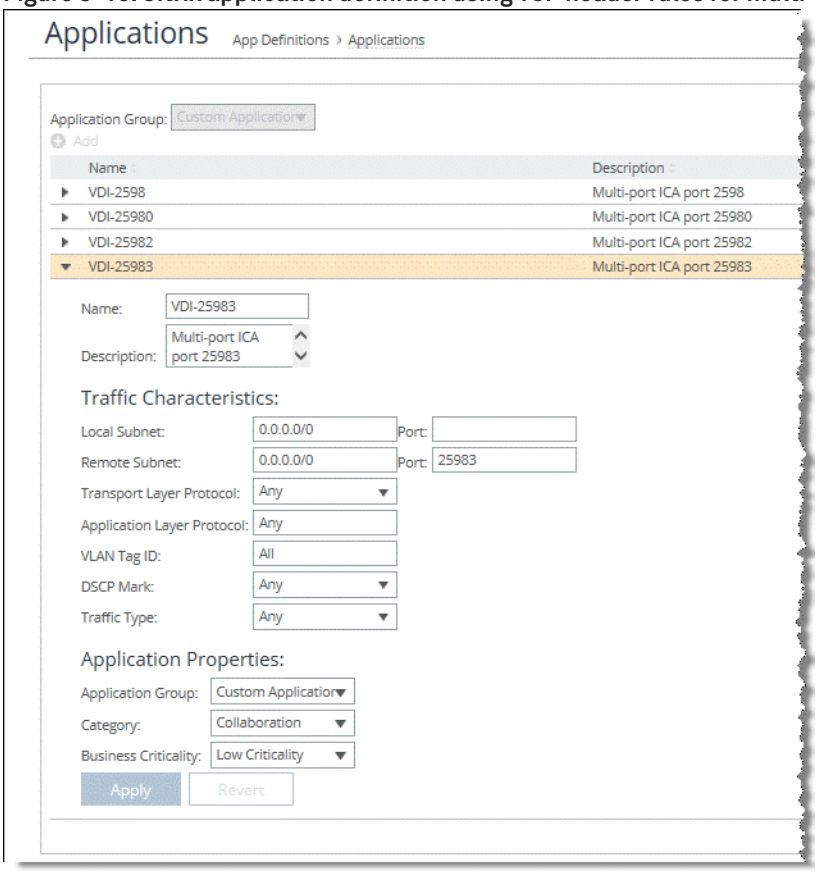
Figure 5-9. Classifying Citrix ICA traffic with AFE



To configure QoS for multi-stream and multi-port ICA traffic

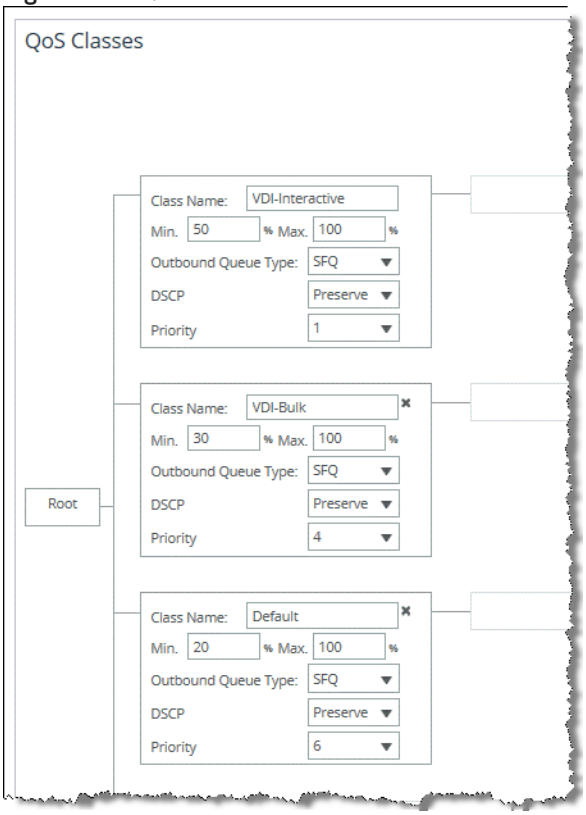
1. Define the Citrix application using TCP header rules in accordance to the TCP ports that you have configured in the multi-port Citrix computer policy. (Figure 5-10).

Figure 5-10. Citrix application definition using TCP header rules for multi-stream and multi-port ICA traffic



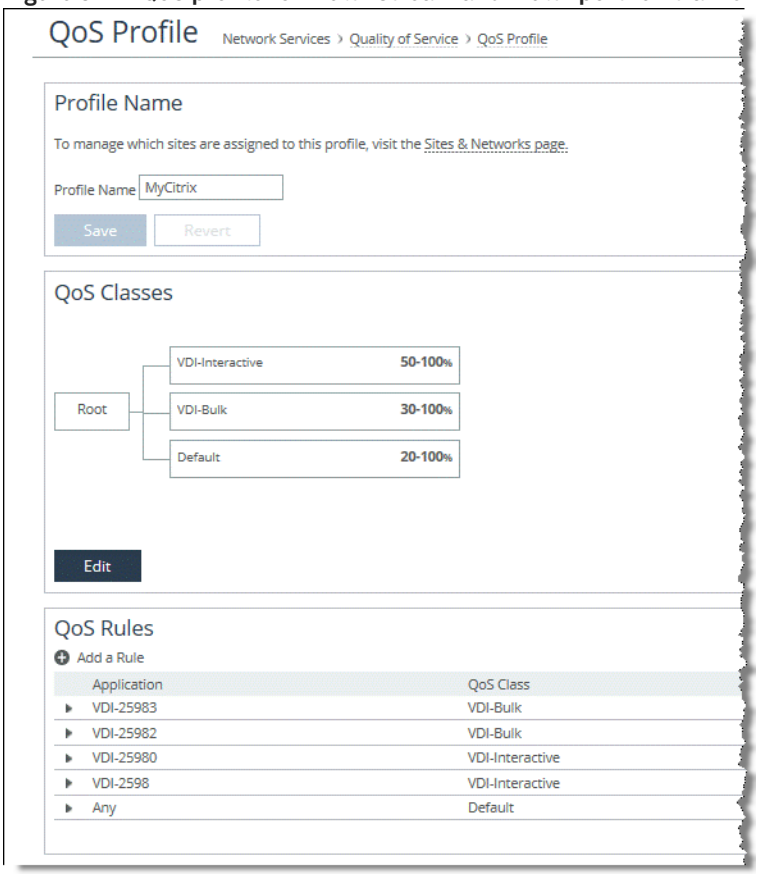
2. Define the QoS Class for Citrix interactive traffic, print traffic, and CDM traffic (Figure 5-11).

Figure 5-11. QoS class definition for multi-stream and multi-port ICA traffic



3. Define the QoS profile for multi-port ICA traffic using QoS rules that map the Citrix application into the appropriate QoS class (Figure 5-12).

Figure 5-12. QoS profile for multi-stream and multi-port ICA traffic



Automatic negotiation of multi-stream ICA traffic for QoS enforcement

RiOS 9.1 and later enable you to automatically configure and negotiate multi-stream ICA with the Citrix client from the client-side SteelHead and automatically use QoS to classify the negotiated multi-stream ICA traffic for QoS enforcement by the same client-side SteelHead. This configuration works with Citrix ICA traffic on both TCP port 1494 and 2598. This feature is independent of the Citrix computer policy for multi-stream and multi-port ICA on the Citrix server and the Citrix user policy for multi-stream ICA.

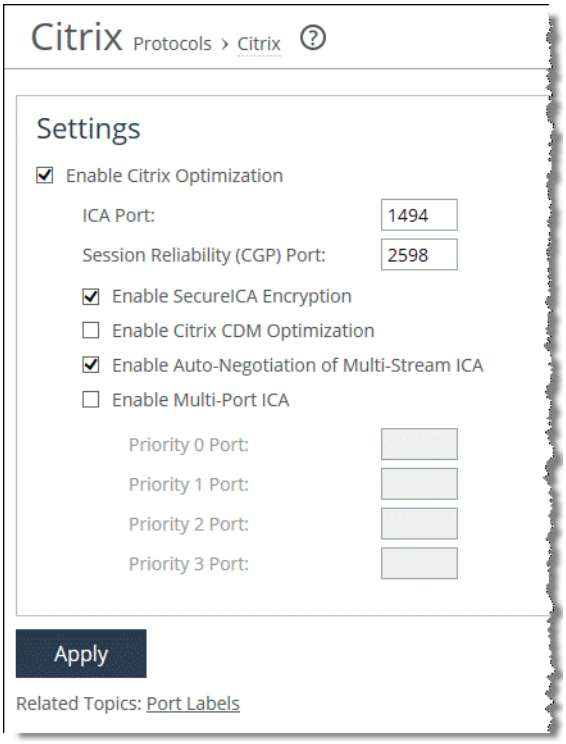
Note: Configure multi-stream ICA for network QoS enforcement of Citrix traffic from either the Citrix server using the Citrix multi-stream/multi-port computer and user policy or the client-side SteelHead using the SteelHead automatic negotiation of multi-stream ICA feature.

To enable RiOS to automatically negotiate multi-stream ICA on the client-side SteelHead

1. From the client-side SteelHead Management Console, choose Optimization > Protocols: Citrix.
2. Select Enable Auto-Negotiation of Multi-Stream ICA (Figure 5-13).

You do not need to enable this setting on the server-side SteelHead.

Figure 5-13. Enable auto-negotiation for multi-stream ICA



3. Click **Apply** to apply the settings to the running configuration.

   You do not need to restart the service.

4. Restart the client connection from Citrix Receiver.

The client-side SteelHead automatically negotiates multi-stream ICA with the Citrix client after it receives a connection for a new Citrix session. If the negotiation is successful, the Citrix client initiates four TCP connections to the Citrix server. Each connection represents an ICA priority group. RiOS automatically classifies the ICA priority group into an equivalent predefined Citrix application (Figure 5-14).

Figure 5-14. Citrix multi-stream application definition

Citrix-Multi-Stream-ICA-Priority-0	Citrix very high priority traffic for real-time traffic like audio.
Citrix-Multi-Stream-ICA-Priority-1	Citrix high priority traffic for interactive traffic like graphics, keyboard, and mouse.
Citrix-Multi-Stream-ICA-Priority-2	Citrix medium priority traffic for bulk traffic like drive mapping.
Citrix-Multi-Stream-ICA-Priority-3	Citrix low priority traffic for background traffic like printing.

Thereafter, you can configure an appropriate QoS profile for Citrix ICA traffic. This feature is relevant only if you are going to use QoS for Citrix ICA traffic.

For more details on configuring QoS for multi-stream ICA traffic, see the section "QoS classification for Citrix traffic" on page 117.

Note: Multi-stream ICA is supported on Citrix XenApp version 6.5, Citrix XenDesktop 5.5, and Receiver 3.0 and later (including online plug-in 13.0).

Reduction for Citrix small packet real-time traffic

We recommend that you enable enhanced data reduction for real-time Citrix traffic that is sent in small packets, such as keystrokes, mouse clicks, and other Citrix packets that are less than 64 bytes.

Optimization for small Citrix packets is disabled by default.

To enable or disable Citrix optimization for small packets, use the following command:

```
[no] protocol citrix smallpkts enable
```

To check whether or not optimization for small Citrix packets is enabled, use the following command:

```
show protocol citrix smallpkts
```

You must perform this configuration on both the client-side and server-side SteelHeads.

Citrix ICA optimization over SSL

Citrix Access Gateway (CAG) is an appliance that provides secure remote access to users of XenApp and XenDesktop over SSL VPN. CAG is also known as Access Gateway Enterprise Edition (AGEE) and Netscaler Gateway. CAG proxies the Citrix ICA traffic delivered from these applications and passes them securely over HTTPS or SSL to the end user.

Note: For more information about SSL, see "Configuring SSL optimization on SteelHeads" on page 174.

Figure 5-15. Citrix ICA client communication through a Citrix Access gateway

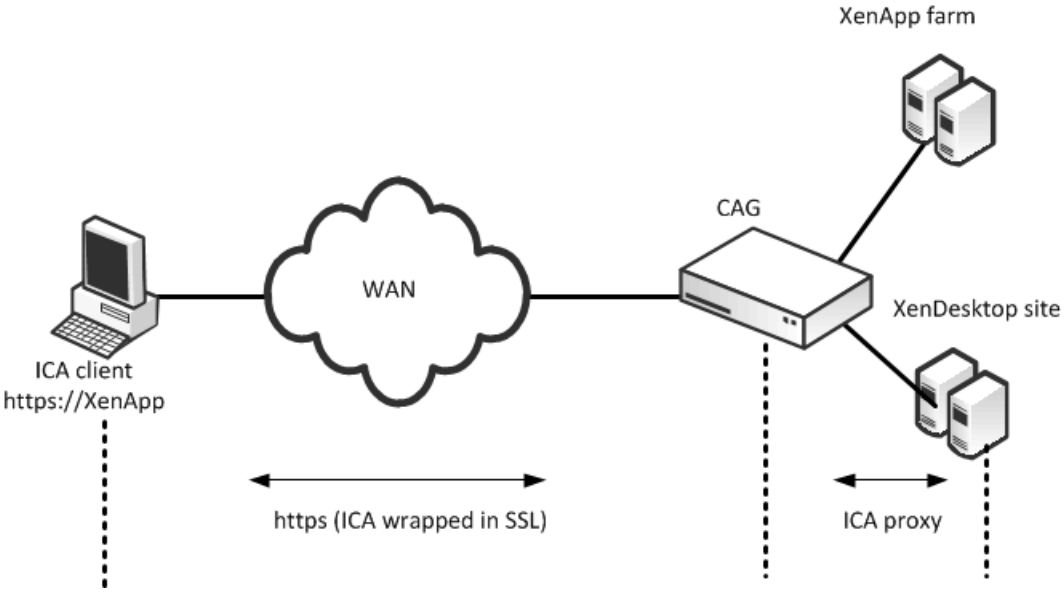


Figure 5-15 shows a CAG deployment. The user reaches the login page by entering the XenApp or XenDesktop secure remote access URL (https://<CAG URL>) in a browser. This page is hosted on the CAG. The user enters their credentials for authentication.

Upon a successful authentication, a list of published applications and desktops is displayed. When the user accesses these applications and desktops, an ICA connection is launched from the user desktop to the XenApp and XenDesktop server. The CAG functions as a gateway to intercept and proxy the user ICA connection to the XenApp and XenDesktop servers on one end, while providing secure remote access over SSL VPN to the user on the other end.

RiOS 7.0 and later can optimize ICA traffic wrapped in SSL using an SSL preoptimization policy to the in-path rule. The in-path rule has several parameters that allow for the chaining of multiple optimization features.

Figure 5-16 shows a SteelHead deployment with CAG.

Figure 5-16. SteelHead deployment with CAG

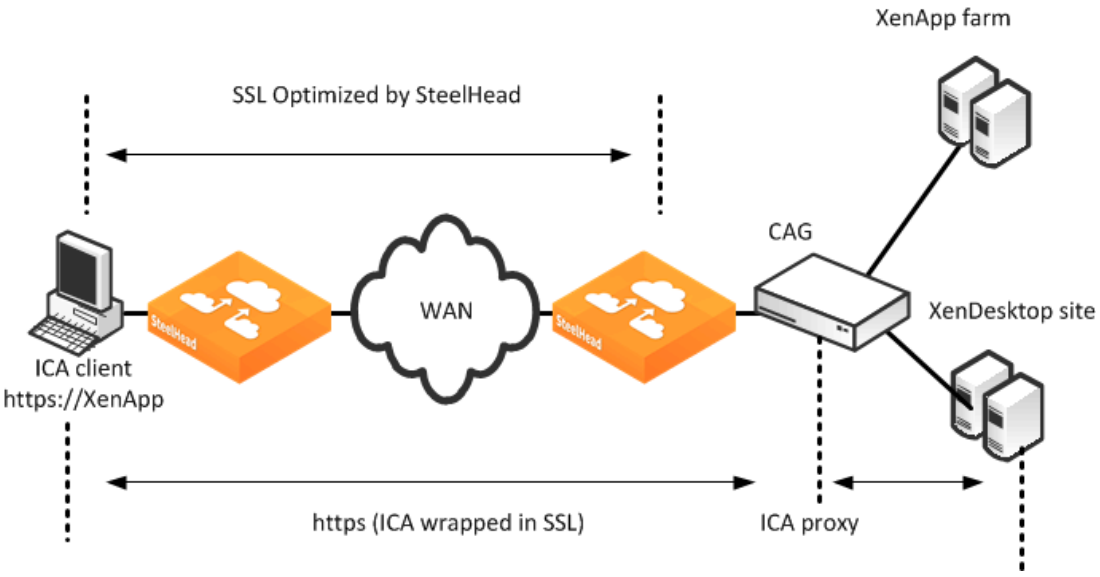
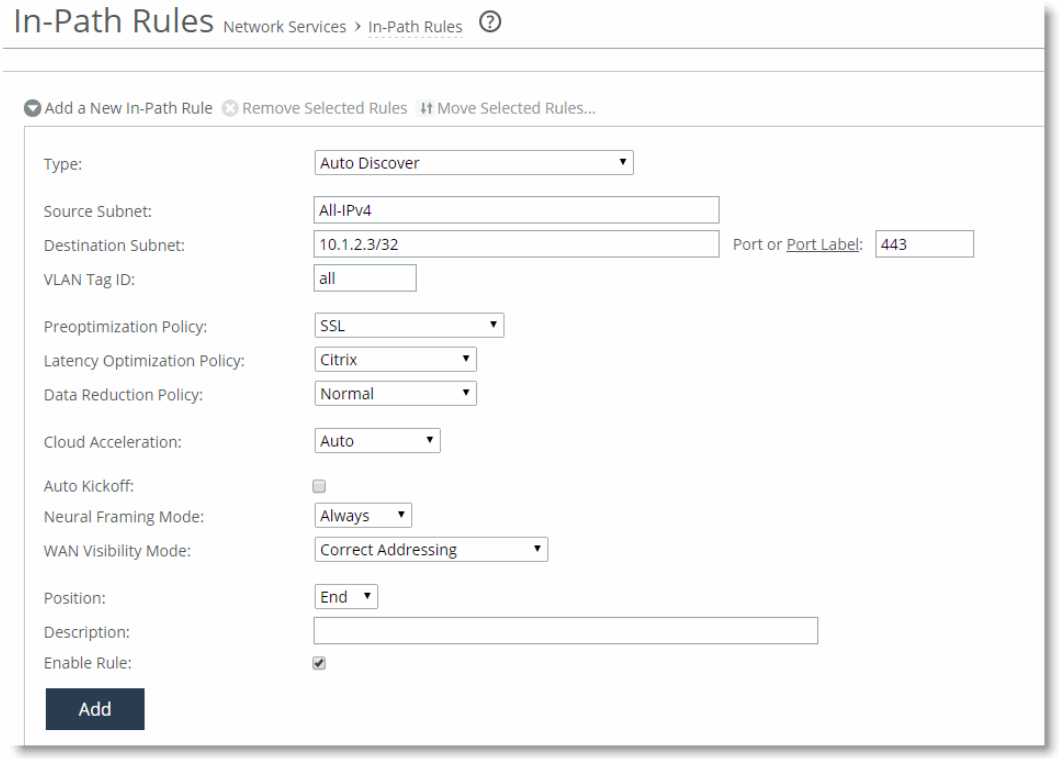


Figure 5-17 shows an in-path rule configuration to optimize Citrix ICA traffic wrapped in SSL.

Figure 5-17. SteelHead in-path rule to optimize Citrix ICA traffic through a CAG



Citrix ICA traffic optimization with CAG has the following requirements:

- Both the client-side and server-side SteelHead must have RiOS 7.0 or later.
- The proxy certificate you use on the SteelHead must be a valid and trusted certificate.

  If you use a self-signed proxy certificate, you must install it to the client's trusted root certificate authority certificate store. The Citrix client does not connect if you use an invalid or untrusted certificate.

6

Secure SMTP Optimization

In RiOS 7.0 and later, you can securely communicate over the internet with Secure Simple Mail Transfer Protocol (SMTPS). This chapter includes the following sections:

- "Overview of SMTP and TLS" on page 127
- "Configuring Microsoft Exchange servers for secure SMTP" on page 128
- "Configuring the SteelHead for START-TLS support" on page 130

Overview of SMTP and TLS

SMTP is the standard for email transport across the internet and the standard communication method for Microsoft Exchange hub servers in Exchange 2007 to Exchange 2013. Prior to RiOS 7.0, SMTP or SMTPS was sent through the SteelHead as a pass-through connection. Because the SMTP session was set up prior to the encrypted session, you could not determine when to start SSL traffic optimization.

RiOS 7.0 and later support StartTLS for Transport Layer Security (TLS) and Secure Socket Layer (SSL) to determine the start of an SMTPS connection and the early finish that enables the setup of encrypted (TLS or SSL) connection to terminate without disconnecting the SMTP session. This functionality provides more efficient use of the protocol, because it does not perform a handshake on every email sent using SMTPS.

For more information about SSL, see "SSL Deployments" on page 169. For more information about delayed Start TLS, see "Mid-session SSL support" on page 190.

Figure 6-1. SMTP connection

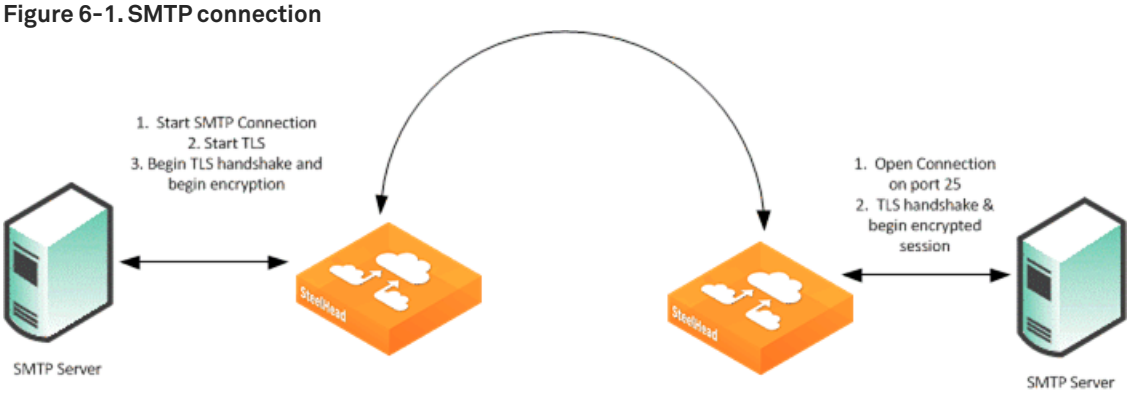


Figure 6-1 shows what happens when the **START-TLS** command is issued after the initial SMTP connection is established. Prior to RiOS 7.0, there was no way to determine that an encrypted session had started, so traffic optimization results on the port were low. It was common to enter a pass-through rule for this connection to prevent unnecessary use of RiOS resources.

In RiOS 7.0 and later, support for Secure SMTP enables the SteelHead to intercept the START-TLS message after it is issued, which enables the SteelHead to optimize native Secure SMTP traffic.

Figure 6-2. Secure SMTP connection setup using TLS

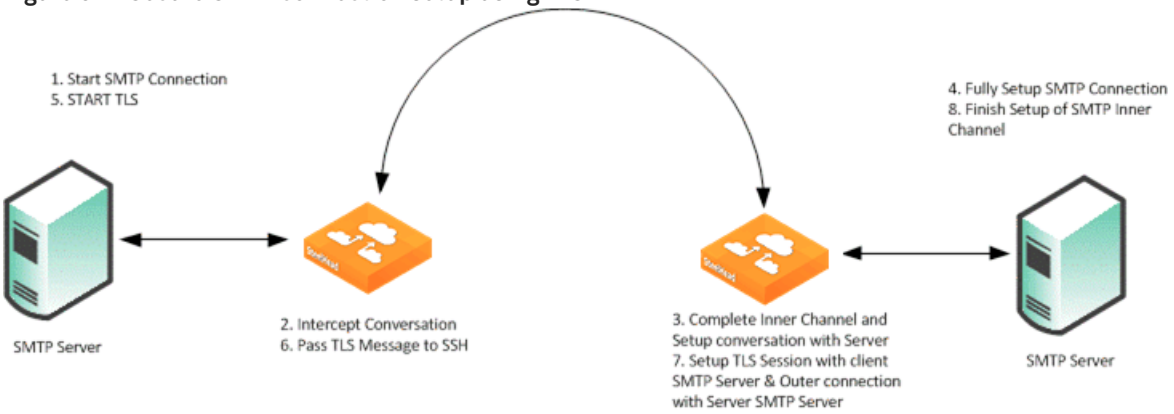


Figure 6-2 shows the setup of an SMTPS session. First, an unencrypted SMTP session is intercepted and optimized through the SteelHead. When the START-TLS is issued, RiOS recognizes that a secure session is imminent and begins the necessary operations to optimize the SSL or TLS session. For more information about SSL optimization, see "SSL Deployments" on page 169.

Note: Despite having TLS in the name, START-TLS does not mean TLS is necessarily used. Both SSL and TLS are acceptable protocols for securing the communication.

In RiOS 9.2 and later, TLSv1.2 support is enabled by default. A software update from an earlier RiOS version to 9.2 or later automatically enables TLSv1.2 support. To display TLS configuration status for a client or server, use the CLI command **show protocol ssl backend**. Use the CLI command **show secure-peering** for peering interface configuration.

Configuring Microsoft Exchange servers for secure SMTP

Microsoft Exchange hub servers have additional authentication mechanisms beyond TLS or SSL. Two specific environments require additional configuration on the Exchange hub server to allow the Secure SMTP optimization to function properly:

- **Single domain multiple hub servers** - All Exchange hub servers are within the same domain. The servers are inherently trusted, no specific SSL client authorization occurs, and the Exchange hub server uses a null client certificate.
- **Multiple domain multiple hub servers** - Exchange hub servers are in untrusted domains and must be trusted through the Microsoft **set-sendconnector** command. Complete the following steps on each Exchange hub server:

```
set-sendconnector -Identity "<send-conn-name>" -DomainSecurityEnabled $True
set-receiveconnector -Identity "<receive-conn-name>" -DomainSecurity Enabled $True
```

Next, set the secure domains to secure traffic between one another:

```
set-transportconfig -TLSReceiveDomainSecureList {<remote-domain>}
set-transportconfig -TLSSendDomainSecureList {<remote-domain>}
```

Some Exchange deployments require a domain trust before mutual authentication will work. Adding a two-way forest trust, preferably with forest-wide authentication, solves this issue.

Configuration with Microsoft Exchange Hub servers requires the following prerequisites:

- You must have an SSL license installed on all SteelHeads participating in SSL or TLS optimization.
- The client-side and server-side SteelHeads must have RiOS 7.0 or later.
- You must generate and import new certificates in the Exchange hub servers.

An Exchange hub server does not allow its private key to be exported. You cannot retrieve either the certificate or the private key from the hub server. For details regarding receive connectors and certificates in an Exchange hub server environment, go to the following websites:

- http://technet.microsoft.com/en-us/library/aa996395.aspx
- http://technet.microsoft.com/en-us/library/aa998327.aspx

You can generate the new certificates with the tool listed in Microsoft Technote 998327 or with OPENSSL. This is an example with OPENSSL:

```
linux#openssl req -x509 -newkey 1024 -keyout my.pem -out my.cert -nodes
linux#openssl pkcs12 -export -inkey my.pem -in my.cert -out my.pfx
```

After you create the new self-signed certificates, you must install them on the remote and local Exchange server—which runs the hub server role—as defined next.

To install a new certificate on the remote and local Exchange server

1. Import the PFX file to the local Exchange server's personal certificate store.
2. Add the SMTP role to the certificate.

   Exchange uses roles to identify which certificates to use in various situations. You need to *tell* the Exchange server that your certificate for encrypted traffic over SMTP.
3. If the Exchange management console (EMC) is not already open, in the search field of the Windows Start menu, search for Exchange management console and open the application.
4. In the left navigation pane, toggle open Microsoft Exchange On-Premises (your server).
5. Click **Server Configuration** and wait for the Exchange certificates in the work (that is, the middle) pane to populate.
6. To open the Assign Services to Certificate wizard, right click the self-signed certificate and select Assign Services to Certificate.

   You can identify your certificate by viewing the Subject and Issuer fields. Your ticket is probably the only one listed that does not have a customized name in the Name field.
7. Select Simple Mail Transfer Protocol (SMTP).

8. Click **Assign**.
9. After the assignment completes successfully, click **Finish**.

   If you are prompted to overwrite the existing SMTP certificate, select Yes.
10. Add the partner's permission group to the local Exchange server's default receive connector.
11. If the EMC is not already open, in the search field of the Windows Start menu, search for exchange management console and open the application.
12. In the left navigation pane, toggle open the Microsoft Exchange On-Premises (your server).
13. Toggle open the Server Configuration.
14. Click **Hub Transport** and wait for the receive connectors in the work pane (that is, the middle) to populate.
15. Click the default receive connection (usually named something similar to Default mach101).
16. In the Actions pane, under the section with the default receive connector's name, select Properties. This opens the default receive connector's properties dialog box.
17. Select the Permission Groups tab.
18. Select Partners.

    Clear the Anonymous users check box if it is selected.
19. Click **OK** to accept the changes to the default receive connector's properties.
20. Import the CERT file to the remote Exchange server's Computer Trusted Root Certificate Authorities.

Configuring the SteelHead for START-TLS support

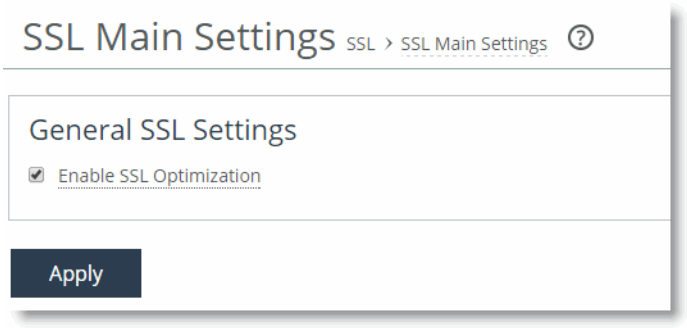
This section describes how to configure the SteelHead to use START-TLS.

To configure the SteelHead for use with START-TLS

1. Enable SSL on the SteelHead.

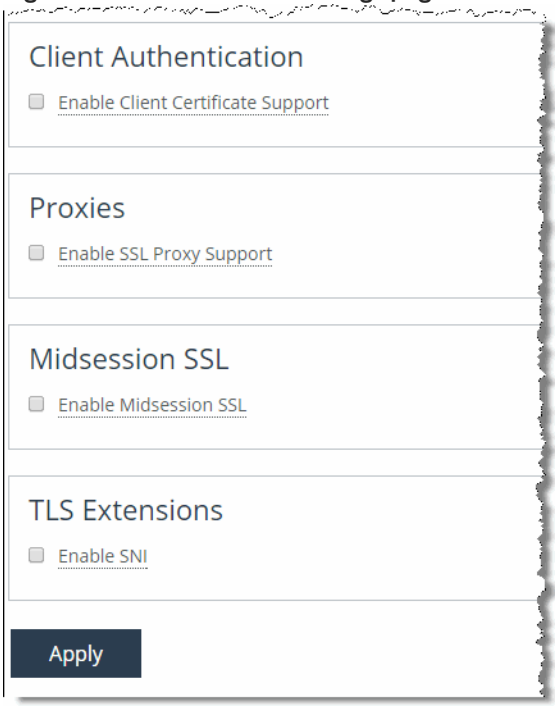
- In the Management Console, choose Optimization > SSL: SSL Main Settings, and select Enable SSL Optimization, or use the **protocol ssl enable** command.

Figure 6-3. SSL Main Settings page



- In the Management Console, choose Optimization > SSL: Advanced Settings and select Enable Client Certificate Support and Enable Midsession SSL, or use the **protocol ssl mid-session-ssl** and **protocol ssl client-cer-auth enable** commands.

Figure 6-4. SSL Advanced Settings page



2. Configure an in-path rule.

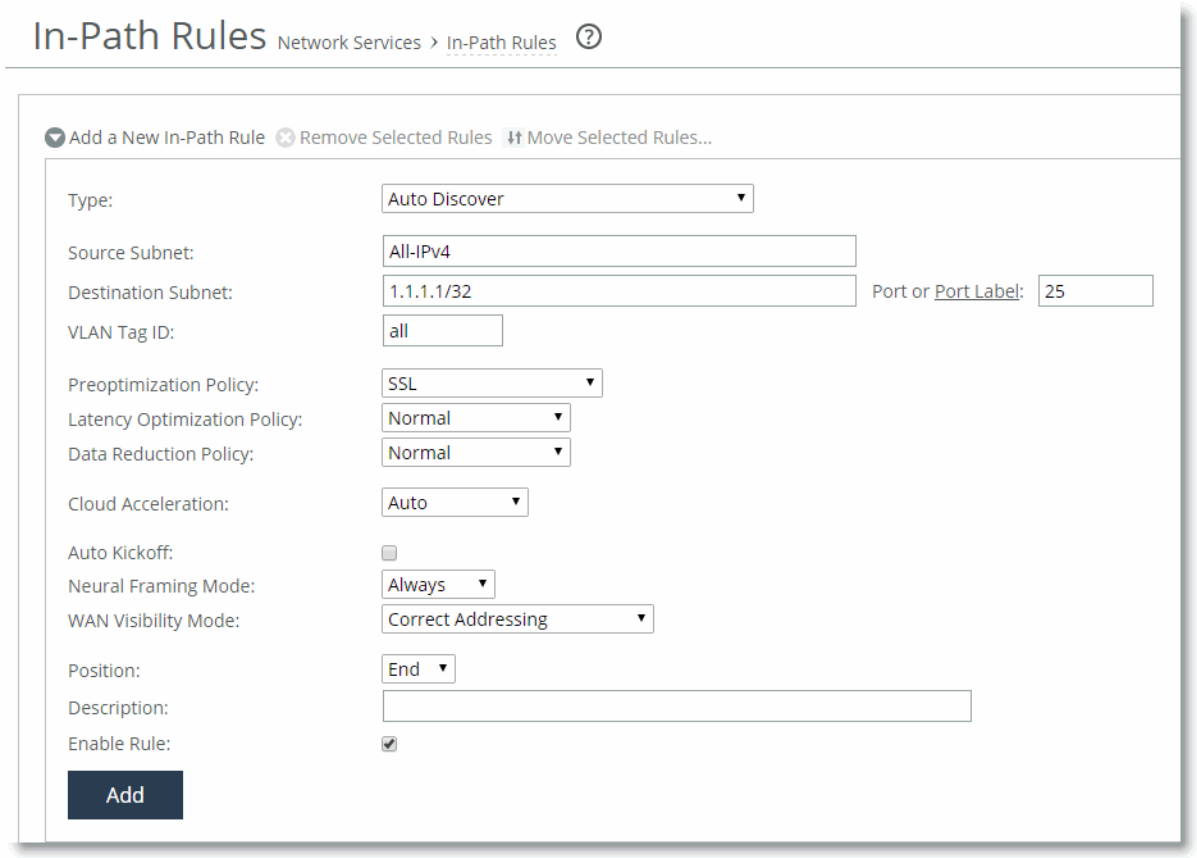
   The in-path rule applies the SSL preoptimization policy to the Secure SMTP traffic.

   - In the Management Console, choose Optimization > Network Services: In-Path Rules.
   - Select Add New In-Path Rule, enter the following parameters, and click **Add**.

- Or, you can use the **in-path rule auto-discover rulenum 4 srcaddr all-ipv4 dstaddr 1.1.1.1/32 dstport 25 preoptimization ssl** command.

Rule setting	Value
Type	Auto Discover
Source Subnet	All-IPv4
Destination Subnet	IP address of the email server using SMTPS
Port or Port Labels	SMTP port, typically 25
VLAN tag ID	all
Preoptimization Policy	SSL
Latency Reduction Policy	Normal
Data Reduction Policy	Normal
Auto Kickoff	Clear the check box
Neural Framing	Always
WAN Visibility	Correct Addressing, Port Transparency, or Full Transparency are all valid configuration options.
Position	Prior to any negating rules
Description	A description of the rules
Enable Rules	Select the check box

Figure 6-5. In-Path Rules page



7

FTP Optimization

This chapter discusses File Transfer Protocol (FTP) behavior and configuration on the SteelHead, including how to configure in-path and QoS rules to accomplish specific functions. This chapter includes the following sections:

- "Overview of FTP" on page 135
- "Configuring in-path rules" on page 137
- "QoS classification for the FTP data channel" on page 138
- "FTP optimization considerations" on page 139
- "SteelCentral Controller for SteelHead Mobile FTP considerations" on page 140

FTP is a standard network protocol used to copy a file from one host to another over a TCP/IP-based network. FTP is built on a client-server architecture and uses separate control and data connections between the client and server.

By default, FTP optimization is enabled on all FTP connections.

Overview of FTP

The FTP protocol consists of two connections: the control connection and the data connection. A client initiates the control connection to the server on TCP port 21. This connection remains open for the duration of the session and sends administrative data (for example, commands, identification, and passwords). After the control connection is established, the data connection transfers the file data.

The data connection uses various originators and ports. When you look at TCP connections and SteelHead optimization, both the control and data connections appear as separate TCP connections. The control connection always appears to go from the FTP client on a random source port to the FTP server on port 21. The data connection properties change significantly, depending on whether active or passive mode is used for the FTP transfer.

Active mode

Figure 7-1 shows the control connection and data connection used in FTP active mode:

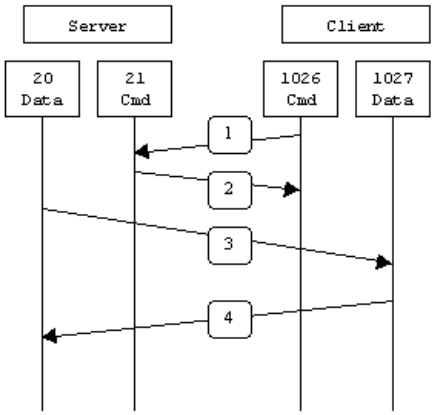
1. The client from a random TCP source port greater that 1024 (shown here as port 1026) connects to the server on port 21 to establish the control connection. The client sends the server the port to establish the data connection on (shown here as TCP port 1027).

2. The server acknowledges the port number.

3. The server initiates the data connection from TCP port 20 to the client on the specified port in Step 1.
4. The client responds with an ACK to complete the establishment of the data connection.

Most of the time TCP port 21 is associated with FTP control and TCP port 20 with data. TCP port 20 is only true with FTP in active mode.

Figure 7-1. Active mode



Source: http://slacksite.com/other/ftp.html (Sept 8, 2010)

A potential issue with active mode FTP is the FTP client does not make the actual connection to the data port of the server—it tells the server what port it is listening on and the server connects back to the specified port on the client. From a client-side firewall this appears to be an outside system initiating a connection to an internal client, which can be blocked. For more information, see the *SteelHead Deployment Guide*.

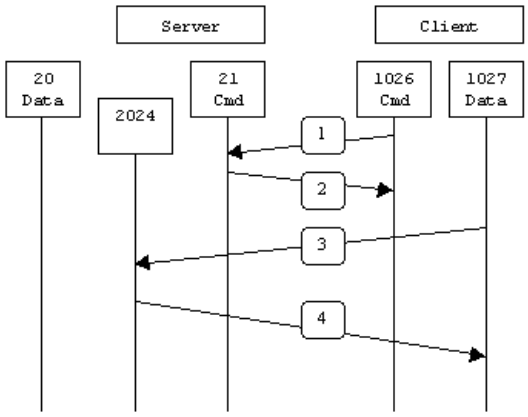
Passive mode

Figure 7-2 shows the control connection and data connection used in FTP passive mode.

1. The client connects from a random TCP source port to the server on port 21. The client requests the server to provide a port that the client can connect to for the data connection.
2. The server replies with the specified port for the data channel (shown here as TCP port 2024).
3. The client initiates the data connection from a random port to the specified server data port.
4. The server sends an ACK to the client.

With passive FTP, both the control and data connections are both originated from the client and that port 20 is not used.

Figure 7-2. Passive mode



Source: http://slacksite.com/other/ftp.html (Sept 8, 2010)

Passive mode FTP solves many of the problems from the client-side security perspective, although it does require the server to accept a remote connection to a range of high numbered ports. Most clients today support both active and passive mode FTP. The default Windows client does not support passive FTP, along with some Unix versions, such as Solaris.

Many people prefer to use their web browser as an FTP client. Most browsers only support passive mode when accessing ftp:// URLs.

Configuring in-path rules

SteelHeads are aware of FTP connections in either active or passive mode, and they correlate the data channels with the control channel.

Optimizing FTP

By default, FTP uses the standard optimization rule for in-path rules. If you use manual optimization rules, follow the standard in-path practice, configuring the client-side SteelHead in-path rules based on IPs or TCP port. With the TCP port, it is only necessary you specify destination port 21, which automatically includes optimization for the data channels, regardless of whether active or passive mode is used.

Passing through FTP

To pass through FTP, we recommend creating pass-through rules based on the FTP server IP, on both the client-side and server-side SteelHeads. You base the rules this way because of varying data connection behavior depending on active or passive mode. Because the source of the data connection is the FTP server in active mode, it is necessary to configure the in-path pass-through rules on the server-side SteelHead—this configuration is unlike other protocols.

QoS classification for the FTP data channel

When configuring QoS classification for FTP, the QoS rules differ depending on whether the FTP data channel is using *active* or *passive* FTP. Active versus passive FTP determines whether the FTP client or the FTP server select the port connection for use with the data channel, which has implications for QoS classification.

Active FTP classification

With active FTP, the FTP client logs in and enters the PORT command, informing the server which port it must use to connect to the client for the FTP data channel. Next, the FTP server initiates the connection towards the client. From a TCP perspective, the server and the client swap roles. The FTP server becomes the client because it sends the SYN packet, and the FTP client becomes the server because it receives the SYN packet.

Although not defined in the RFC, most FTP servers use source port 20 for the active FTP data channel.

For active FTP, configure a QoS rule on the server-side SteelHead to match source port 20. On the client-side SteelHead, configure a QoS rule to match destination port 20.

You can also use the Application Flow Engine (AFE) to classify active FTP traffic.

Passive FTP classification

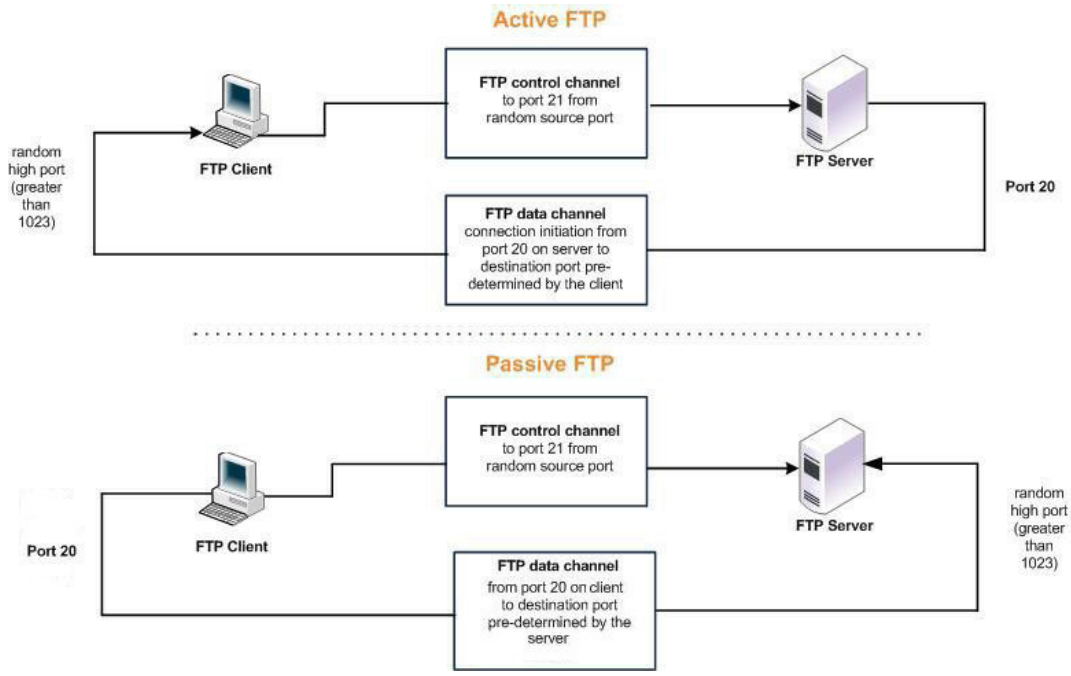
With passive FTP, the FTP client initiates both connections to the server. First, it requests passive mode by entering the PASV command after logging in. Next, it requests a port number for use with the data channel from the FTP server. The server agrees to this mode, selects a random port number, and returns it to the client. Once the client has this information, it initiates a new TCP connection for the data channel to the server-assigned port. Unlike active FTP, there is no role swapping and the FTP client initiates the SYN packet for the data channel.

The FTP client receives a random port number from the FTP server. Because the FTP server cannot return a consistent port number to use with the FTP data channel, RiOS does not support QoS Classification for passive FTP in versions earlier than RiOS 4.1.8, 5.0.6, or 5.5.1. Later RiOS releases support passive FTP and the QoS Classification configuration for passive FTP is the same as active FTP.

When configuring QoS Classification for passive FTP, port 20 on both the server-side and client-side SteelHeads means the port number is being used by the data channel for passive FTP, as opposed to the literal meaning of source or destination port 20.

Note: The SteelHead must intercept the FTP control channel (port 21), regardless of whether the FTP data channel is using active or passive FTP.

Figure 7-3. Active and passive FTP



With RiOS 8.0.4 and later, the AFE monitors the FTP control connection. AFE learns the negotiated port numbers and connection initiator from the FTP control connection. You can then use AFE to classify active and passive FTP connections for IPv4.

The AFE is unable to classify FTP correctly in a server-side out-of-path (SSOOP) SteelHead deployment with the exception of optimized FTP in active mode.

Note: FTP with IPv6 is currently not supported in AFE and QoS.

For more information about QoS and AFE, see the *SteelHead Deployment Guide* and the *SteelHead User Guide*.

FTP optimization considerations

Although the FTP data and control connections are separate TCP connections, the two channels are correlated. For example, the sender requests to close the data connection over the control channel when a file has finished transmitting.

With WAN optimization in place, the sender can incorrectly believe that the file transfer has complete and send a request over the control channel to prematurely close the data connection. Because the SteelHead acts as a TCP proxy, the sender is actually communicating with the local SteelHead. In certain environments with high latency or low data reduction, the local SteelHead can still be sending the file, but the sender sends the request to close the data connection before the file is completely finished.

If you find that the data connection is closing prematurely, you can adjust a manual delay to the sender request. Use the **protocol ftp delay-close-sec <time-in-seconds>** command on the SteelHead local to the FTP server. We recommend 30 seconds for most environments.

SteelCentral Controller for SteelHead Mobile FTP considerations

SteelCentral Controller for SteelHead Mobile only optimizes connections originating from the SteelHead Mobile. With active FTP, the data connection is opened from the server to the client. SteelCentral Controller for SteelHead Mobile does not optimize active FTP because the data connection originates from the server. Although the majority of FTP client software supports passive mode, ensure your client is capable and configured correctly to use passive FTP when using SteelCentral Controller for SteelHead Mobile.

For more information about SteelCentral Controller for SteelHead Mobile, see the *SteelHead Deployment Guide*.

8

Other Protocol Optimization

In addition to the protocols previously discussed, this chapter describes the basic steps for configuring SteelHead protocol optimization for the following protocols:

- "Oracle Forms optimization" on page 141
- "NFS optimization" on page 142
- "Lotus Notes optimization" on page 144

Oracle Forms optimization

You can display and modify Oracle Forms optimization settings in the Configure > Optimization > Oracle Forms page.

Oracle Forms is a platform for developing user interface applications to interact with an Oracle database. It uses a Java applet to interact with the database in either native, HTTP, or HTTPS mode. The SteelHead decrypts, optimizes, and then reencrypts the Oracle Forms traffic.

You can configure Oracle Forms optimization in these modes:

- **Native** - The Java applet communicates with the backend server, typically over port 9000. Native mode is also known as socket mode.
- **HTTP** - The Java applet tunnels the traffic to the Oracle Forms server over HTTP, typically over port 8000.
- **HTTPS** - The Java applet tunnels the traffic to the Oracle Forms server over HTTPS, typically over port 443. HTTPS mode is also known as SSL mode.

Use Oracle Forms optimization to improve Oracle Forms traffic performance. RiOS 5.5.x and later support 6i, which comes with Oracle Applications 11i. RiOS 6.0 and later support 10gR2, which comes with Oracle E-Business Suite R12.

This feature does not need a separate license and is enabled by default. However, you must also set an in-path rule to enable this feature.

Note: Optionally, you can enable IPSec encryption to protect Oracle Forms traffic between two SteelHead appliances over the WAN or use the secure inner channel on all traffic.

Determining the deployment mode

Before enabling Oracle Forms optimization, you must know the mode in which Oracle Forms is running at your organization.

To determine the Oracle Forms deployment mode

1. Start the Oracle application that uses Oracle Forms.
2. Click a link in the base HTML page to download the Java applet to your browser.
3. On the Windows task bar, right-click the Java icon (a coffee cup) to access the Java console.
4. Choose Show Console (Initiator) or Open <version> Console (Sun JRE).
5. Locate the “connectMode=” message in the Java Console window. This message indicates the Oracle Forms deployment mode at your organization: for example,

```
connectMode=HTTP, native
connectMode=Socket
connectMode=HTTPS, native
```

For more information about configuring Oracle Forms optimization, see the *SteelHead User Guide*.

NFS optimization

NFS optimization provides latency optimization improvements for NFS operations by prefetching data, storing it on the client SteelHead for a short amount of time, and using it to respond to client requests. You enable NFS optimization in high-latency environments.

You can configure NFS settings globally for all servers and volumes or you can configure NFS settings that are specific to particular servers or volumes. When you configure NFS settings for a server, the settings are applied to all volumes on that server unless you override settings for specific volumes.

Note: NFS optimization is not supported in an out-of-path deployment.

Note: NFS optimization is only supported for NFSv3.

For each SteelHead, you specify a policy for prefetching data from NFS servers and volumes. You can set the following policies for NFS servers and volumes:

- **Global Read/Write** - Choose this policy when the data on the NFS server or volume can be accessed from any client, including LAN clients and clients using other file protocols. This policy ensures data consistency but does not allow for the most aggressive data optimization. Global Read/Write is the default value.
- **Custom** - Create a custom policy for the NFS server.
- **Read-only** - Any client can read the data on the NFS server or volume but cannot make changes.

After you add a server, the Management Console includes options to configure volume policies.

For detailed information, see the *SteelHead User Guide*.

Implementing NFS optimization

This section describes the basic steps for using the Management Console to implement NFS. For detailed information, see the *SteelHead User Guide*.

Basic steps

Perform the following basic steps to configure NFS optimization.

To configure NFS optimized connections

1. Enable NFS in the Optimization > Protocols: NFS page.

   Enable NFS on all desired client and server SteelHeads.

2. For each client SteelHead, configure NFS settings that apply by default to all NFS servers and volumes. For details, see the *SteelHead User Guide*.

   Configure these settings on all desired client SteelHeads. These settings are ignored on server-side SteelHeads. If you have enabled NFS optimization (as described in the previous step) on a server-side SteelHead, NFS configuration information for a connection is uploaded from the client-side SteelHead to the server SteelHead when the connection is established.

Note: If NFS is disabled on a server-side SteelHead, the appliance does not perform NFS optimization.

3. For each client-side SteelHead, override global NFS settings for a server or volume that you specify. You do not need to configure these settings on server-side SteelHeads. If you have enabled NFS optimization on a server-side SteelHead, NFS configuration information for a connection is uploaded from the client-side SteelHead to the server-side SteelHead when the connection is established.

   If you do not override settings for a server or volume, the global NFS settings are used. If you do not configure NFS settings for a volume, the server-specific settings, if configured, are applied to the volume. If server-specific settings are not configured, the global settings are applied to the server and its volumes.

   When you configure a prefetch policy for an NFS volume, you specify the desired volume by an FSID number. An FSID is a number NFS uses to distinguish mount points on the same physical file system. Because two mount points on the same physical file system have the same FSID, more than one volume can have the same FSID.

   For details, see the *SteelHead User Guide*.

4. If you have configured IP aliasing for an NFS server, specify all of the server IP addresses in the SteelHead NFS-protocol settings.

5. View and monitor NFS statistics in the Management Console Reports > Optimization: NFS Statistics.

Configuring IP aliasing

If you have configured IP aliasing (multiple IP addresses) for an NFS server, you must specify all of the server IP addresses in the SteelHead NFS protocol settings for NFS optimization to work properly.

To configure IP aliasing on a SteelHead

1. In the Management Console, choose Optimization > Protocols: NFS.

2. Select **Add New NFS Server** to expand the page.

3. In the Name box, specify the name of the NFS server.
4. Enter each server IP address in a comma-separated list in the Server IP box.
5. Click **Add**.

Lotus Notes optimization

You can enable and modify Lotus Notes optimization settings in the Configure > Optimization > Lotus Notes page.

Lotus Notes is a client-server collaborative application that provides email, instant messaging, calendar, resource, and file sharing. RiOS provides latency and bandwidth optimization for Lotus Notes 6.0 and later traffic across the WAN, accelerating email attachment transfers and server-to-server or client-to-server replications.

RiOS saves bandwidth by automatically disabling socket compression, which makes SDR more effective. RiOS also saves bandwidth by decompressing Huffman-compressed attachments and LZ-compressed attachments when they are sent or received and recompressing them on the other side. This process allows SDR to recognize attachments that have previously been sent in other ways (such as over CIFS, HTTP, or other protocols), and also allows SDR to optimize the sending and receiving of attachments that are slightly changed from previous sends and receives.

To use this feature, both the client-side and server-side SteelHeads must be running RiOS 5.5.x or later. To enable optimization of encrypted Lotus Notes connections, both the client-side and server-side SteelHeads must be running RiOS 7.0 or later.

Enabling Lotus Notes provides latency optimization regardless of the compression type (Huffman, LZ, or none).

Before enabling Lotus Notes optimization, be aware that it automatically disables socket-level compression for connections going through SteelHeads that have this feature enabled.

For information about configuring Lotus Notes optimization, see the *SteelHead User Guide*.

Optimizing encrypted Lotus Notes

You can optimize encrypted Lotus Notes traffic in RiOS 7.0 and later. When you enable the encrypted Lotus Notes feature, traffic between SteelHeads is decrypted and the current Lotus Notes protocol optimization (RiOS 5.5) is applied.

Lotus Notes authentication

The Lotus Notes and the Domino server relies on a the Notes ID file for proper authentication. This file contains information for authentication and encryption between the client and the server in the Lotus Notes and Domino system. The Notes ID file is usually stored on the client. You must have a password to decrypt the ID file and use its contents, but you do not need a password to authenticate with the server (for example, MS-Exchange or other systems).

This section requires that you be familiar with Lotus Notes and Domino servers.

Optimization architecture

To optimize an encrypted connection between a Notes client and a Domino server, you must import the Domino servers ID file into the server-side SteelHead, because the SteelHeads in the path of the connection need to be able to decrypt and reencrypt the sent data. Next, configure the Domino server with a port on which it accepts unencrypted connections. This port can be either the standard port or an auxiliary port. Now, when a Notes client connects to the Domino server, the server-side SteelHead forwards the connection to the auxiliary port of the server.

After the connection is authenticated, the server-side SteelHead resets the connection of the Notes client but maintains the unencrypted connection with the Domino server on the auxiliary port. The Notes client now tries to establish a new encrypted connection, which the server-side SteelHead intercepts and handles as if it were the Domino server.

The server-side SteelHead (acting as the Domino server) generates the information necessary to encrypt the connection to the Notes client. The result is a connection that is encrypted between the Notes client and server-side SteelHead but unencrypted between the server-side SteelHead and the Domino server.

Configuring optimized encrypted Lotus Notes

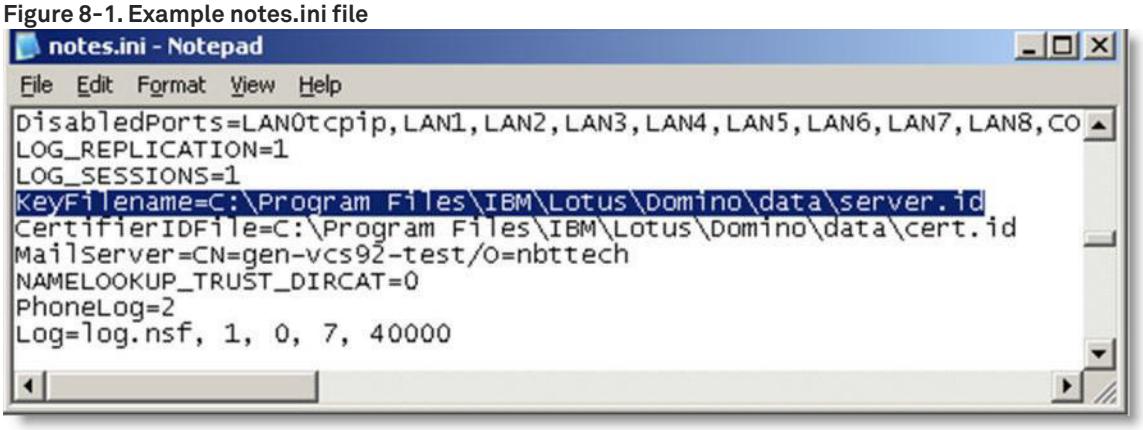
This section describes how to configure optimized encrypted Lotus Notes.

To import the server ID file of Domino servers that require optimization into the server-side SteelHead

1. Log in to the respective Domino servers and identify the location of the server ID file in the notes.ini file. This file is usually is located on a Windows server in C:\Program Files\IBM\Lotus\Domino\data.

2. Open the notes.ini file with a text editor.

Figure 8-1. Example notes.ini file

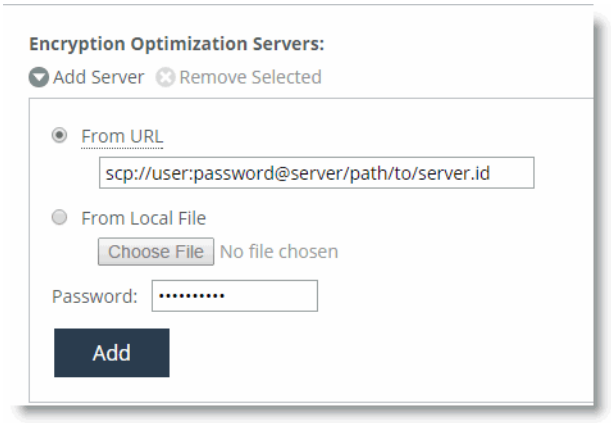


3. Import all server ID files into the server-side SteelHead. You do not need to do this on the client-side SteelHead).
   - From the Management Console, choose Optimization > Protocols: Lotus Notes. You can also use the **protocol notes encrypt import server-id <url> [password <password>]** command.
   - Select Add Server.
   - Specify the server ID file URL (or use the Browse button) and password.

Server IDs might not have passwords.

- Click **Add**.

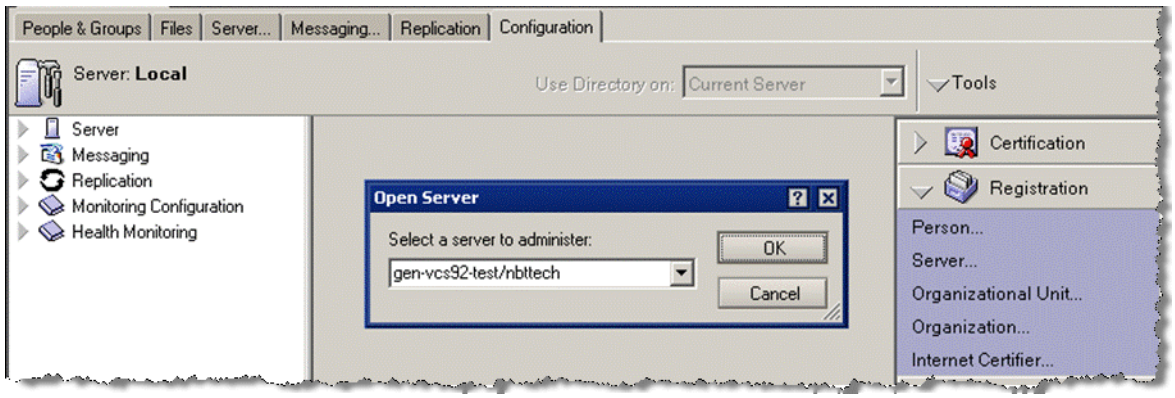
Figure 8-2. Import the Server ID file



To configure Domino servers to accept unencrypted connections on an auxiliary port

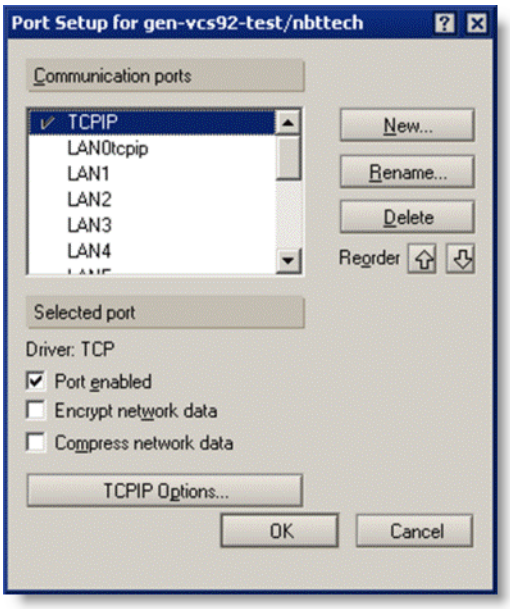
1. Connect to the Domino server.
2. Open the Domino Administrator and connect to the desired Domino server.

Figure 8-3. Domino server Administrator



3. From the Domino Administrator, choose Configure > Server > Setup Ports to open the Setup Ports page.

Figure 8-4. Setup Ports page



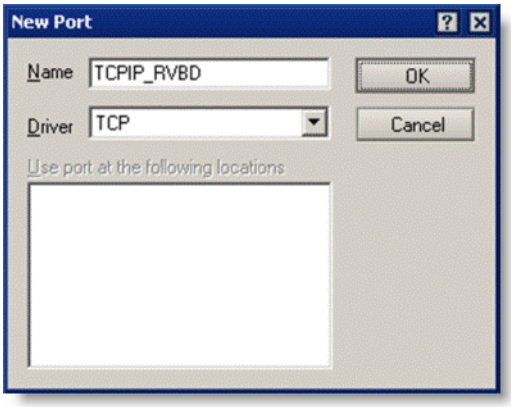
4. In the Setup Ports page, click **New** to create a new port.

5. Name the port.

   This example shows the port named TCPIP_RVBD and has the TCP selected in the Driver drop-down menu.

6. Click **OK**.

Figure 8-5. Creating a new port

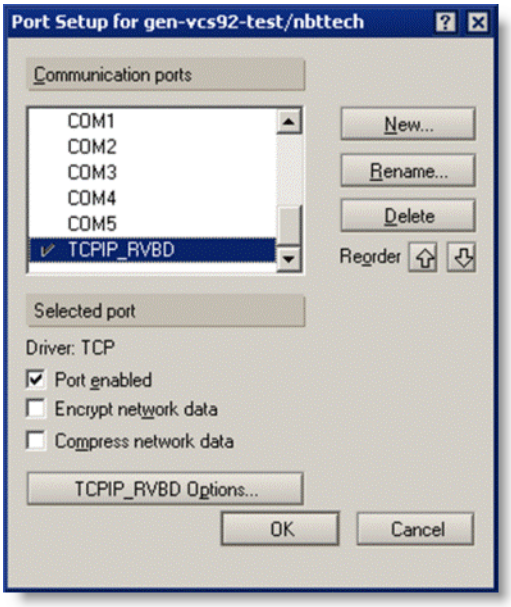


7. To enable the newly created port, select the new port in the Setup Ports page.

   Make sure that Port enabled is selected and Encrypt network data is not selected.

8. Click **OK**.

Figure 8-6. The new port on the Setup Ports page



To set a new TCP port number and restart the port

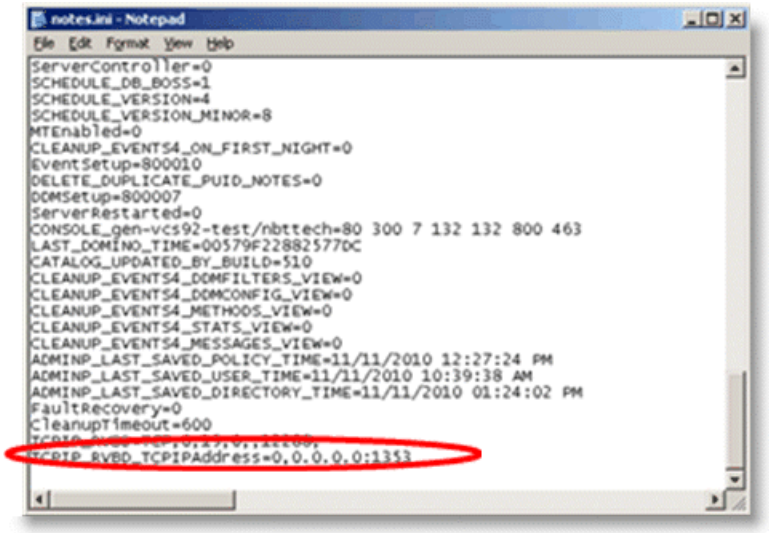
1. Open the Domino server's notes.ini file.

   The file is usually located in C:\Program Files\IBM\Lotus\Domino.

2. Add a line using the format <port-name>_TCPIPAddress=0,<ip-address>:<port>.

   Use the IP address 0.0.0.0; Domino listens on all server IP addresses.

Figure 8-7. Line added to the notes.ini file

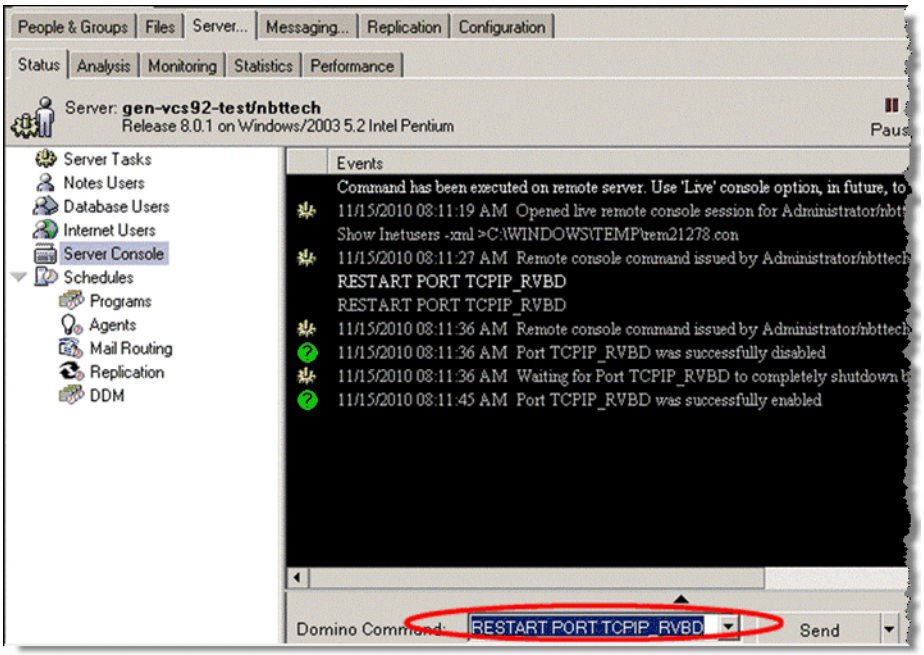


3. Restart the new port.

The Domino server starts to listen on the new port after a restart of the port or the server.

4. Select the Server tab.
5. Choose Server Console in the left tree structure.
6. In the drop-down menu at the bottom of the page, choose restart port TCPIP_RVBD.

Figure 8-8. Restart the port

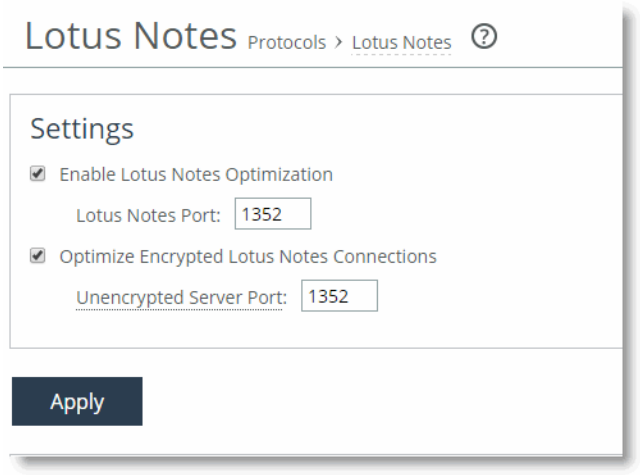


To enable Notes Encryption Optimization on both SteelHeads and set the appropriate alternate unencrypted port number on the server-side SteelHead

1. From the Management Console, choose Optimization > Protocols: Lotus Notes.
2. Select Enable Lotus Notes Optimization.
3. Select Optimize Encrypted Lotus Notes Connections.
4. Specify the configured unencrypted port number.

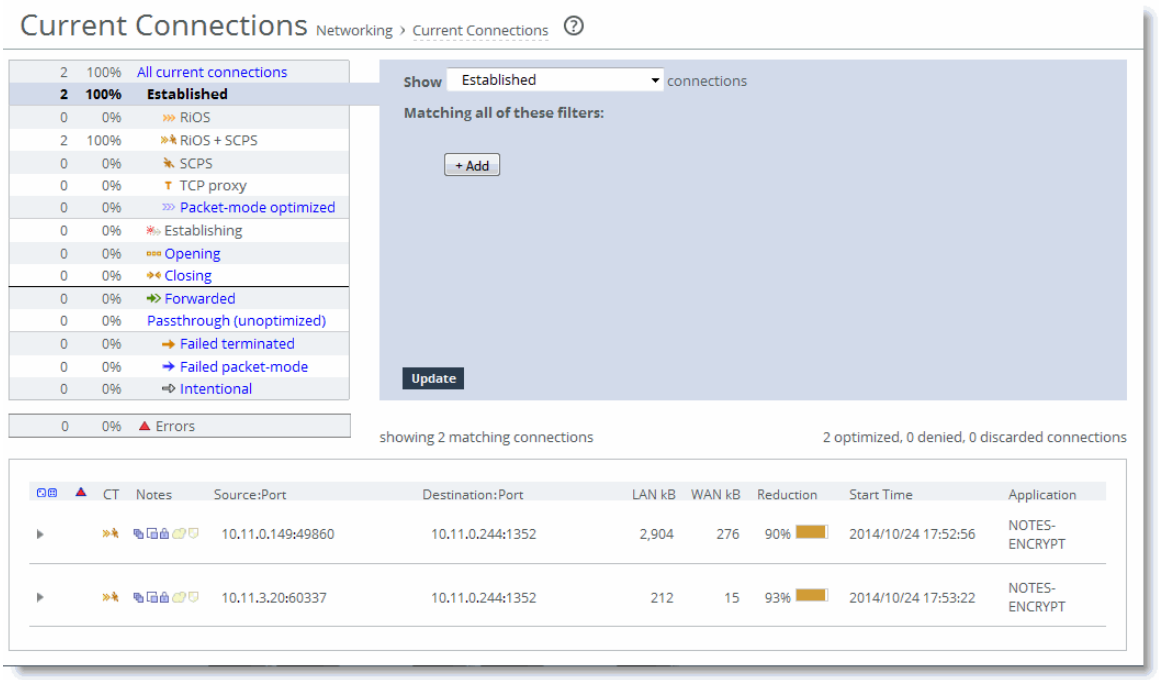
5. Click **Apply**.

Figure 8-9. Lotus Notes page



When the Notes client has connected to the Domino server, you see an optimized encrypted Notes connection in the current connections table. An example client-side SteelHead is shown in Figure 8-10.

Figure 8-10. Client-side SteelHead Current Connections page with optimized encrypted Lotus Notes



We recommend that you also enable secure peering on the SteelHeads. The optimized encrypted Notes connection between the client-side and the server-side SteelHeads is unencrypted. To secure this connection, you can enable secure peering on the SteelHeads. For more details, see “Deploying secure peering for all optimized traffic” on page 186.

Troubleshooting

If the SteelHead encounters an error with the Domino server, the IP address of the Domino server might be added to a blacklist at the server-side SteelHead to avoid disruption to the traffic. This includes configuration errors. New connections to or from the IP address are not optimized.

You can view blacklist entries with the **show protocol notes encrypt blacklist** command. You can clear the entire blacklist or a single entry with the **protocol notes encrypt blacklist remove-ip <ip-address>** command.

The best practice is to check the system logs of the SteelHead to see if something about your Lotus Notes optimization is not working properly: for example, there are messages if the wrong server ID file is imported or the unencrypted port is not configured on the Domino server.

Important notes:

- When you enable encrypted Lotus Notes optimization, you force all connections between a Notes client and a Domino server to be encrypted.
- Enabling Encrypted Notes optimization forces the client to always perform slow authentication.
- The Notes client might try to reuse a previously assigned ticket during authentication by sending an Auth request after the server's Hello response. This is called *fast authentication* because it saves several round trips. Fast authentication does not work when you enable Encrypted Notes optimization.

9

Video Optimization

This chapter describes how to optimize video across your network. Video is one of the fastest growing technologies in business today. Video supports an increasing number of applications, business processes, and training videos. Video on the WAN frequently consumes 30 to 60 percent of the bandwidth. Whether as video-heavy applications or streaming media, video can comprise more than half of traffic.

With RiOS 9.8 and later, SteelHead appliances support Dynamic Adaptive Streaming over HTTP (DASH), also known as MPEG-DASH.

This chapter includes the following sections:

- "Overview of Video optimization" on page 153
- "HTTP stream splitting" on page 154
- "Video on-demand with HTTP prepopulation" on page 157
- "On-demand video caching" on page 157

Overview of Video optimization

You can use video to disseminate information either through live or recorded content. As of RiOS 9.1, these are the solutions for video optimization:

- **Live Streaming with split-streaming** - *Live* streams are only available at a specific time. Examples include video streams of a live sporting event or broadcasting executive events to the workforce. In RiOS 7.0, you can optimize live streaming video with HTTP stream splitting.

  For more information about HTTP stream splitting, see "HTTP stream splitting" on page 154.

- **On-demand with HTTP prepopulation** - *On-demand* streams are stored on a server and transmitted when requested by a user: for example, training videos. In RiOS 7.0 and later, you can optimize pre-recorded video by using HTTP prepopulation.

  For more information about on-demand with HTTP prepopulation, see "Video on-demand with HTTP prepopulation" on page 157 and the "HTTP prepopulation" on page 165.

- **On-demand video caching** - On-demand video streams with cache eligible headers are stored as entire objects by the web proxy feature on the client-side SteelHead, enabling subsequent requests for the same video to be served locally from the cache.

  For more information about web proxy, "Overview of the web proxy feature" on page 97 and the *SteelCentral Controller for SteelHead Deployment Guide*. For more information about on-demand video caching, see "On-demand video caching" on page 157.

Distribution of video impacts not only the overall bandwidth use, but it also impacts other services running on the network. In the case of live broadcasting, you might need multiple simultaneous streams per broadcast to support multiple bit rates (for example, remote or wireless workers might watch video at a lower bit rate, while viewers in the office might watch video at a higher bit rate for a larger screen). As the number of streams increases, the likelihood increases that other services, such business applications, are affected.

Depending on the codec, resolution, and software in use, video distribution can use anywhere between 16 Kbps (for a low resolution, highly compressed video stream optimized for a small display) to 15 Mbps or more (for a high resolution HDTV stream) per stream. Typical enterprise video streams use between 350 and 500 Kbps to support a single user desktop application.

Connectivity can be overwhelmed quickly and does not scale well when you compare these speeds to typical branch office connectivity, in which you send a single stream for each user. For example, a typical T1circuit running at 1.544 Mbps (or an E1 running at 2.048 Mbps) might be able to serve users' needs for many business applications, but it cannot support more than 4 or 5 concurrent video streams. When the entire company might be watching a video broadcast or video conference at the same time, the need for efficient delivery of video across the WAN becomes clear.

When you optimize video streams of either type, you reduce the overall impact of video traffic on the WAN, thereby ensuring service continuity and optimal user experience.

HTTP stream splitting

RiOS uses HTTP stream splitting to optimize the following different live video technologies:

- Microsoft Silverlight (RiOS 7.0 and later)
- Adobe HTTP Dynamic Streaming (RiOS 7.0 and later)
- Apple HTTP Live Streaming (RiOS 8.5 and later)

You can optimize video technologies for on-demand video. For details, see "Video on-demand with HTTP prepopulation" on page 157.

Note: This section requires you be familiar with your origin server and video encoder.

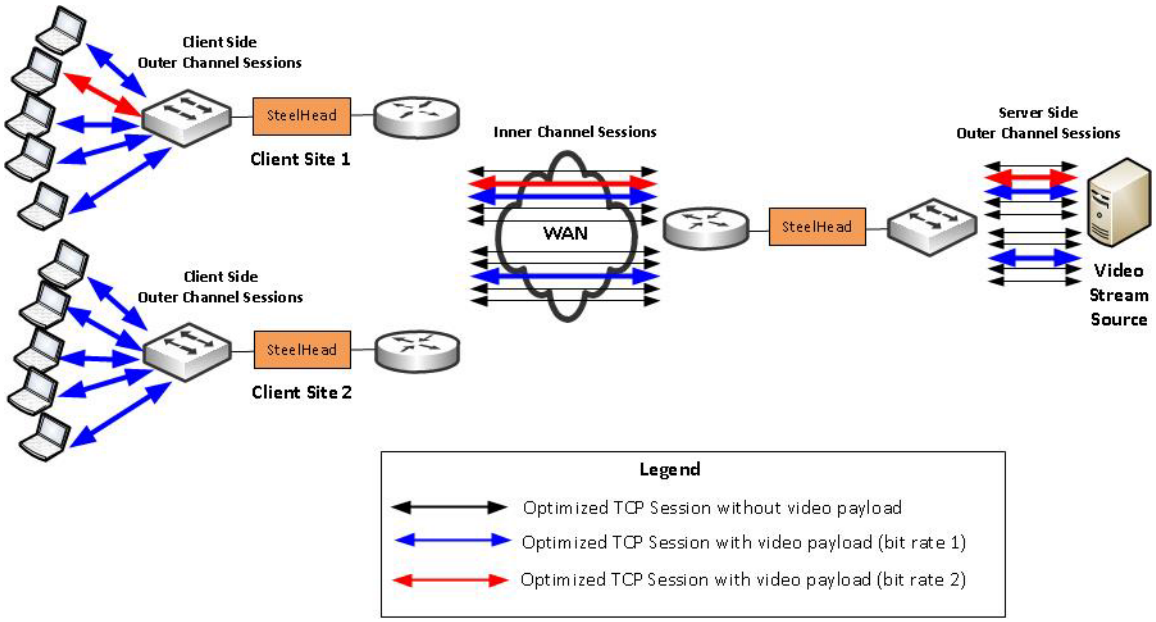
An unoptimized live video stream can saturate a T1 link with as few as four viewers. Normally, each client that connects to view video draws its own stream, quickly exhausting the resources of smaller branch office links. With stream splitting, one stream per bit rate is sent from the data center to each branch office, and software at each of the branch offices splits or replicates the stream for each individual client connecting.

Use HTTP stream splitting to reduce the number of duplicate streams operating between the server-side video server and the branch office clients. When you enable stream splitting, the first request for a video stream at a certain bit rate is sent out over the WAN, and the requests made by additional clients at the same branch are sent from the SteelHead. These additional requests are serviced after the client-side SteelHead has cached the first chunk of the video (typically, a single video consists of multiple video chunks or segments). As a result, only one copy of the stream is sent across the WAN no matter how many viewers are tuned in for the live stream. In RiOS 9.2 and later, the SteelHead identifies these streams using the video manifest file or video header metadata.

By using new streaming technology, you can ensure viewers always get the quality of video best suited for their conditions. Viewers with more bandwidth and processing power receive a higher-quality video stream than viewers with less bandwidth and processing power. If another user requests the video at a different bit rate, another stream is created for that rate.

Figure 9-1 shows a streaming video session between a single streaming video source and multiple clients at two client sites. The client-side SteelHeads provide a video stream for each client site. One client at Client Site 1 requested the video at a different bit rate and another stream at that rate is provided for that client. Both a client-side and a server-side SteelHead is required for this deployment. Each client maintains an active TCP session with the video stream source; this session is generally used for licensing purposes.

Figure 9-1. Server-client video streaming



RiOS 9.1 and later improve live video stream splitting with the following enhancements:

- The stream splitting cache holds more video fragments for a longer period of time to account for clients that could be out of sync or slower to play back.
- A new report plots the cache hit count over time for a particular live video indicating the amount of video requests that were served locally from the cache instead of being fetched over the WAN. The graph also includes a plot for the number of total live video sessions intercepted.
- The ability to enable video stream splitting on a per-host basis. The ability to selectively enable stream splitting on a particular host ensures that the cache does not fill up with recreational content.

For more information about these enhancements, see the *SteelHead User Guide*.

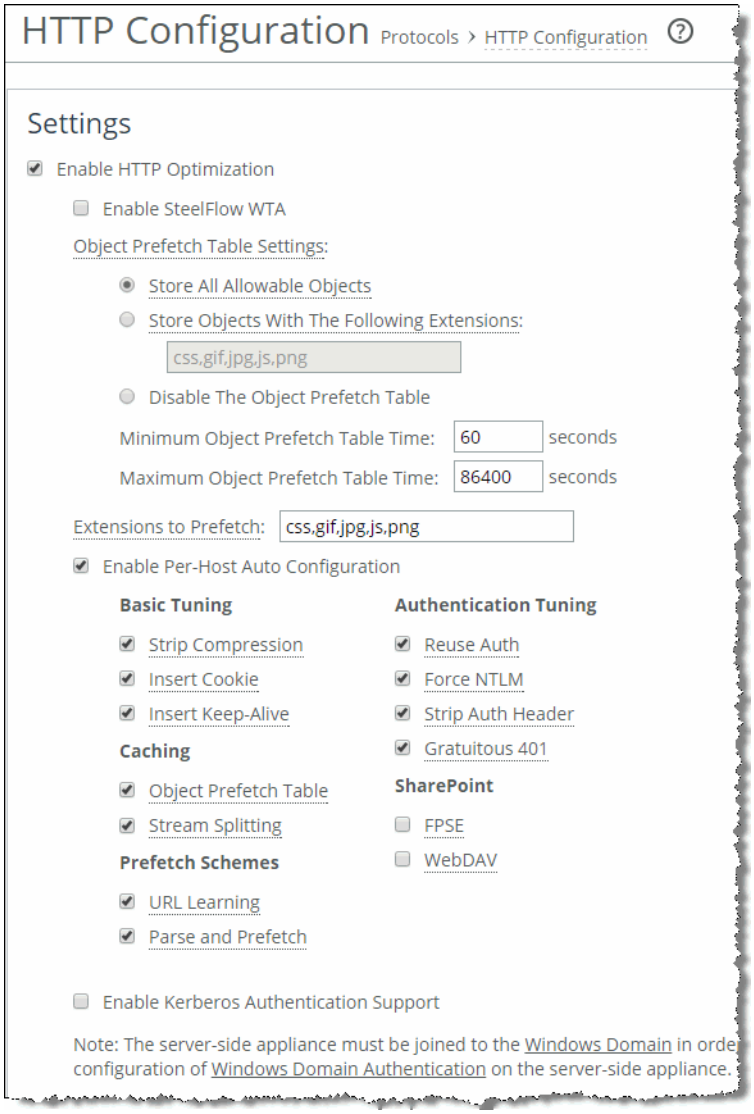
To enable HTTP stream splitting

1. Set up the video origin server.
2. On the SteelHead Management Console, choose Optimization > Protocols: HTTP.

3. Select Stream Splitting.

   Your video is now automatically optimized.

Figure 9-2. HTTP configuration page



For CLI commands associated with this feature, see the *Riverbed Command-Line Interface Reference Manual*.

The following resources provide more information:

- For information about Microsoft Silverlight smooth streaming, go to http://www.silverlight.net/.
- For more information about Adobe dynamic HTTP streaming, go to http://help.adobe.com/en_US/flashmediaserver/devguide/WSC1835B89-E6FF-46cd-AC7D-0E7A8EB331DDDev.html.
- For more information about Apple HTTP live streaming, go to http://developer.apple.com/library/ios/#technotes/tn2224/_index.html.

Video on-demand with HTTP prepopulation

Company updates and new internal training videos are examples of video content that can cause bursts of traffic when they are first made available. Video content is generally accessible only from web servers and can only be accessed using HTTP. Prewarming RiOS data store during off hours helps to reduce WAN bandwidth consumption during peak hours. While HTTP prepopulation enables you to prepopulate any information over HTTP, this feature is geared toward video optimization.

For more information about HTTP prepopulation, see "HTTP prepopulation" on page 165.

On-demand video caching

You enable the web proxy feature from the SCC, and you must have the SteelHead xx70 hardware platform running RiOS 9.1. You can configure rules to select traffic that is cached on the client-side SteelHead. You must mark the content you want to cache for web proxy so that the video files are stored as web objects on the disk.

Using the web proxy feature achieves the same benefits as HTTP prepopulation, but web proxy uses a different area of disk inside the SteelHead, enabling larger objects to be cached. Any subsequent requests for the same content that is in the web proxy cache are served, in their entirety, from the cache. Serving the video locally removes the overhead of transferring the redundant request over the network.

For more information about web proxy, see "Overview of the web proxy feature" on page 97 of the *SteelCentral Controller for SteelHead Deployment Guide*, the *SteelCentral Controller for SteelHead User's Guide*, and the *SteelHead User Guide*.

10

CIFS and HTTP Prepopulation

You can avoid the penalties of a cold transfer by using Common Internet File System (CIFS) and HTTP prepopulation to warm the SteelHead with data not yet requested by end users. Prepopulation is fully integrated into RiOS 7.0 and later.

The prepopulation of data is extremely beneficial for end users who know of data crossing the WAN: for example, operating system patch updates, antivirus update files, a video training session posted on an internal file server for offline viewing, or a directory of a user share located at the data center.

You configure prepopulation only on the client-side SteelHead. You must configure and connect the client-side SteelHead primary interface properly for prepopulation to work. The primary interface must be able to establish connection with the server.

The primary interface of the SteelHead acts as a client and requests data from the share you want to use to warm the RiOS data store. To warm both SteelHeads, data flows from the server, passes the server-side SteelHead, and ends at the client SteelHead.

The traffic leaves the primary interface and it must traverse a client-side in-path connection (for example, LAN0_0/WAN0_0 for an inline deployment, or WAN0_0 for a logical in-path deployment) before reaching the server-side SteelHead. Prepopulation does not work if the primary interface bypasses client-side optimization first.

This chapter includes the following sections:

- "CIFS prepopulation" on page 159
- "HTTP prepopulation" on page 165

CIFS prepopulation

CIFS prepopulation uses a flexible scheduler to prepopulate the RiOS data store. When a client on the remote LAN requests data again, only new or modified data is sent over the WAN, which dramatically increases the rate of data transfers. Unlike with Proxy File Service (PFS), the client-side SteelHead drops the actual data files it has received, but it maintains SDR data in the RiOS data store.

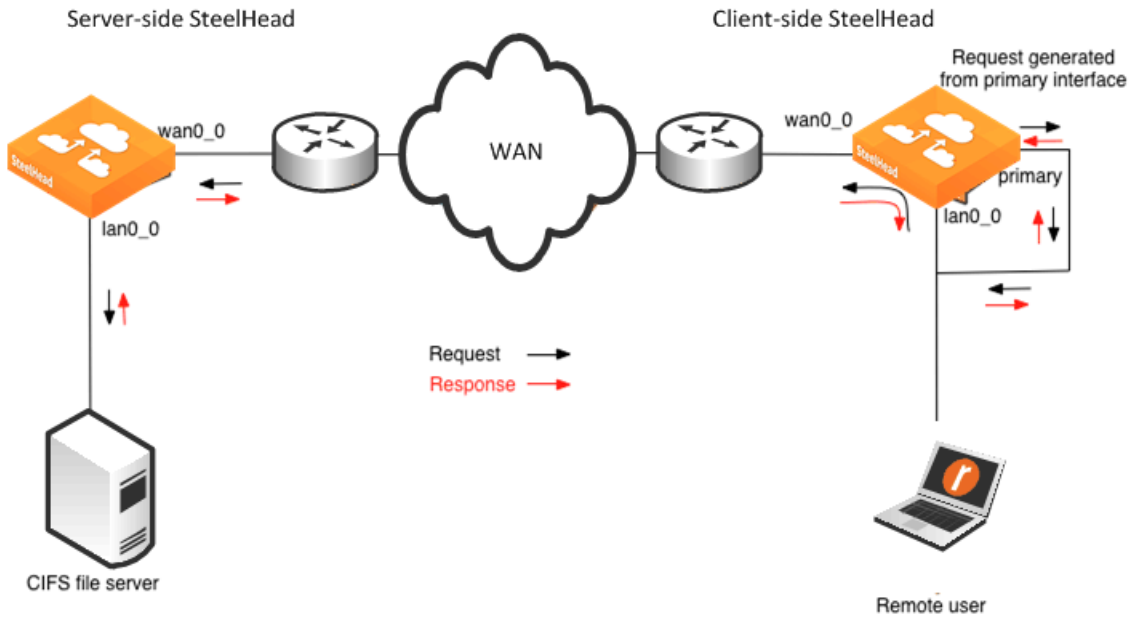
Because CIFS prepopulation uses the CIFS/SMB protocol, you can use a Windows machine, or any other CIFS/SMB capable NAS filer, for the file server.

In RiOS 7.0 and later, CIFS shares that require SMB signing are fully supported. Depending on the file server configuration, the authentication for SMB signing can be NTLM or Kerberos, but both methods are supported by CIFS prepopulation. To populate SMB signed CIFS shares, you must join the SteelHead to the domain in which the shares are published or to a trusted domain.

If the file server authentication is NTLM, then you must configure NTLM transparent mode on the server-side SteelHead. If the file server authentication is Kerberos, then you must configure the AD environment that the server-side SteelHead has joined (by creating a replication user account) and configure the server-side SteelHead to support Kerberos.

For more information, see “Encrypted MAPI optimization” on page 25 and “Kerberos” on page 56. For more information about SMB signing, see “Signed SMB and Encrypted MAPI Optimization” on page 45.

Figure 10-1. Traffic flow for CIFS prepopulation



When you create a share using CIFS prepopulation, the SteelHead attempts to connect to the destination server on port 139. You must enable NetBIOS over TCP/IP on a Microsoft Windows machine (server or client) to accept the connection on port 139. On a Windows server, enter the following command to see if it is listening on port 139:

```
C:\>netstat -an | find ":139"
TCP 192.168.1.20:139 0.0.0.0:0 LISTENING
```

If you do not see similar output with the correct host IP, NetBIOS over TCP/IP is not enabled on the interface. You can enable it on the relevant interface from the WINS tab in the Advanced TCP/IP settings.

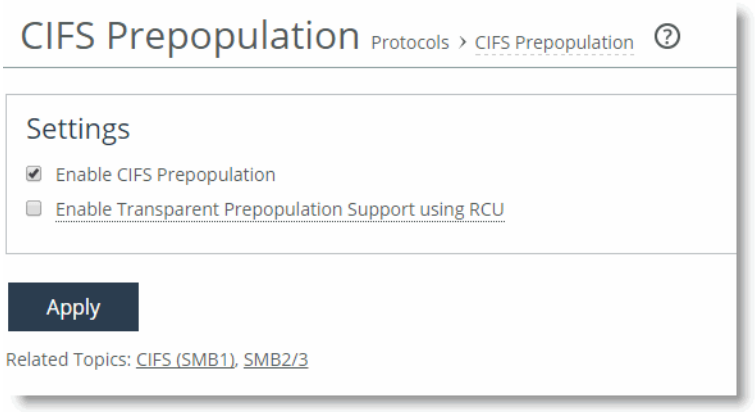
In RiOS 8.5 and later, you can create policies and rules for a higher level of control over which files the system transfers to warm the RiOS data store. A policy is a group of rules that select particular files to prepopulate; for example, you can create a policy that selects all PDF files larger than 300 MB created since January 1st of 2013.

To configure CIFS prepopulation and add a CIFS share using the Management Console

1. From the Management Console, choose Optimization > Protocols: CIFS Prepopulation.

2. Select Enable CIFS Prepopulation and click **Apply**.

Figure 10-2. CIFS Prepopulation page



3. Select Add a Prepopulation Share.
4. Specify the remote path where data is stored in UNC standard: for example, `\\bw-sfowad1\shared`.
5. Specify the account username. This username is the login name that has a minimum of read access to the remote path.
6. Specify your password and confirm your password.
7. Add a comment about the share. Comments cannot include an ampersand (&).
8. Specify a time limit that the synchronization job should not exceed.
9. Specify a limit on the amount of data in the synchronization job.
10. Select either current files for synchronization or use the latest share snapshot.

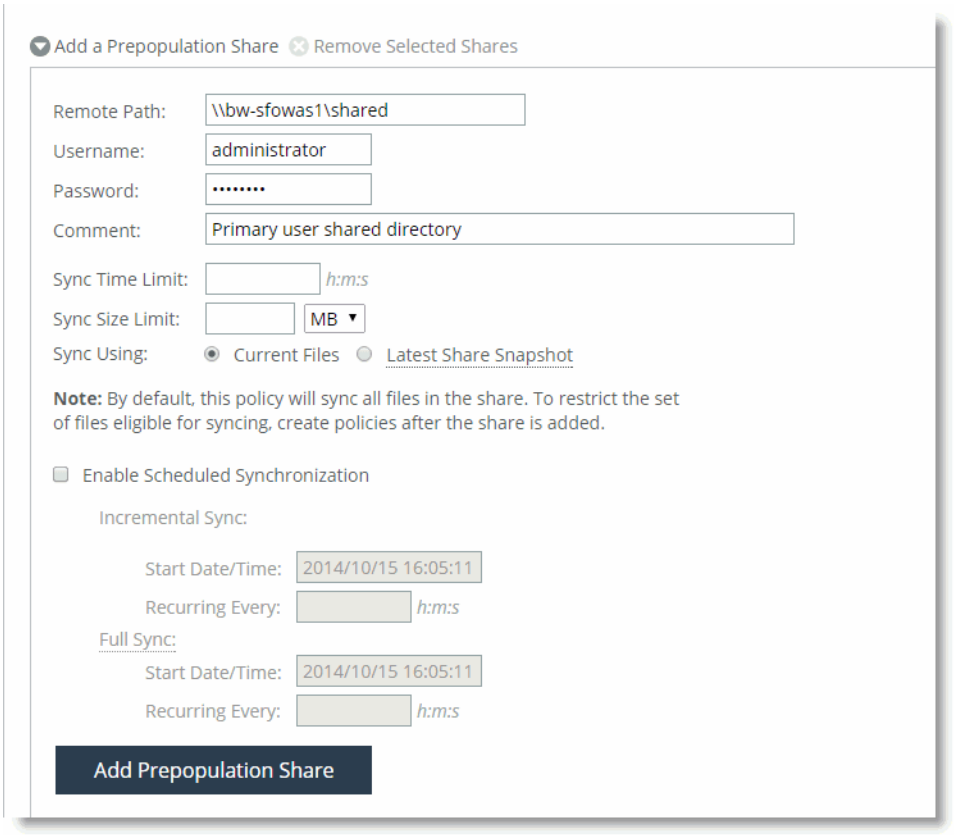
    If no snapshots are available, the system uses the current files.
11. To schedule regular prepopulation events, select Enable Scheduled Synchronization (optional).

    You can schedule full synchronization or incremental synchronization. Use Full Sync for data store wrap to keep potentially evicted data warm.

    If a Full Sync and an Incremental Sync occur simultaneously, the Full Sync takes precedence.
12. Click **Add Prepopulation Share**.

13. Click **Apply**.

Figure 10-3. CIFS Prepopulation Share information



Share synchronization begins, based on the parameters you entered.

Note: You cannot configure CIFS shares to be guest, anonymous, or public Samba shares without a password.

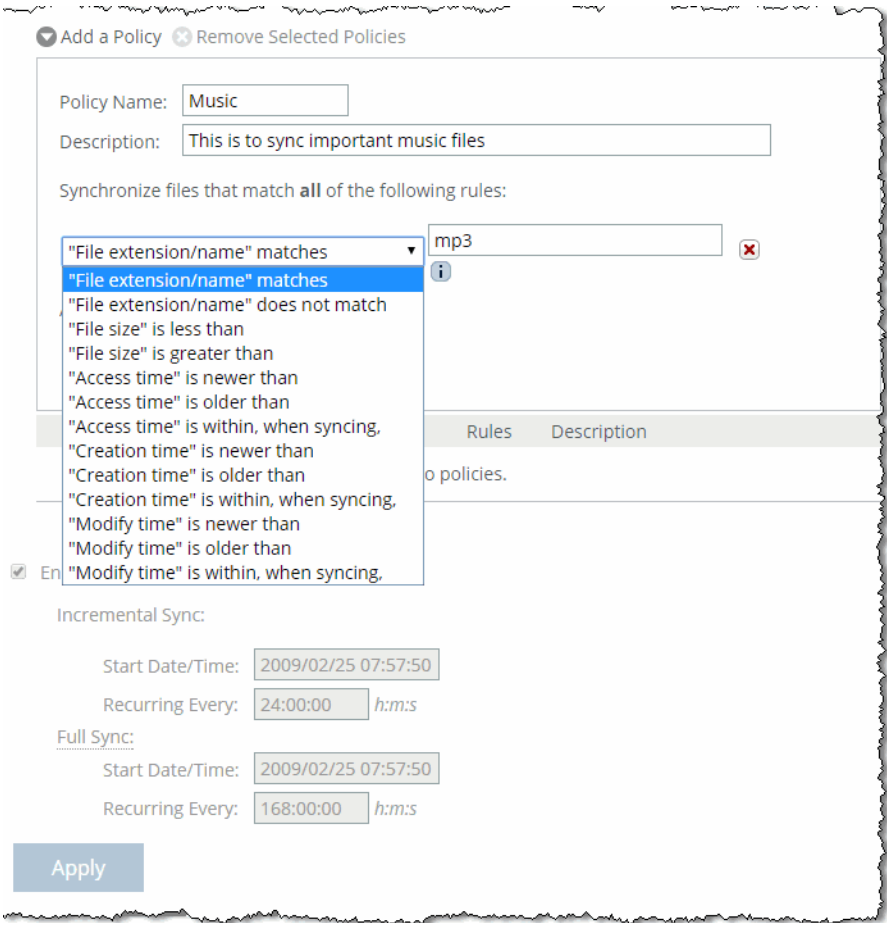
You can create synchronization policies in RiOS 8.5 and later. Synchronization polices enable you to define granular synchronization using filters for inclusion and exclusion specific data types.

To configure CIFS prepopulation policies

1. Open an existing prepopulation share.
2. Select Add a Policy.
3. Specify the policy name and a description.
4. Select data types from the drop-down list to create filers for the policy:
   - File extension
   - File size
   - Access time
   - Creation time

5. Modify time.

Figure 10-4. CIFS synchronization policy filers



For more information about the CIFS prepopulation CLI commands and configuration, see the *Riverbed Command-Line Interface Reference Manual*.

Design considerations

This section describes several design considerations.

Prewarming the RiOS data store during off-hours helps to reduce WAN bandwidth consumption during peak hours. When you configure CIFS prepopulation, it tries to use all available bandwidth. If you want to control bandwidth usage with CIFS prepopulation, you can configure QoS so that only a portion of the bandwidth is used for CIFS prepopulation.

To configure QoS to work with CIFS prepopulation

1. On the server-side SteelHead, enable QoS and set the proper link speed.
2. Set up a QoS class. Name it PREPOP and set an upper limit (for example, 10%).
3. Set up a QoS rule for the class PREPOP with a destination of the client-side SteelHead primary IP and give it a lower priority.

For more information about QoS, see the *SteelHead Deployment Guide*.

In RiOS 7.0 and later, clients can authenticate to servers requiring signing and synchronizing transparently from a Windows 2003 server DFS share.

Note: NTLM authentication mode is supported in RiOS 7.0 and later. In previous RiOS versions, CIFS prepopulation could only be performed against unsigned servers and required the use of transparent prepopulation support and the Riverbed Copy Utility (RCU) on the server.

In RiOS 7.0 and later, you can set the length of time a CIFS prepopulation command is to run. The time is reflected in seconds. You can run a test report on the client-side SteelHead to view the expected changes to be synchronized. A sample report follows:

```
(config) # prepop share modify remote-path '\\fileserver\share' max-duration 1800
#--- Set the maximum duration of a prepopulate run.
(config) # prepop share dry-run share-name '\\fileserver\share'
#--- Generate a dry run report where they can view a report of the expected changes.
(config) # show prepop log dry-run remote-path '\\fileserver\share'
```

RiOS 8.5 and later enhance CIFS prepopulation with a new policy-based system available in the CLI and the SteelHead Management Console. You can create specific policies and associate them with the CIFS prepopulation shares previously created.

The use of policies is a function of the client-side SteelHead. When you use policies, you can synchronize files:

- created since a given set time.
- modified since a given time.
- matching an expression.
- matching a given size range.
- accessed since a given time.

An example policy creation follows:

```
(config) # prepop share policy share-name '\\fileserver\share' policy-name policy1
#--- To create a policy 'policy1' using share '\\fileserver\share'.
(config) # prepop share policy share-name '\\fileserver\share' policy-name policy1 create-time time
'2011/10/01 00:00:00' compare-op after
#--- To prepopulate files created after that specific date.
(config) # prepop share policy share-name '\\fileserver\share' policy-name policy1 create-time time
'2011/10/05 00:00:00' compare-op before
#--- To prepopulate files created before the earlier specific date.
(config) # show prepop share policy share-name '\\fileserver\share' policy-name policy1
#--- To display the specifics of the earlier mentioned policy
(config) # prepop share policy share-name '\\fileserver\share' policy-name policy1 file-name
'A_*.pdf;*.txt' compare-op matches
#--- To only include files with the.pdf and.txt extensions to be prepopulated
#--- Notice the use of the wild character "*". Additional matches need to be separated by a
#--- semicolon.
(config) # prepop share policy share-name '\\fileserver\share' policy-name policy1 file-name '*.dat'
compare-op not-matches
#--- To exclude files with the.dat extension to be prepopulated.
config) # prepop share policy share-name '\\fileserver\share' policy-name policy1 file-size 10M
compare-op less
#--- To prepop files less than 10meg in size.
(config) # prepop share policy share-name '\\fileserver\share' policy-name policy1 file-size 5M
compare-op greater
#--- To prepopulate files greater than 5 MB in size.
```

You can delete a complete policy directly or delete a specific rule within a policy with the following command.

```
(config) #  no prepop share policy share-name '\\fileserver\share policy-name policy1
```

HTTP prepopulation

HTTP prepopulation, in RiOS 7.0 and later, is an enhanced HTTP-based data delivery method. HTTP prepopulation delivers data to the remote site by using the HTTP protocol to prewarm your RiOS data store. HTTP prepopulation uses the HTTP protocol to read and deliver data. The feature delivers any data content residing on a web server, including on-demand video streams at remote site. Remote users can have an enhanced viewing experience. For more information about video optimization, see the *SteelHead Deployment Guide*.

HTTP prepopulation requires that you configure and connect the primary interface for full functionality. HTTP prepopulation is only available through the command line. Configuration is only on the client-side SteelHead.

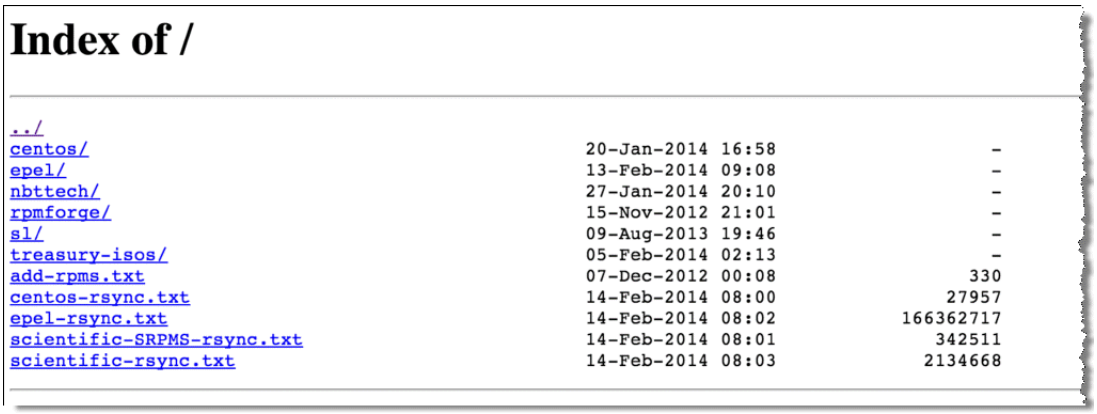
Note: If you are using HTTP prepopulation over IPv6, you must have a an IPv6 address on the primary interface. For more information about IPv6, see the *SteelHead Deployment Guide*.

To configure HTTP prepopulation you create a list composed of URLs that contain the data you want optimized. You can configure up to 100 lists, and you can include an unlimited number of URLs within each list.

For example, you can combine URL links to multiple Human Resource training videos in one list named *HRvideos*. The URL link must reference a filename with an extension in the URL link such as http://www.example.com/video/example.mp4, and not reference a general URL link such as http://www.youtube.com/videoexample.

Figure 10-5 shows a web server directory listing. The HTTP prepopulation functionality recursively downloads the entire folder content.

Figure 10-5. Web server directory listing



The following example shows how to configure HTTP prepopulation with the list name HRvideos. List names are case sensitive.

To configure HTTP prepopulation

1. On the client-side SteelHead, connect to the CLI in configuration mode.
2. Build the list name, for example:

   ```
   (config)# protocol http prepop list HRvideos
   ```

3. Add URLs to the list:

   ```
   protocol http prepop list HRvideos url http://intranet/hr/HowToInterview.mp4
   ```

   You can optimize content located on a secured web server (HTTPS). If HTTPS is specified as a protocol, you must configure SSL on the client-side SteelHead.

4. Start the process to transfer data across and prewarm your RiOS data store:

   ```
   protocol http prepop list HRvideos start
   ```

   The CLI session is unresponsive until the prepopulation runs to completion.

HTTP prepopulation time varies, depending on transfer time. If you want to maintain access to the CLI, we recommend that you use the **job** command to run the transfer at a specified time during off hours.

An example of the **job** command is as follows:

```
(config) # job 1 name prepop
(config) # job 1 command 1 "protocol http prepop list HRvideos start"
(config) # job 1 date-time 12:15:00
(config) # job 1 enable
```

To cancel an HTTP prepopulation job that has not begun

- On the client-side SteelHead, connect to the CLI and enter the following command:

  ```
  job <job-number> disable
  ```

To cancel HTTP prepopulation in progress

- On the client-side SteelHead, connect to the CLI and enter the following command:

  ```
  protocol http prepop list HRvideos cancel
  ```

Additional various supporting commands follow:

```
show protocol http prepop lists
show protocol http prepop status all
show protocol http prepop status list HRvideos
```

For more information about HTTP prepopulation commands, see “Microsoft Silverlight, Apple HLS, and Adobe Flash” on page 166 and the *Riverbed Command-Line Interface Reference Manual*.

Microsoft Silverlight, Apple HLS, and Adobe Flash

In RiOS 7.0 and later, the SteelHead supports prepopulation for Microsoft Silverlight On-demand streaming content. In RiOS 8.5 and later, the SteelHead supports prepopulation for Microsoft Silverlight Video, Apple HLS, and Adobe Flash.

To prepopulate video, you must specify the manifest URL containing the manifest file. The manifest file includes information about the different bit rates available, the timeline markers, and so on. Manifest files are generally produced by video encoding software that supports their related format: for example, the Microsoft expressions encoder for Microsoft Silverlight is MP4.

In RiOS 8.5 and later, you can specify a URL (as opposed to specifically referencing the manifest file in the URL link as in previous RiOS versions). RiOS 8.5 and later automatically:

- parse the content.
- identify the manifest file.
- extract the proper information.
- extract the constituent segment URLs.
- represent the videos for all bit rates to download.

You need a browser plug-in on the client machine to play and watch the video. For example, Microsoft Silverlight is compatible with Internet Explorer, Firefox, and Safari.

The Microsoft Silverlight manifest file is an XML file, with a .ISM extension, that describes the live or on-demand video presentation. The Apple HLS manifest file is a text-based manifest file with a .M3U8 or .M3U extension that directs the player to additional manifest files for each of the encoded streams. The Adobe HTTP Dynamic Streaming manifest file has a .F4M extension.

Note: A manifest file is invisible to the end user; only end-user video players use it.

To configure HTTP prepopulation, use the following commands on the client-side SteelHead:

- Create or delete a new prepopulation list with a user-specified name:

```
[no] protocol http prepop list *
```

- Add or remove URL to the list that needs to be recursively prepopulated. All the objects in the page are fetched recursively (1 level deep):

```
[no] protocol http prepop list * url *
```

- Start the prepopulation operation:

```
protocol http prepop list * start
```

- Cancel the prepopulation operation:

```
protocol http prepop list * cancel
```

- Display HTTP prepopulation status:

```
show protocol http prepop status all
```

- Display specified list status info:

```
show protocol http prepop status list *
```

- Display specified list configured URLs:

```
show protocol http prepop list *
```

- Display all the configured lists:

```
show protocol http prepop lists *
```

11

SSL Deployments

This chapter describes how to configure SSL support. This chapter includes the following sections:

- "The Riverbed SSL solution" on page 169
- "Overview of SSL" on page 171
- "Configuring SSL optimization on SteelHeads" on page 174
- "Advanced SSL features" on page 188
- "SSL optimization with SteelHead Mobile" on page 192
- "Troubleshooting and verification" on page 192
- "Interacting with SSL-Enabled web servers" on page 193
- "Deploying a SteelHead with a SafeNet Network HSM" on page 196

This chapter requires that you be familiar with the SSL protocol.

The Riverbed SSL solution

The Riverbed SSL solution optimizes data transfers that are encrypted using SSL, provided that SteelHeads are deployed locally to both the client-side and server-side of the network. All of the same optimized connections that are applied to normal nonencrypted TCP traffic, you can also apply to encrypted SSL traffic. SteelHeads accomplish this without compromising end-to-end security and the established trust model. Your private keys remain in the data center and are not exposed in the remote branch office location where they might be compromised.

The Riverbed SSL solution starts with SteelHeads that have a configured trust relationship, enabling them to exchange information securely over their own dedicated SSL connection. Each client uses unchanged server addresses and each server uses unchanged client addresses; no application changes or explicit proxy configuration is required. Riverbed uses a unique technique to split the SSL *handshake*.

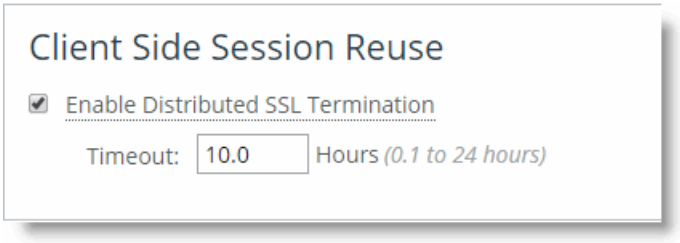
The handshake is the sequence of message exchanges at the start of an SSL connection. In an ordinary SSL handshake, the client and server first establish identity using public-key cryptography, and then they negotiate a symmetric session key to use for data transfer. When you use Riverbed's SSL optimization, the initial SSL message exchanges take place between the client application (for example, a web browser) and the server-side SteelHead.

Prior to RiOS 6.0, the SSL handshakes from the client are always handled by the server-side SteelHead, including session reuse handshakes.

RiOS 6.0 and later provide an alternative strategy, called *distributed termination*, in which initial full handshakes are terminated on the server-side SteelHead, while subsequent reuse handshakes are terminated by the client-side SteelHead.

Distributed termination is enabled by default and is on the Optimization > SSL: Advanced Settings page. The time-out value specifies the amount of time the client can reuse a session with an SSL server after the initial connection ends. The range is 6 minutes to 24 hours. The default value is 10 hours.

Figure 11-1. Distributed termination setting on the Advanced Settings page



Distributed termination improves performance by reducing the CPU load on the server-side SteelHead and shortens the key negotiation process by avoiding WAN round trips to the server. Distributed termination also shortens the key negotiation process by avoiding WAN round trips to the server. You can configure reuse of a client-side session for distributed termination on the Optimization > SSL: Advanced Settings page in the Management Console. For more information about distributed termination, see "How SteelHeads terminate an optimized SSL connection" on page 172.

Riverbed has worked with large enterprise design partners to ensure that SSL optimization delivers real world benefits in real-world deployments, specifically:

- sensitive cryptographic information is kept in the secure vault—a separate, encrypted store on the disk.
- built-in support for popular Certificate Authorities (CAs) such as VeriSign, Thawte, Entrust, and GlobalSign. In addition, SteelHeads allow the installation of other commercial or privately operated CAs.
- import of server proxy certificates and keys in PEM, PKCS12, or DER formats. SteelHeads also support the generation of new keys and self-signed certificates. If your certificates and keys are in another format, you must first convert them to a supported format before you can import them into the SteelHead. For details, see "SSL optimization required components" on page 174.
- separate control of cipher suites for client connections, server connections, and peer connections.
- bulk export or bulk import server configurations (including keys and certificates) from or to, respectively, the server-side SteelHead.
- that you can use the SCC to streamline setup of SteelHead trust relationships.

Note: For more information, see the *SteelHead User Guide* and *SteelCentral Controller for SteelHead User's Guide*.

The SteelHead has a secure vault that stores all SSL server settings, other certificates (that is, the CA, peering trusts, and peering certificates), and the peering private key. The secure vault protects your SSL private keys and certificates when the SteelHead is not powered on.

Initially the secure vault is keyed with a default password known only to the RiOS software. This allows the SteelHead to automatically unlock the vault during system start up. You can change the password, but the secure vault does not automatically unlock on start up. To optimize SSL connections, the secure vault must be unlocked.

Overview of SSL

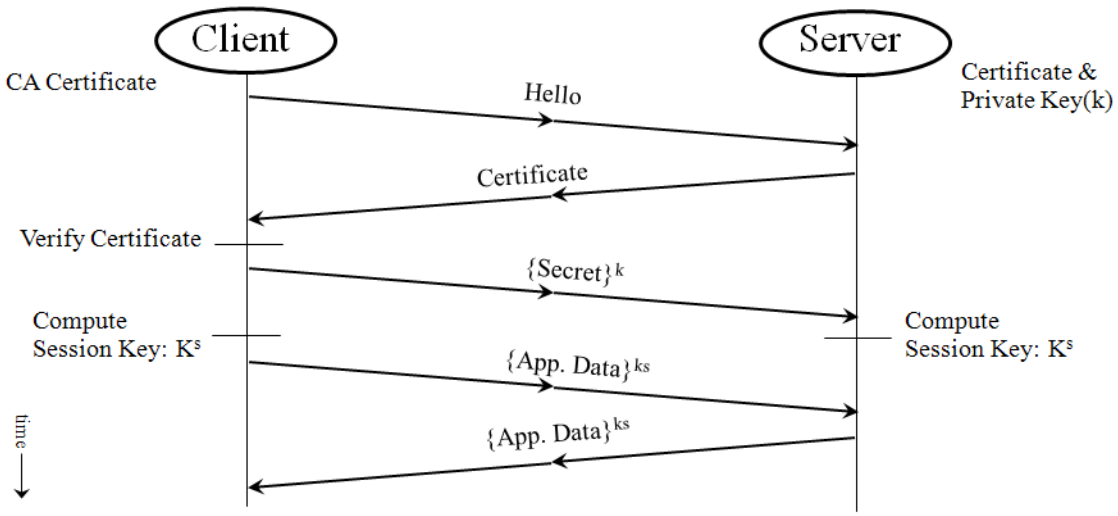
Because a complete description of SSL is outside the scope of this guide, this section provides a brief overview of the relevant components of SSL security.

SSL provides a way for client and server applications to communicate securely over a potentially insecure network. It provides authentication and prevents eavesdropping and tampering. In the most common use case, you use can SSL to transport HTTP traffic and provide one-way identification: only the web server authenticates itself to the web browser.

Note: The SSL features have changed with each release of RiOS. If you are running a version of RiOS that is earlier than 6.0, consult the appropriate documentation for that software release.

Figure 11-2 shows a simplified SSL handshake where the server authenticates itself to the client.

Figure 11-2. A simple SSL handshake



In response to the Hello message, the server sends back its certificate. The certificate contains the server's public key, some identifying information about the server (such as its name and location), and a digital signature of all this information. This digital signature is issued by an entity called a Certificate Authority (CA) that both the client and the server trust, and it serves as proof that no one has tampered with the certificate.

Upon receiving the certificate, the client verifies that it has not been tampered with and that it does belong to that particular server. Then, the client generates a random number N, encrypts it with the server's public key, and sends it to the server. At this point, both the client and the server use the same function to derive the session key, ks, from N. In the simplest case, after the initial SSL handshake between the client and the server and the creation of the session keys, the same session keys are used for the duration of the session, without the need to go through another SSL handshake.

How SteelHeads terminate an optimized SSL connection

At a high level, SteelHeads terminate an optimized SSL connection by making the client think it is communicating to the server and making the server think it is talking to the client. In fact, the client is talking securely to the SteelHeads. You require some special provisioning to accomplish optimization that appears transparent to the client.

To enable SSL connection termination, you must configure the server-side SteelHead to include proxy certificates and private keys for the purpose of emulating the server. For details, see "Generating the proxy certificate and private key pair" on page 181. When the SteelHead poses as the server, there does not need to be any change to either the client or the server. The security model is not compromised—the optimized SSL connection continues to guarantees server-side authentication and prevents eavesdropping and tampering.

When transferring data over the WAN on behalf of an optimized SSL connection, the client-side and server-side SteelHeads ensure that their inner connection provides all the security features the original SSL connection would have, had it not been optimized. SteelHeads accomplish this by establishing their own SSL connection between themselves. To secure the inner SteelHead channel, you must configure each SteelHead with the certificate of the peer SteelHead (secure peering). There are various methods to accomplish a secure inner channel between the SteelHeads. The authentication is mutual, with the client initiating the connections to authenticate the server, and the server authenticates the client. For more details, see "SteelHead secure peering scenarios" on page 183.

To securely terminate an SSL connection to an SSL server, the following list is a high-level configuration for the server-side SteelHead (for more information about how to configure SSL on a SteelHead, see the *SteelHead User Guide*):

- A certificate and private key pair for the server. This certificate and private key pair does not have to be the same as the one used by the actual server. In a production environment, it would typically be signed by a CA trusted by the client. You can import a signed server certificate into a SteelHead without a private key if you have used the same SteelHead to generate the certificate signing request for that server certificate.

  For more details, see *Importing CA signed certificate - based on the CSR generated by SteelHead* at https://supportkb.riverbed.com/support/index?page=content&id=S20404&actp=REPORT.

  To avoid confusion in this chapter, the certificate residing on the server-side SteelHead (for the server), is referred to as the *proxy certificate*. The proxy certificate might or might not be the same as the actual server certificate, but it serves the same function. The one exception is if you use client certificates. For details, see "Client certificate support" on page 188.
- The certificate of the client-side SteelHead.

The client-side SteelHead has only the server-side SteelHead certificate for secure peering between the client-side and server-side SteelHeads. The client-side SteelHead does not need the certificates or keys of the server. This behavior discourages any tampering at the branch office.

The following list describes distributed termination:

- Full SSL handshakes terminate on the server-side SteelHead.
- The master secret containing information that allows the computation of the session key for reusing the session is transported to the session cache of the client-side SteelHead.
- The subsequent handshakes are reused and terminate on the client-side SteelHead.

Figure 11-3 shows a high-level view of application data crossing the network from the client to the server. After the SSL connection is terminated, there are only three session keys involved: kc, ks, and kt. The data is encrypted with session key kc; then it is decrypted, optimized, and reencrypted with session key kt for the SSL secure inner channel between the two SteelHeads:

- k^c is the session key between the client initiating the transmission and the server-side SteelHead.
- k^s is the session key between the server-side SteelHead and the server.
- k^t is the session key between the client-side SteelHead and the server-side SteelHead for the SSL secure inner channel.

Figure 11-3. Typical data transfer operations on SSL connections optimized by SteelHeads

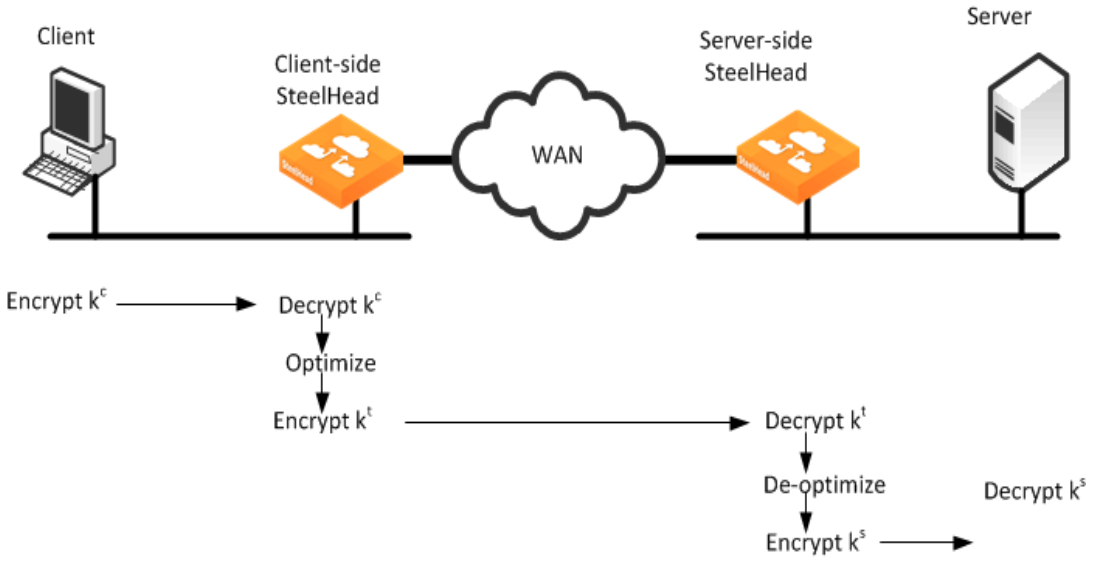
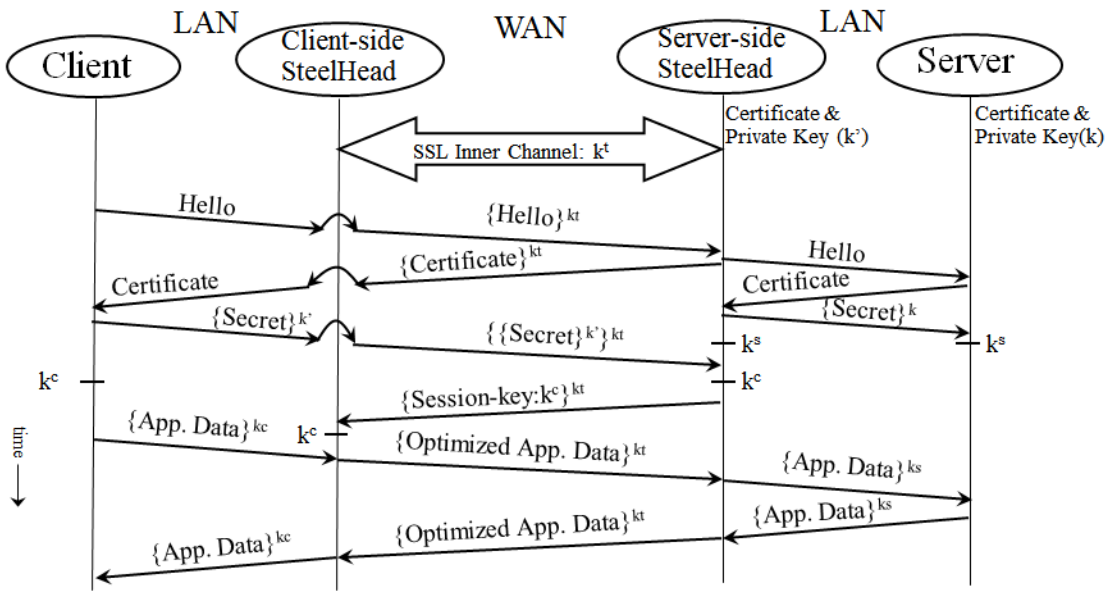


Figure 11-4 shows in time sequence the complete set of actions to set up an SSL connection.

Figure 11-4. Time sequence diagram



Configuring SSL optimization on SteelHeads

This section describes how to deploy SSL optimization on SteelHeads and provides common configuration examples. This section includes the following topics:

- "SSL optimization required components" on page 174
- "Setting up a simple SSL optimization deployment" on page 177
- "Generating the proxy certificate and private key pair" on page 181
- "SteelHead secure peering scenarios" on page 183
- "Deploying secure peering for all optimized traffic" on page 186

SSL optimization required components

You need the following SSL components to deploy SSL on SteelHeads:

- "Enhanced cryptography license key" on page 175
- "Proxy certificate and private key" on page 175
- "Certificate chain discovery" on page 176
- "Certificate authority certificates" on page 176
- "Peer certificates" on page 176

Enhanced cryptography license key

U.S. export restrictions require that this license is installed on each SteelHead that has SSL optimization enabled. You can acquire an enhanced cryptography license key by filling out the online form available at http://sslcert.riverbed.com.

Proxy certificate and private key

The proxy certificate is the certificate you configure on the server-side SteelHead for the server. Do not confuse this certificate with the certificate used for the secure inner channel or secure peering between the two SteelHeads and, also, the server certificate that is installed on the web and the application server.

Note: To avoid confusion in this chapter, the certificate residing on the server-side SteelHead (for the server), is referred to as the *proxy certificate*. The proxy certificate might or might not be the same as the actual server certificate, but it serves the same function. The one exception is if you use client certificates. For details, see "Client certificate support" on page 188.

The proxy certificate can be self-signed, signed by a well-known CA, or signed by your organization's own CA. It can be the same as, or different from, the certificate used by the actual server. The correct type of certificate depends on your deployment.

You can use a single wild card certificate and its private key as the proxy certificate for multiple servers.

For example, you might have three distinct servers using different certificates: sales.riverbed.com, internal.riverbed.com, and marketing.riverbed.com. You can add each of the three server certificates (as the proxy certificates) and their corresponding private keys, or you can add a single wild card certificate (*.riverbed.com) with its private key.

Note: In RiOS versions earlier than 6.0, you must configure a proxy certificate for each server. RiOS 6.0 and later support the use of wild card proxy certificates.

RiOS 6.0 and later simplify the SSL configuration process because the server-side SteelHead does not need to have a proxy certificate for each server individually. Prior to 6.0, you had to provide an IP address, port, and certificate to enable SSL optimization for a server. This method is impractical in cases where multiple servers shared the same server certificate because each server had to have a corresponding entry on the server-side SteelHead.

In RiOS 6.0 and later, you simply add unique proxy certificates to a certificate pool on the server-side SteelHead. When a client initiates an SSL connection with a server, the SteelHead matches the common name of the certificate on the server with one in its proxy certificate pool. If it finds a match, it adds the server to the list of optimizable servers, and all subsequent connections to that server are optimized with the same proxy certificate.

If it does not find a match, it adds the server name to the list of bypassed servers and all subsequent connections to that server are not optimized. The optimizable and bypassed server lists appear on the server-side SteelHead SSL Main Settings page of the Management Console.

If your proxy certificate and private key are in a format other than PEM, PCKS12, or DER, you must convert the certificate and key before you import them. The file extension does not necessarily dictate the encoding, as many common extensions such as .crt, .cer or .key can be either PEM or DER encoded.

To convert a certificate and private key, we recommend using an open source toolkit such as OpenSSL. For details, go to http://www.openssl.org/.

For information about converting a Base64 encoded PCKS12 certificate to PEM, see the Riverbed Knowledge Base article Converting a PCKS12 Certificate/Key to PEM at https://supportkb.riverbed.com/support/index?page=content&id=S13301.

Certificate chain discovery

You can use intermediary certificates instead of signing all certificates with a root level certificate. In releases earlier than RiOS 5.5, you needed to put all of the intermediate certificates on the server-side SteelHead and keep them up-to-date. With RiOS 5.5 and later, you can configure the server-side SteelHead to automatically pull down intermediate certificates from the back-end server.

Use the following command on the server-side SteelHead to enable certificate chain discovery:

```
protocol ssl backend server chain-cert cache enable
```

Alternatively you can use the Management Console to enable certificate chain discovery. Choose Optimization > SSL: Advanced Settings and select Enable SSL Server Certificate Chain Discovery. By default, this option is disabled.

Certificate authority certificates

Before the server-side SteelHead can establish an SSL connection to the server, it needs to verify the authenticity of the server certificate. This verification is based upon the model of trust relationships established within a public key infrastructure.

In this model, the issuer of the certificate is a third party that is trusted by both the subject (owner) of the certificate and the party (client) relying on the certificate. A certificate authority (CA) is an entity that issues a certificate. A client establishes a trust relationship to the CA first and, subsequently, uses the certificate of the CA to verify the authenticity of a server certificate. A SteelHead establishing an SSL connection acts as a client.

A SteelHead includes a list of commonly used public CA certificates preinstalled. When the certificate of your server is signed by a private CA, you must install that private CA certificate in the server-side SteelHead. This certificate establishes the trust relationship to the private CA and, consequently, the server certificates that this private CA issues.

Note: With RiOS 9.8 and later, Server-side SteelHead appliances can validate SSL certificates returned by the server using the online certificate status protocol (OCSP).

Note: Before adding a CA, you must verify that it is genuine; a malicious CA can compromise network security.

Peer certificates

Each SteelHead participating in SSL optimization requires a peer certificate. The peer certificates allow the client-side and server-side SteelHeads to establish a secure inner channel. You can distribute these certificates several different ways: manual cut-and-paste; using the white, gray, and black peering lists; or through the SCC.

The peer certificates between the SteelHeads can be, and in most cases are, self-signed certificates. The session keys derived between the SteelHeads are used for encrypting data within the secure inner channel. The encrypted data ensures that the SSL session is secure end to end. For details, see "Overview of SSL" on page 171.

Note: We strongly recommend that you use the SCC to centrally configure SteelHead secure peering and manage the peering certificates in a medium-to-large size SteelHead deployment. Using the SCC allows you to simplify SteelHead management and increase your operational efficiency.

Setting up a simple SSL optimization deployment

This section describes a simple deployment to help you set up, test, and understand basic SSL optimization functionality. This simple deployment chooses simplicity over security and installs a self-signed proxy certificate on the server-side SteelHead. You can test this deployment against an operational web server that uses SSL.

Note: This deployment example uses a self-signed certificate. We recommend that you verify with your enterprise IT security officer that the use of self-signed certificates is in compliance with your enterprise IT security practices.

To complete this deployment, you need two SteelHeads, a WAN simulator, a LAN connection between the client and client-side SteelHead, and another LAN connection between the server and the server-side SteelHead. This example uses a self-signed certificate; however, the configuration steps remain the same for a CA-signed certificate.

Note: You can enable a Certificate Authority (CA) service in SCC 8.6 and later. You can configure the CA service on the SCC as a root CA or an intermediate CA. This CA service enables you to issue trusted CA-signed secure peering certificates and proxy certificates to the SteelHeads that are managed by the SCC.

This deployment assumes that your web server is configured for SSL connections and is running and that the client-side and the server-side SteelHeads can optimize non-SSL TCP traffic.

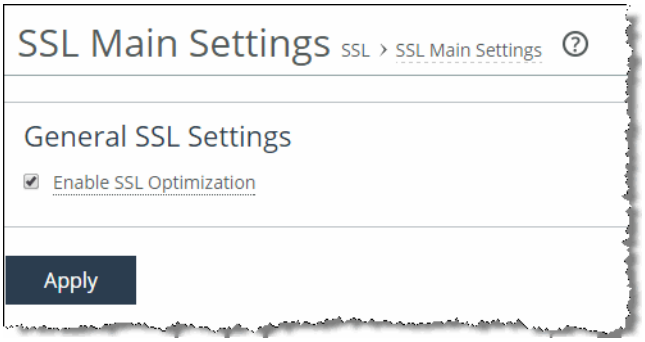
To set up a simple deployment for SSL optimization

1. In the SteelHead Management Console, choose Administration > Maintenance: Licenses.
2. Verify that both the client-side and server-side SteelHead have the Enhanced Cryptographic License Key installed to enable SSL optimization.
3. To set up the client-side SteelHead:
   - On the client-side SteelHead, choose Networking > App Definitions: Port Labels.
   - Select the secure port label, remove port 443, and click **Apply**.

   The default in-path rules contain a pass-through rule for all traffic destined to the ports defined in the secure port label. RiOS intercepts SSL traffic on the well-known TCP port 443. Use an in-path rule with a custom port and preoptimization policy set to SSL to intercept SSL traffic on another TCP port.
4. Enable SSL Optimization on both the client-side and server-side SteelHeads.
   - From the SteelHead Management Console, choose Optimization > SSL: SSL Main Settings (Figure 11-5).

- Enable SSL Optimization and click **Apply**.
- Restart the optimization service.

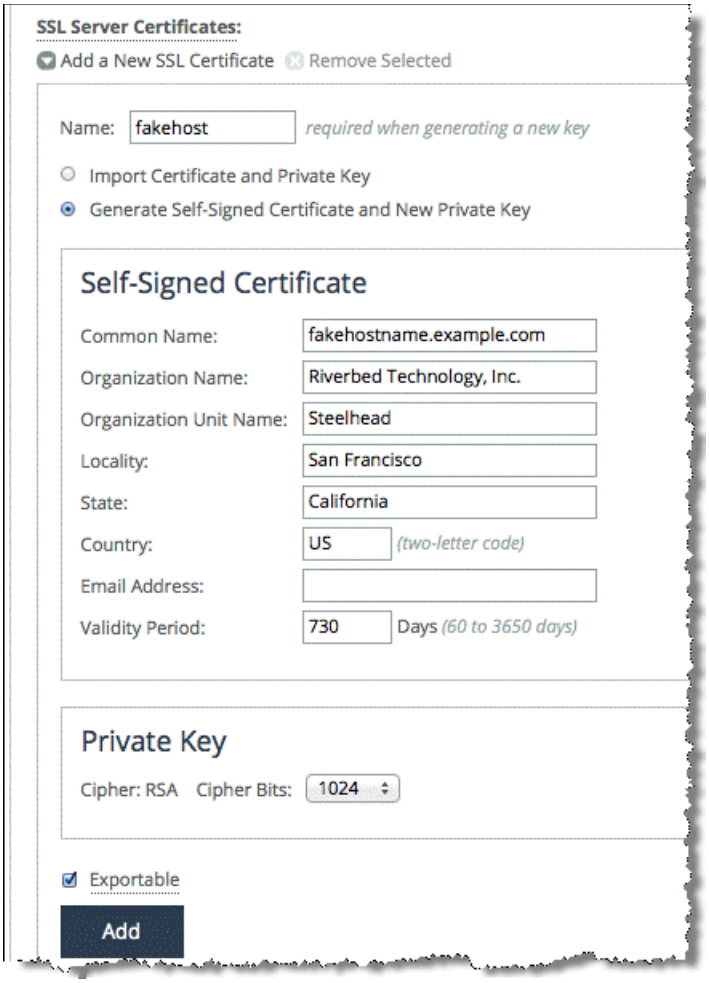
Figure 11-5. SSL Main Settings page



5. Create a proxy certificate on the server-side SteelHead:
   - Choose Add a New SSL Certificate (Figure 11-6).
   - Enter a name for the proxy certificate.
   - Select Generate Self-Signed Certificate and New Private Key.
   - Enter the proxy certificate common name and other details.

     The common name must match the server certificate common name or one of its subject alternative names.
   - Click **Add**.

Figure 11-6. Add a new SSL certificate

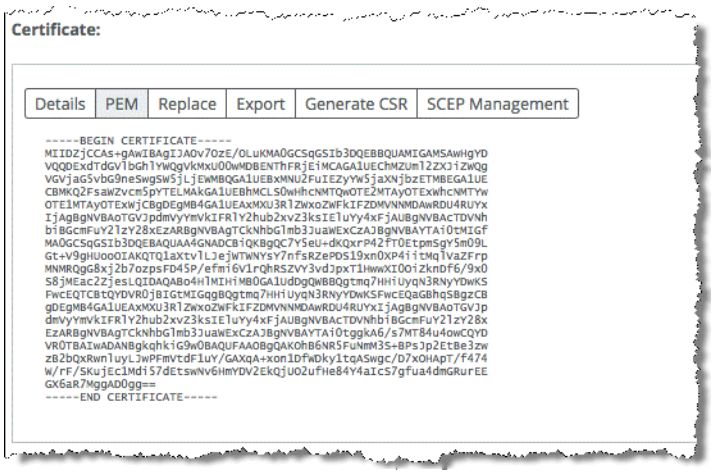


6. Configure secure peering (SSL) between the client-side and server-side SteelHeads.
   - From the client-side SteelHead, choose Optimization > SSL: Secure Peering (SSL).

     The default SSL secure peering settings is SSL Only. This setting instructs the SteelHead to pass all optimized SSL traffic through the secure inner channel.
   - Under Certificate, select PEM, and copy the client-side SteelHead secure peering (SSL) certificate (Figure 11-7).

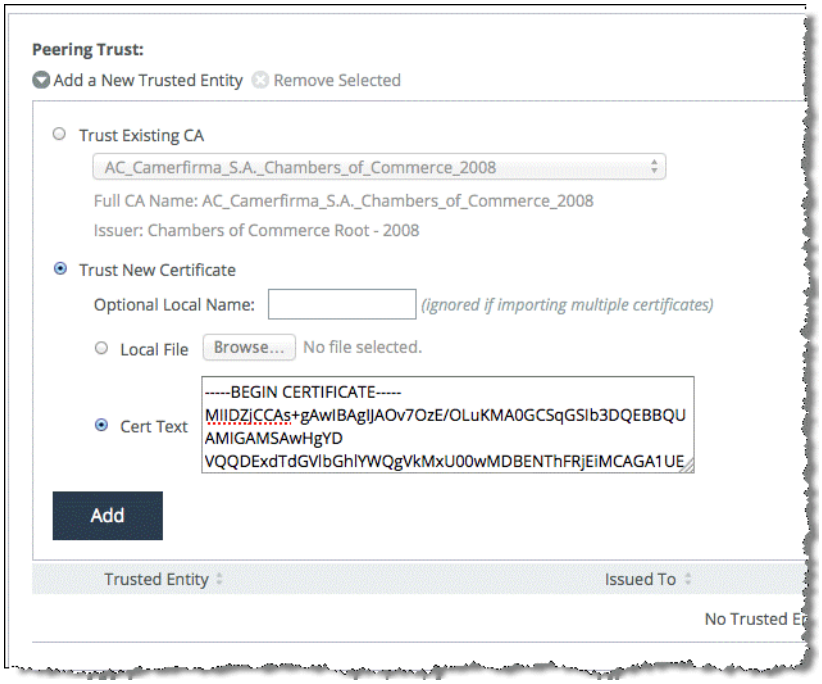
Do not confuse this certificate with the proxy certificate.

Figure 11-7. Secure peering (SSL) in PEM format



- From the server-side SteelHead, add the client-side SteelHead as a trusted peer by selecting Add a New Trusted Entity (Figure 11-8).
- Select Trust New Certificate.
- Select Cert Text and paste the client-side SteelHead secure peering (SSL) certificate into the text box.
- Click **Add**.

Figure 11-8. Adding a new trusted entity



7. Repeat the procedure to configure the server-side SteelHead as a trusted peer in the client-side SteelHead.

To verify that SSL optimization is successful

1. Connect to the secure web server from the client.
2. Choose Reports > Networking: Current Connections report in the Management Console on the server-side SteelHead and verify that the connections are being optimized without any protocol errors.

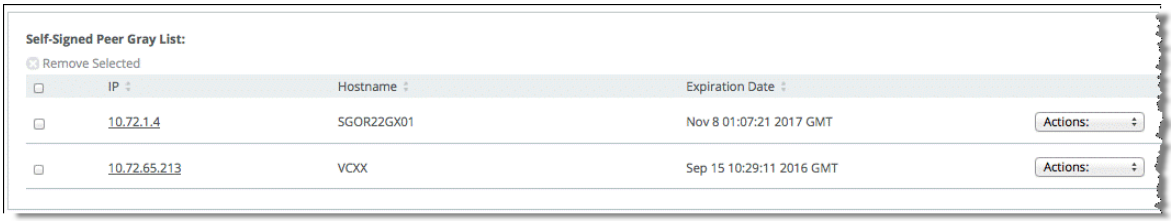
   If you see any protocol errors, expand the connection and examine the details.

For information about troubleshooting and verification, see "Troubleshooting and verification" on page 192.

The SteelHeads are capable of automatically detecting and configuring secure peering (SSL) between themselves if they are both using self-signed secure peering certificates. If SSL optimization is correctly configured on both the client-side and server-side SteelHeads, the self-signed secure peering certificate of the client-side SteelHead automatically populates into the server-side SteelHead self-signed peer gray list (Figure 11-9) and vice versa after an SSL connection is attempted across the SteelHeads. Use the drop-down menu to move the peer SteelHead into either self-signed peer white list or self-signed peer black list.

For more information, see "Secure peering using the self-signed peer white, gray, and black lists" on page 183.

Figure 11-9. Self-signed peer gray list



The client uses the proxy certificate to secure the SSL session instead of the server certificate in a SteelHead optimized SSL connection. This usage is because of the way the SteelHead terminates an optimized SSL connection. For more information, see "How SteelHeads terminate an optimized SSL connection" on page 172.

Note: Prior to RiOS 5.5, port 443 was not in the list of ports that had HTTP-specific optimization enabled. To turn this port on for an SSL-enabled web application in RiOS versions prior to 5.5, you must add the appropriate in-path rule on the client-side SteelHead to turn on HTTP optimization, in addition to the above steps.

Generating the proxy certificate and private key pair

The following table shows the four options available for generating the proxy certificate and its private key. The proxy certificate and its private key are installed in the server-side SteelHead. To optimize an SSL session, a proxy certificate for the server is required.

In an optimized SSL session, the client receives the proxy certificate instead of the server certificate. The proxy certificate might or might not be the same as the server certificate. This scenario is the same for its private key. Nonetheless, the choice of proxy certificate is transparent to the client as the server common name in both certificates is the same, and the connection is secure end-to-end.

For more information, see "How SteelHeads terminate an optimized SSL connection" on page 172 and "Proxy certificate and private key" on page 175.

Proxy certificate	Private key	Description
Real server certificate	Real private key	You can reuse the real server certificate and private key pair as the proxy certificate and private key pair. This option is ideal but is not always be available. For example, you can connect to a cloud-hosted application over SSL that uses a public certificate. The real server certificate and its private key are not available to you because you do not belong to the organization that owns the application.
New CA-signed proxy certificate	Real private key	You may reuse the real private key but create a new proxy certificate that is signed by a different CA. The real private key is required in situations where there is a security dependency from the client application, for example, when using client certificates for authentication. Choose this option if you want to administer all of your SteelHead proxy certificates (and secure peering certificates) from a different CA, for example, the SCC CA. You can also have some other requirements that cannot be fulfilled by the real CA, for example, the security requirements preclude server impersonation, the subject alternative names are a problem, you desire a longer validity period than the original certificate, or you want to use a wild card common name that is broader than the original. When using a CA-signed certificate, the CA certificate must be installed in the trusted root certification authorities store of the client machine.
New CA-signed proxy certificate	New private key	You can generate a new CA-signed proxy certificate and private key pair. This option can be an administratively quick way to obtain a new proxy certificate and private key pair for SteelHead SSL optimization. If your SteelHeads are managed with SCC 8.6 or later, you can use the SCC CA to generate the new proxy certificate and private key pair.
New self-signed proxy certificate	New private key	You can generate a new self-signed proxy certificate and private key pair. A SteelHead is capable of generating a new self-signed certificate and private key pair for its proxy and peering certificates. This is the fastest option available and can be used temporarily to fulfill the requirement. However, for security purposes, some browsers and applications do not accept self-signed certificates. You cannot establish the SSL connection in this case.

SteelHead secure peering scenarios

You can use the following methods to configure the peer certificates used for secure peering (SSL) between the client-side and server-side SteelHeads:

- "Secure peering using the self-signed peer white, gray, and black lists" on page 183
- "Secure peering with ca-signed certificates" on page 183
- "Managing SteelHead secure peering trusts with an SCC" on page 185

Secure peering using the self-signed peer white, gray, and black lists

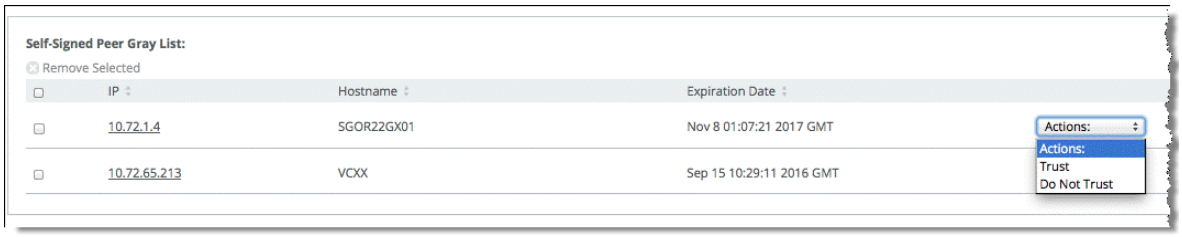
You can use peering lists to configure peer certificates on SteelHeads running RiOS 5.0 or later. When SSL optimization is enabled, and an optimizable SSL connection is attempted across the SteelHeads, the remote peer is detected and its peering certificate is automatically populated into the self-signed peer gray list. The connection remains unoptimized because the remote peer is listed in the self-signed peer gray list.

You can examine the peer certificate for authenticity and move any peer from the gray list to either the white list or the black list. If you trust the remote peer, move it into the self-signed peer white list to begin optimizing the SSL connection. If you do not trust the remote peer, move it into the self-signed peer black list to prevent the SSL connection from being optimized.

Note: We strongly recommend that you verify the authenticity of the secure peering certificate (and any other certificate in general as a best practice for IT security) before you trust it.

Figure 11-10 shows how you can move the peer certificate into either the white or black list with the drop-down menu.

Figure 11-10. Self-signed peer gray list



To verify that the SSL connection is successfully optimized after you have configured secure peering, view the connection in Reports > Networking: Current Connections. For information about verifying SSL connections, see "Troubleshooting and verification" on page 192.

Secure peering with ca-signed certificates

If you use CA-signed peer certificates, you can configure the SteelHead to trust the CA certificate instead of the peer certificate. The SteelHead implicitly trusts all certificates that the trusted CA issues. Alternatively, you can manually add the peer certificate into the peering trust list.

To add a remote peer SteelHead as a trusted peer using the peering trust list

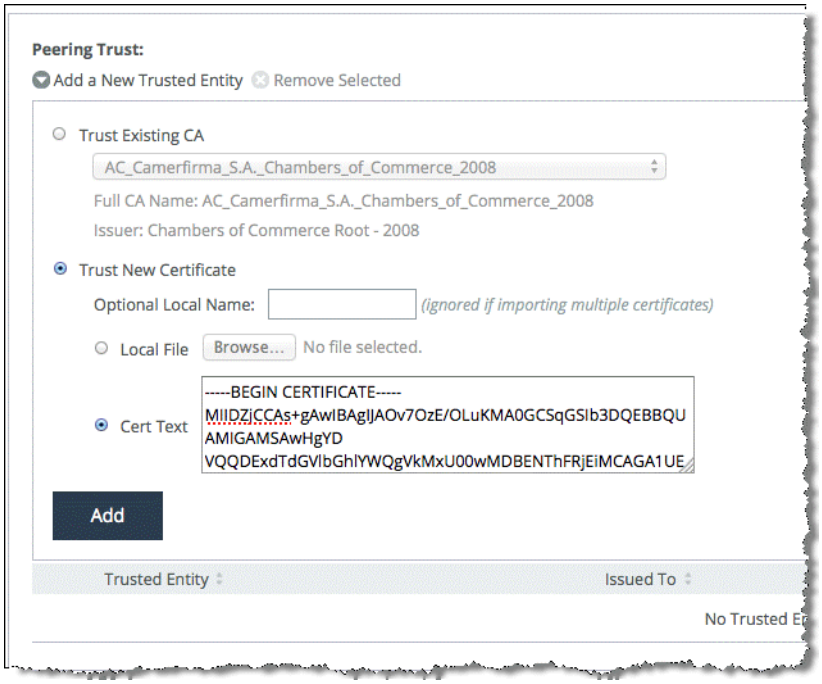
1. Choose Optimization > SSL: Secure Peering (SSL).
2. Select Peering Trust > Add a New Trusted Entity.

3. Select either Trust Existing CA or Trust New Certificate.

4. Choose Trust New Certificate (Figure 11-11), and copy and paste the peer certificate of the remote peer SteelHead into the Cert Text box or import the certificate file.

5. Click **Add**.

Figure 11-11. Adding a trusted remote peer



Alternatively, you can choose to trust the CA that issues the peer certificate. The SteelHead then trusts all the certificates that are issued by this trusted CA.

- Choose Trust Existing CA (Figure 11-12), and select the CA from the drop-down list. The list of CAs shown here is configured in Optimization > SSL: Certificate Authorities (Figure 11-13)

Figure 11-12. Configuring secure peering trust—trust existing CA

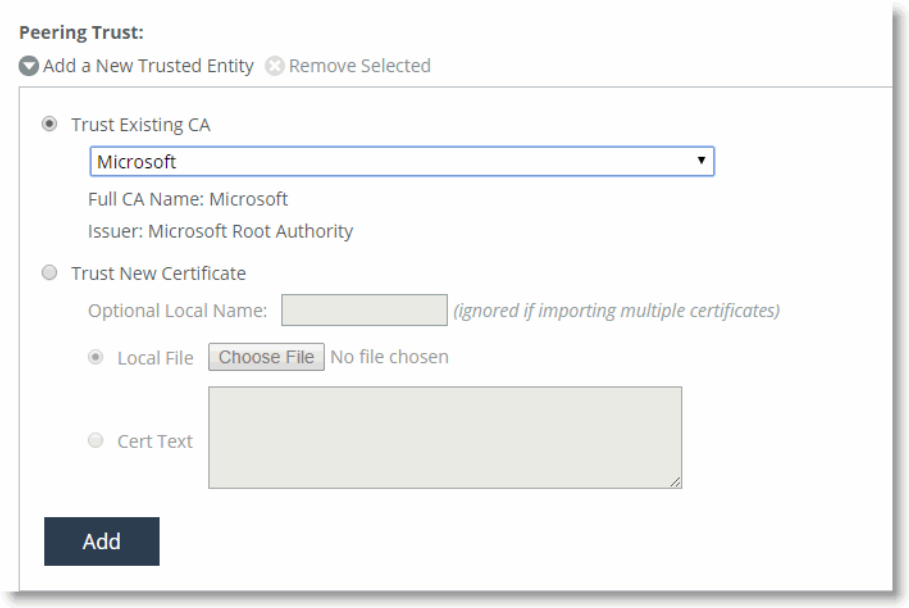
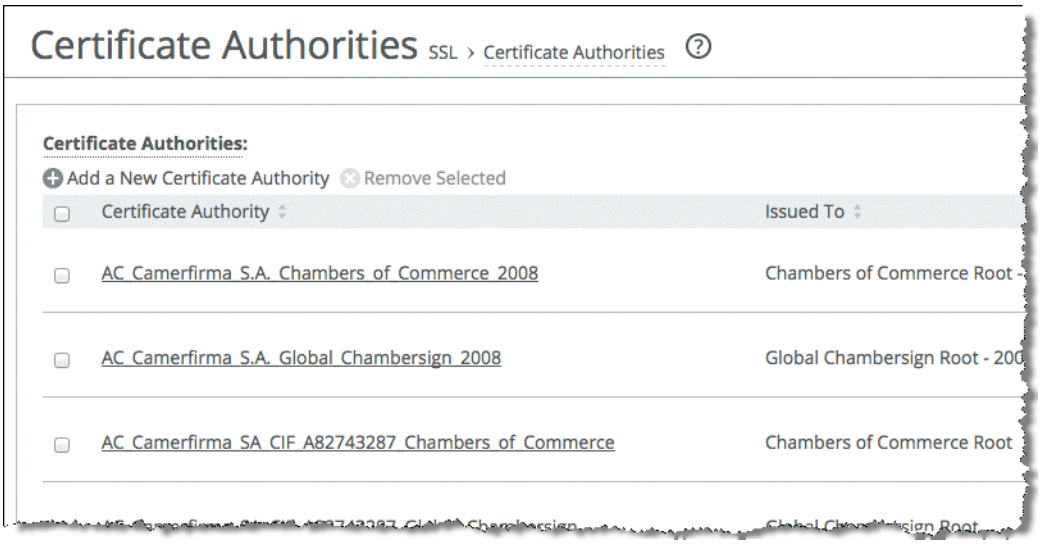


Figure 11-13. SteelHead Trusted Certificate Authorities List



If you have an intermediary CA, you must add them with the root CA. In secure peering, both the client-side and server-side SteelHead functions as an SSL client and server at the same time. For more details, go to https://supportkb.riverbed.com/support/index?page=content&id=S14925.

Managing SteelHead secure peering trusts with an SCC

We recommend that you configure a medium-to-large SteelHead deployment for SSL optimization with the help of an SCC. The SCC can simplify the configuration of SSL optimization and secure peering and increase the operational efficiency of a SteelHead deployment through central management thereby enabling the deployment size to easily scale.

The configuration effort required to manage the secure peering certificates and trusts in a SteelHead deployment increases proportionally with the number of SteelHeads deployed. The increase is due to the following reasons:

- Because multiple SteelHeads are involved, there are more secure peering certificates and trust relationships to manage.
- The number of proxy certificates might increase, and it is possible the same proxy certificate might be distributed to more than one SteelHead.
- You can reduce the number of proxy certificates by preparing a wild card certificate that is suitably valid, trusted, and matches the names of several target servers rather than generating one for each server.
- You can incorporate other features to increase the security of your SteelHead deployment: for example, you might encrypt the RiOS data store in selected locations or change the secure vault password. The secure vault stores sensitive information from your SteelHead configuration. The SCC is the only device that can supply the password over a secure channel to automatically unlock the secure vault whenever a SteelHead starts up.

A secure inner channel must be set up between the client-side and server-side SteelHeads in a deployment where SSL traffic or any other secure protocol traffic, such as signed-SMB and encrypted-MAPI, are optimized. Secure protocol traffic, including SSL traffic, are passed through the secure inner channel to maintain end-to-end data security. SCC 8.6 and later incorporates a certificate authority that can be used to centrally issue, distribute, and manage the secure peering certificates and peering trusts, in addition to the proxy certificates for SSL optimization. Using and trusting only CA-signed certificates increases the security model of an SSL deployment.

For information about using the SCC CA to manage the secure peering certificates and trust, see the *SteelCentral Controller for SteelHead Deployment Guide*. For information about adding a certificate authority to the secure peering trust configuration, see “Secure peering with ca-signed certificates” on page 183.

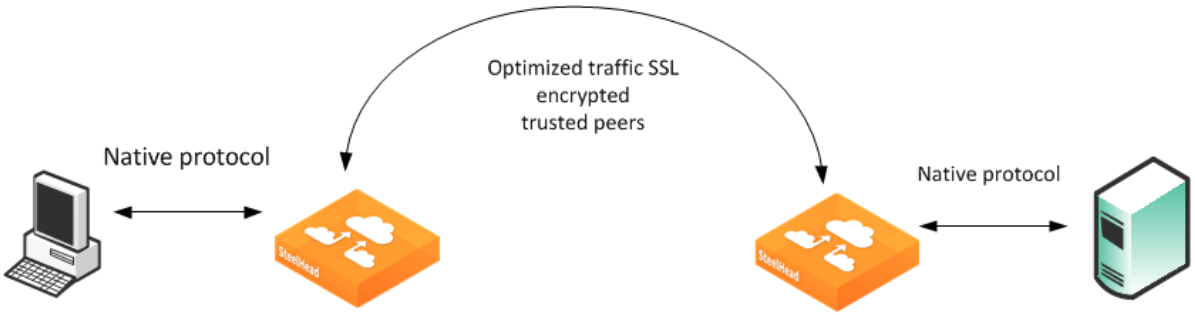
Deploying secure peering for all optimized traffic

In a high-security environment, you might want only known and trusted SteelHeads to peer for optimized traffic flows. Prior to RiOS 6.0, you could control peering only by modifying the peering tables, which was limiting because a secure model of peering trust could not be established between the SteelHeads. Modifying the peering tables creates situations in which SteelHeads can peer and optimize flows that might not be explicitly trusted. Using SSL secure peering, and forcing all traffic flows to flow through the secure inner channel, you can create an environment of explicit trust—only SteelHeads that can trust its peers can peer and optimize traffic flows.

In federated networks, it is common for organizationally separate SteelHeads to be aware of each other. If you use only secure peering, you create an environment of trust between the SteelHeads. If the secure inner channel connection fails, all conversations between the SteelHeads pass through. For more information about setting up secure peering, see “Setting up a simple SSL optimization deployment” on page 177 or “SteelHead secure peering scenarios” on page 183.

Figure 11-14 shows a client connection to a server. The default behavior of the SteelHead is to optimize this connection. In an environment in which secure peering is used for all optimized traffic, the SteelHeads have to trust each other before a secure inner channel can be established to carry the optimized traffic. Use the following procedure to set up a SteelHead to secure all optimized traffic and only optimize through a secure inner channel.

Figure 11-14. Trusted peers



To ensure all optimized traffic is secure

1. Choose Configure Optimization > SSL: Secure Peering (SSL) in the Management Console.
2. Select All from the Traffic Type drop-down list.
3. Clear the Fallback to No Encryption check box.

Figure 11-15. Secure Peering (SSL) page



4. Click **Apply**.

If you select All from the Traffic Type you can cause up to a 10% performance decline in higher-capacity SteelHeads. Take this performance metric into account when sizing a complete secure SteelHead peering environment.

Note: We recommend that you use NTP time synchronization, instead of manually synchronizing the clocks, on both the server-side and client-side SteelHeads. It is critical that the peer SteelHead time is the same for the trust relationship to work. Choose Administration > System Settings > Date/Time. Under Date and Time, select Use NTP Time Synchronization.

Advanced SSL features

This section describes the advanced SSL feature set:

- "Client certificate support" on page 188
- "Proxy server support" on page 190
- "Mid-session SSL support" on page 190
- "Server name indication" on page 191

Client certificate support

Client certificate support enables optimization of SSL traffic to SSL servers that authenticate SSL clients. The SSL server verifies the SSL client certificate. In the client authentication SSL handshake, each client has a unique client certificate and the SSL server, in most cases, maintains the state that is specific to each client when answering the client's requests. The SSL server must receive exactly the same certificate that is originally issued for a client on all the connections between the client and the server. Typically the client's unique certificate and private key are stored on a smart card, such as a Common Access Card (CAC), or on a similar location that is inaccessible to other devices on the network.

Enabling the client authentication feature allows SteelHeads to compute the encryption key and the SSL server continues to authenticate the original SSL client exactly as it would without the SteelHeads. The server-side SteelHead observes the SSL handshake messages as they go back and forth. With access to the SSL server's private key, the SteelHead computes the session key exactly as the SSL server does. The SSL server continues to perform the actual verification of the client, so any dependencies on the uniqueness of the client certificate for correct operation of the application are met. Because the SteelHead does not modify any of the certificates (or the handshake messages) exchanged between the client and the server, there is no change to their trust model. The client and server continue to trust the same set of CAs as they did without the SteelHeads optimizing their traffic.

Client authentication is also supported as part of a Mobile Controller deployment. You must have SteelHead Mobile 4.6 or later. For more information specific to the Mobile Controller, see the *SteelHead Deployment Guide*.

If the data center has a mixed environment with a few SSL servers that authenticate clients along with those that do not authenticate clients, we recommend enabling client authentication.

There are a few caveats when using client-side certificates:

- Both the client-side and the server-side SteelHeads must be running RiOS 6.5 or later. If you are running Mobile Controller, you must be running SteelHead Mobile 4.6 or later.
- Enable client certificate support on the server-side SteelHead.
- The server-side SteelHead must have access to the exact private key used by the SSL server.
- You must configure the SSL server to ask for client certificates.
- The SteelHead must have a compatible cipher chosen by the server.
- SSL sessions that reuse previous sessions that are unknown to the SteelHead cannot be decrypted.

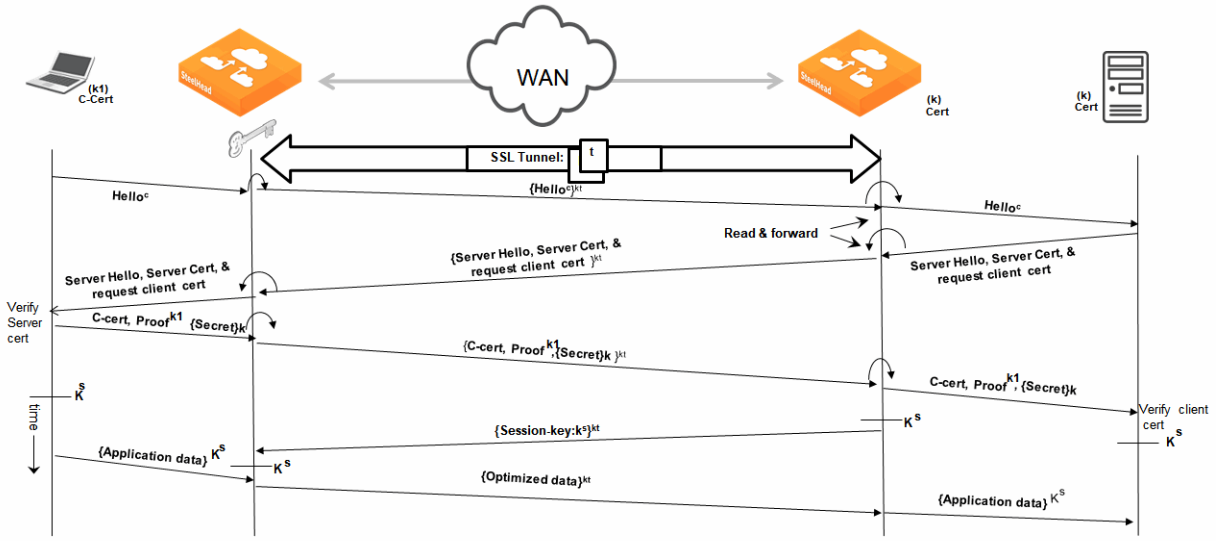
- Client-side certificates with renegotiation handshakes are not supported.
- Client-side certificate supports the RSA key exchange only. It does not support the Diffie-Hellman key exchange.

To enable client authentication

1. Perform the basic steps to enable SSL optimization. For details, see "Setting up a simple SSL optimization deployment" on page 177.
2. On the server-side SteelHead, choose Optimization > SSL: Advanced Settings, select Enable Client Certificate Support, and click **Apply**.
3. Choose Optimization> SSL: SSL Main Settings, import the SSL server private key and certificate, and click **Save** to save the configuration. You do not need to restart the optimization service.
4. If the client-side SteelHead is also running SteelHead Mobile, then you must import the Mobile Controller signing CA into the server-side SteelHead. To complete the task, choose Optimization > SSL: Secure Peering (SSL) and select Add a new Mobile Entity in the in the Mobile Trust box.

Figure 11-16 shows the time sequence of actions to set up an SSL connection in which the SSL server authenticates the SSL client.

Figure 11-16. SSL Server authenticates the SSL client



Note: We recommend that you use strong security ciphers, such as AES. Avoid stream ciphers.

Verification

To verify client authentication, on the server-side SteelHead, check the Discovered Server (Optimizable) table on the SSL Main Settings page. Optimizable servers that are using client authentication appear as optimizable streams. For servers that are not using client authentication, the server appears in the Discovered Server (bypassed, not optimizable streams) table with the reason *No proxy certificate configured for the server*.

Proxy server support

RiOS 7.0 and later include support for enterprise proxy servers that route web traffic (including secure SSL web traffic) on behalf of clients to a secure SSL server.

For more information, including a configuration example, see “Configuring HTTP SSL proxy interception” on page 83.

Mid-session SSL support

RiOS 7.0 and later support mid-session SSL support. A normal SSL session starts with an SSL handshake. The rest of the session is encrypted using session keys until the session is torn down. There are some scenarios where the session starts unencrypted but might need to encrypt certain traffic during the session. In these scenarios, an SSL handshake occurs mid-session to secure the specific traffic.

An example mid-session SSL is START Transport Layer Security (TLS) during SMTP sessions. The client starts an unencrypted SMTP session on port 25, to the server. The server accepts the unencrypted session and indicates to the client that it supports STARTTLS. In the middle of session, the client initiates STARTTLS. Normal TLS negotiations resume and the rest of the conversation is encrypted. For more details, go to http://en.wikipedia.org/wiki/STARTTLS.

RiOS 8.6 and later support TLS versions 1.0, 1.1, and 1.2. Note that the SteelHead can negotiate to earlier versions of TLS if the peer SteelHead uses a RiOS release prior to 8.6.

Figure 11-17. Example STARTTLS during SMTP session

```
S: <waits for connection on TCP port 25>
C: <opens connection>
S: 220 mail.example.org ESMTP service ready
C: EHLO client.example.org
S: 250-mail.example.org offers a warm hug of welcome
S: 250 STARTTLS
C: STARTTLS
S: 220 Go ahead
C: <starts TLS negotiation>
C & S: <negotiate a TLS session>
C & S: <check result of negotiation>
C: EHLO client.example.org[9]
. . .
```

Source: http://en.wikipedia.org/wiki/STARTTLS (May 3, 2012)

To enable mid-session SSL support

1. Choose Optimization > SSL: Advanced Settings.
2. Select Enable Midsession SSL.
3. Click **Apply**.

Server name indication

Server name indication (SNI) is an extension to the TLS protocol (RFC 6066) by which a client indicates which hostname it is attempting to connect to at the start of the TLS handshake process. This extension allows a server that hosts multiple hostnames (also known as *name-based virtual hosting*) using a single IP address to present the correct SSL/TLS certificate to the client. The hostname is the fully qualified DNS hostname of the server, as understood by the client.

When a client makes an HTTP connection to a server, the hostname is indicated in the HTTP request. However, when a client makes an HTTPS connection, the hostname is not sent until the secure connection is established. SNI allows a client to include the hostname that it is requesting in the TLS Hello message. This message allows the server to determine the correct named virtual host to service the request.

The following is an example from a packet capture illustrating the SNI in the client SSL-Hello message:

```
Extension: server_name
  Type: server_name (0x0000)
  Length: 30
  Server Name Indications extension
    Server Name list length: 28
    Server Name Type: host_name (0)
    Server Name: length: 25
    Server Name: nexus.officeaps.live.com
```

Without the SNI extension from a client application, the server returns the default certificate that matches the server IP. The challenge is that this certificate might or might not be the hostname that the client has requested. Browsers pop up a warning message when it receives a certificate with a different hostname than that it requested, and client-based applications might disconnect.

You must use client applications (for example, browsers) and servers that have end-to-end support for SNI. You must enable SNI on the server-side SteelHead if you want to intercept every HTTPS/TLS connection to determine the virtual hostname that the client is requesting in the SNI extension. Enabling SNI on the server side ensures that the server-side SteelHead selects and returns an acceptable proxy certificate to the client. No configuration is necessary on the client-side SteelHead.

In RiOS 9.2 and later, you do not need to enable SNI support for the following dependencies. The SteelHead always:

- forwards the SNI extension server name header from a client application (for example, the browser) to the origin server.
- checks the SNI extension field of client hello before a proxy certificate is returned to client.

  For more information, see https://supportkb.riverbed.com/support/index?page=content&id=S17744&actp=search.

You must enable SNI support on the Management Console to intercept every HTTPS/TLS connection to determine the virtual hostname that the client is requesting in the SNI extension.

To prevent unnecessary intercepts, we recommend that you bypass traffic from nonoptimizable hostnames by IP addresses using an in-path rule because intercepting the HTTPS/TLS connection to determine the virtual hostname consumes a connection count even if the connection is eventually bypassed. This consumption has an impact on the SteelHead licensed connection limit, especially in environments in which only a small percentage of HTTPS/TLS traffic is optimized and no bypass rules are configured for the rest of the traffic from the unoptimizable hostnames.

SSL optimization with SteelHead Mobile

SteelHead Mobile 2.0x and later support SSL optimization similar in function to that of a client-side SteelHead. SteelHead Mobile relies on the Mobile Controller to configure SSL optimization policies and secure peering. For more details on configuring SSL optimization with SteelHead Mobile, refer to the *SteelCentral Controller for SteelHead Mobile User Guide* and the *SteelHead Deployment Guide*.

Troubleshooting and verification

Use these tools to verify that you have configured SSL support correctly:

- **SSL Optimization** - After completing the SSL configuration on both SteelHeads and restarting the optimization service, access the secure server from the web browser. These events take place in a successful optimization:
  - If you specified a self-signed proxy certificate for the server on the server-side SteelHead, a pop-up window appears on the web browser. View the certificate details to ensure that it is the same as the certificate on the server-side SteelHead.
  - In the Management Console, the Current Connections report lists the new connection as optimized without a **Protocol Error** flag.
  - In the Management Console, the Traffic Summary report displays encrypted traffic (typically, HTTPS).
  - Verify that the back-end server IP appears in the SSL Discovered Server Table (Optimizable) in the SSL Main Settings page.

Note: Because all the SSL handshake operations are processed by the server-side SteelHead, all the SSL statistics are reported on the server-side SteelHead. No SSL statistics are reported on the client-side SteelHead.

- **Monitoring SSL Connections** - Use these tools to verify SSL optimization and to monitor SSL progress:
  - On the client web browser, click the **Lock** icon to obtain certificate details. The certificate must match the proxy certificate installed on server-side SteelHead.
  - In the Current Connections report, verify the destination IP address, port **443**, the Connection Count as **Established** (three yellow arrows on the left side of the table), **SDR Enabled** (three cascading yellow squares on the right side of the table), and that there is no **Protocol Error** (a red triangle on the right side of the table).
  - In the SSL Statistics report (on the server-side SteelHead only) look for connection requests (established and failed connections), connection establishment rate, and concurrent connections.
- **Monitoring Secure Inner Channel Connections** - Use these tools to verify that secure inner channels are in use for the selected application traffic types:
  - In the Current Connections report, look for the **Lock** icon and three yellow arrows, which indicate the connection is encrypted and optimized. If the Lock icon is not visible or is dimmed, click the magnifying glass to view a failure reason that explains why the SteelHead is not using the secure inner channel to encrypt the connection. If there is a red protocol error, click the magnifying glass to view the reason for the error.
  - Search the client-side and server-side SteelHead logs for **ERR** and **WARN**.

– Check that both SteelHeads appear in the white peering trust list on the client-side and server-side SteelHeads, indicating that they trust each other.

If you are experiencing issues with your SSL traffic being optimized, see the Riverbed Knowledge Base article *Troubleshooting your SSL Configuration* at http://supportkb.riverbed.com/support/index?page=content&id=S15107.

Interacting with SSL-Enabled web servers

This section describes how to obtain the server certificate and private key on two web servers: Apache and Microsoft IIS. This section includes the following topics:

- "Obtaining the server certificate and private key" on page 193
- "Generating self-signed certificates" on page 195

Obtaining the server certificate and private key

SSL is a protocol that enables the underlying application to transmit data securely over an insecure network. At the very foundation of SSL is the assumption that an authenticated party (for example, a web server) has exclusive access to its private key. If any other entity has this private key, it can mount a man-in-the-middle attack on a connection to the authenticated party.

SteelHeads optimize SSL traffic when you configure the server-side SteelHead with the server's certificate and private key, which enables it to intercept all SSL connections to the server.

Apache certificates and private keys

The following procedures explain how to locate the Apache server certificate and private key and import them into the server-side SteelHead.

To obtain the server certificate and private key from an Apache-based web server

1. Locate the Apache httpd.conf configuration file.
2. Look through the file for lines beginning with SSLCertificateFile and SSLCertificateKeyFile, for example:

```
SSLCertificateFile /etc/foo/bar/server.crt
SSLCertificateKeyFile /etc/foo/bar/server.key
```

   The filename following SSLCertificateFile is the server certificate. The filename following SSLCertificateKeyFile is the server private key. After you locate these files, you can import them into the server-side SteelHead configuration.

To import the certificate and private key

1. On the server-side SteelHead, choose Optimization > SSL: SSL Main Settings in the Management Console.
2. Select Add a New SSL Certificate.

3. Select Import Existing Private Key and CA-Signed Public Certificate (Two Files in PEM or DER formats).
4. Under Import Private Key, select Local File, click **Browse**, and go to the certificate key file.
5. Under Import Public Certificate, select Local File, click **Browse**, and go to the server certificate file.
6. Click **Add**.

IIS certificates and private keys

The following procedures explain how obtain the server certificate and private key from an IIS web server and import them into the server-side SteelHead.

To obtain the server certificate and private key from an IIS web server

1. From the Windows Start > Run menu, enter **mmc** to launch the Microsoft Management Console (MMC).
2. Within the IIS snap-in, go through the tree to the web server in question. (If the IIS snap-in does not exist, choose File > Add/Remove Snap-in, select the web server, and click **Add**.)
3. Right-click the server item and select Properties.
4. Select the **Directory Security** tab.
5. Select **View Certificate**.
6. Select the **Details** tab.
7. Select **Copy to File**.
8. Select Yes, export private key.

Both the certificate and the private key are now stored in a single file with the filename you specified. The filename ends with the .pfx extension.

To import the certificate and private key

1. On the server-side SteelHead, choose Optimization > SSL: SSL Main Settings in the Management Console.
2. Under SSL Server Certificates, select Add a New SSL Certificate.
3. Select Import Existing Private Key and CA-Signed Public Certificate (One File in PEM or PKCS12 formats).
4. Under Import Single File, select Local File, click **Browse**, and go to the .pfx file.

   Because the file .pfx file is not scrambled with a password, you can leave the Decryption Password field blank.

5. Click **Add**.

Generating self-signed certificates

In certain situations you might not want to, or might not be able to, use the server's real private key. If that is the case, you can generate a self-signed certificate and private key pair for the server and install them on the server-side SteelHead. This certificate is not signed by any real certificate authority, but it is instead signed by the private key itself, and is thus called a self-signed certificate.

During SSL connection establishment, when the server-side SteelHead presents the self-signed certificate to the client (for example, a web browser), the client cannot verify the authenticity of the certificate. From the client's point of view, security might have been compromised, and you are typically alerted with a message to this effect.

Generating self-signed certificates with Apache

A typical SSL-enabled Apache installation comes with a utility called OpenSSL, which you can use to generate the self-signed certificate. Enter the following command:

```
$ openssl req -new -x509 -nodes -out server.crt -keyout server.key
```

This command adds two files to the current directory, server.crt and server.key. These files correspond to the certificate and the private key, respectively. The next step is to import the files into the server-side SteelHead configuration.

To import the certificate and private key

1. On the server-side SteelHead, SteelHead, choose Optimization > SSL: SSL Main Settings in the Management Console.
2. Under SSL Server Certificates, select Add a New SSL Certificate.
3. Select Import Existing Private Key and CA-Signed Public Certificate (Two Files in PEM or DER formats).

   Because the file server.key is not scrambled with a password, you can leave the Decryption Password field blank.
4. Click **Add**.

Generating self-signed certificates with IIS

If you want to generate a self-signed certificate for an IIS-based web server, you have two options.

To generate self-signed certificates with IIS

- Install Cygwin and include the OpenSSL package in the installation. This package gives you access to the OpenSSL utility from X.2.1, which you can use to generate the certificate and private key. To install, go to http://www.cygwin.com/.

  —or—

- Download and install a set of IIS 6.0 Resource Kit Tools from Microsoft at http://www.microsoft.com/en-us/download/details.aspx?id=17275.

  This package contains a utility called SelfSSL that you can use to generate a self-signed certificate and private key for a web server. SelfSSL also automatically installs the certificate for that web server instance of IIS, so you must follow the steps in X.1.2 to extract the certificate into a file.

  SelfSSL replaces an existing certificate for a web server instance.

Deploying a SteelHead with a SafeNet Network HSM

Starting with RiOS 9.2, you can use a Gemalto SafeNet Network Hardware Security Module (HSM) to store SSL and TLS cryptographic keys, instead of storing the certificates in the secure vault on the SteelHead. Because all certificates can be stored on the HSM, this deployment offers security and management benefits.

Figure 11-18 shows a deployment with an HSM connected to a server-side SteelHead.

Figure 11-18. Example of using a SafeNet Network HSM in a network

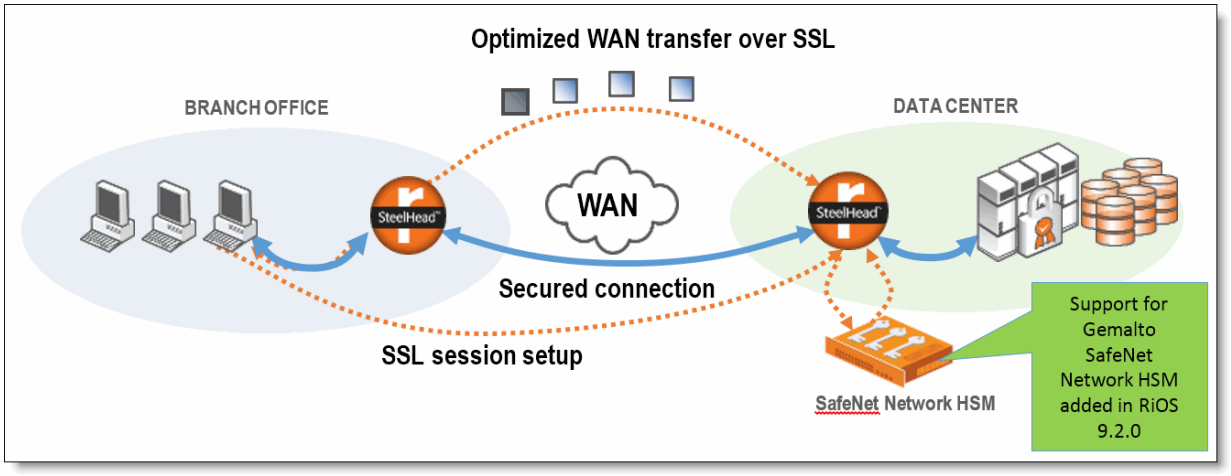
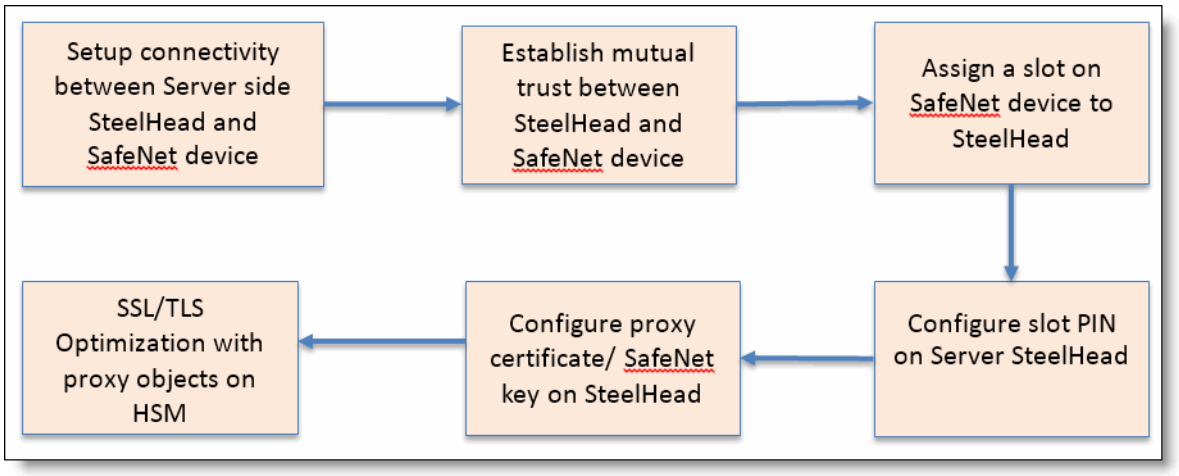


Figure 11-19 lists the tasks you perform to deploy an HSM with a server-side SteelHead. The complete procedure is in "To deploy a SteelHead with a SafeNet Network HSM" on page 198.

Figure 11-19. Deploying an HSM with a server-side SteelHead - summary steps



For more information about the SteelHead commands used in this deployment, see the *Riverbed Command-Line Interface Reference Manual*.

Requirements

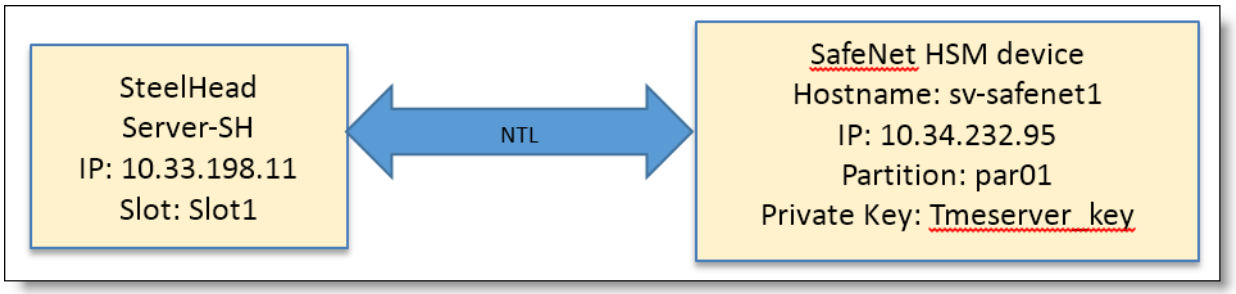
- The server-side SteelHead must be running RiOS 9.2 or later.
- The HSM must be able to communicate with the SteelHead's Primary or Aux interface. We recommend that this connection have a latency at LAN speed (around 1 ms or less). Longer latencies could affect the time taken for setup and authentication of optimized SSL connections.
- We recommend placing the server-side SteelHead and HSM in the same LAN so that the SSL handshake between the server-side SteelHead and the clients can be completed quickly, which reduces or eliminates any performance impact.
- The required private keys must already be added and present on the SafeNet Network HSM. Administrators configuring this deployment should be aware of the partition number on the SafeNet device that contains the private keys that are accessed by the SteelHead.
- This deployment requires command-line interface (CLI) access to the server-side SteelHead appliance and the SafeNet Network HSM.

Limitations

- This deployment is supported for the Gemalto SafeNet Network HSM (formerly SafeNet Luna HSM) only. No other HSMs are supported.
- This deployment is compatible with SafeNet Network HSMs running SafeNet Client software versions 5.3.5 and 5.4.7. Other software versions have not been tested.
- SteelHead xx50 series devices are not supported.

The examples in this procedure use the IP addresses and network information shown in Figure 11-20. The Network Trust Link (NTL) in this figure is a secure, authenticated network connection between the SafeNet Network HSM and its client, which uses two-way digital certificate authentication and SSL data encryption.

Figure 11-20. IP address and network information used in this procedure



To deploy a SteelHead with a SafeNet Network HSM

1. Open a CLI session with the server-side SteelHead.
2. Generate a client certificate for the SteelHead by entering the **protocol ssl hsm safenet generate-cert name** {<**hostname**> | <**ip-address**>} command, where **hostname** or **ip-address** is the hostname or IP address of the SteelHead.

   Figure 11-21 shows the certificate being generated using an IP address.

Figure 11-21. Generating a certificate with an IP address

```
Server-SH # conf t
Server-SH (config) # protocol  ssl hsm safenet generate-cert name ?
<string>         hostname/ip address of this SH accessible from HSM device
Server-SH (config) # protocol  ssl hsm safenet generate-cert name 10.33.198.11
Successfully created certificate with common name: "10.33.198.11".

-----BEGIN CERTIFICATE-----
MIIDYzCCAkugAwIBAgIBADANBgkqhkiG9w0BAQsFADB1MQswCQYDVQQGEwJVUzET
MBEGA1UECBMKQ2FsaWZvcm5pYTEWMBQGA1UEBxMNU2FuIEZyYW5jaXNjbzEiMCAG
A1UEChMZUml2ZXJiZWQgVGVjaG5vbG9neSwgSW5jLjEVMBMGA1UEAxMMMTAuMzMu
MTk4LjExMB4XDTE2MDMwNzAwNDQyNloXDTI2MDMwNjAwNDQyNlowdTELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNhbiBGcmFuY2lz
Y28xIjAgBgNVBAoTGVJpdmVyYmVkIFRlY2hub2xvZ3ksIEluYy4xFTATBgNVBAMT
DDEwLjMzLjE5OC4xMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALk3
MRDlxtnFDKjijsP8OZphYt9VifInBM6Nr8VuEsYtKy/h8rfBw/pBCrFe4xqLDx6b
4gxViKC/bT0yHcuVF2bCyovlAlm1+f6YkKa0mnu3XLpT5REQ9No6xgFQbr8fKabV
G+X4MXtjk9eySwDfkAAZj2isqkof39OYDBRTgIGbZUe0mCTFNQa9beiIUCAkggjd
XYq0kYTTzKwpQ6TfxBPjpaXBdiKk3k9PDYb4FzOSru9Q0ZL0P7UHOr5/9pKiai/K
Gn0yESWDiwlBZJ9QjGVHM1cAF9yuu4Bz9bmJtcwVqU/iAvcX9S1ataK2O72BaOUI
orNe5gKDE/isVxYxnM8CAwEAATANBgkqhkiG9w0BAQsFAAOCAQEAEf6amwLOWX0B
SElTbDnNeuQc8sv0s3PDHcNyVMx8/rCfcqDyaBItqQtmwkX+yjnttXnvTJo3DK+E
P5/ChtHJqjWiGi4rxKJ58LeUal+YIViDL3sMsPKcm5kVKLlI8ncW7iHShrV//5B1
qRi7tTphpdYdgQ+XkQ052nrwPy8xs7HQhaLFTVUNIOfLzu0MlFCcPWKc+Ox9jaxS
iraoYJPu5pfOqnUtEfRmvY82YCHmxPhjV1pBoOBG7tLF2d9vrP4OcSyOOK2DpY1x
TZ17EXMUDPp8Wenut9ZkXibr1Gyp+qG5Ze4mE+uz4cU9z83ftexr1yR9yxYshYR+
WyrVSn5AoQ==
-----END CERTIFICATE-----
```

IP address being used in this example

3. From the server-side SteelHead, export the SteelHead client to the HSM by entering the **protocol ssl hsm safenet export-cert** command.

Figure 11-22 shows the generated certificate.

Figure 11-22. Exporting and viewing the certificate

```
Server-SH (config) # protocol  ssl hsm safenet export-cert
-----BEGIN CERTIFICATE-----
MIIDYzCCAkugAwIBAgIBADANBgkqhkiG9w0BAQsFADB1MQswCQYDVQQGEwJVUzET
MBEGA1UECBMKQ2FsaWZvcm5pYTEWMBQGA1UEBxMNU2FuIEZyYW5jaXNjbzEiMCAG
A1UEChMZUml2ZXJiZWQgVGVjaG5vbG9neSwgSW5jLjEVMBMGA1UEAxMMMTAuMzMu
MTk4LjExMB4XDTE2MDMwNzAwNDQyNloXDTI2MDMwNjAwNDQyNlowdTELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNhbiBGcmFuY2lz
Y28xIjAgBgNVBAoTGVJpdmVyYmVkIFRlY2hub2xvZ3ksIEluYy4xFTATBgNVBAMT
DDEwLjMzLjE5OC4xMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALk3
MRDlxtnFDKjijsP8OZphYt9VifInBM6Nr8VuEsYtKy/h8rfBw/pBCrFe4xqLDx6b
4gxViKC/bT0yHcuVF2bCyovlAlm1+f6YkKa0mnu3XLpT5REQ9No6xgFQbr8fKabV
G+X4MXtjk9eySwDfkAAZj2isqkof39OYDBRTgIGbZUe0mCTFNQa9beiIUCAkggjd
XYq0kYTTzKwpQ6TfxBPjpaXBdiKk3k9PDYb4FzOSru9Q0ZL0P7UHOr5/9pKiai/K
Gn0yESWDiwlBZJ9QjGVHM1cAF9yuu4Bz9bmJtcwVqU/iAvcX9S1ataK2O72BaOUI
orNe5gKDE/isVxYxnM8CAwEAATANBgkqhkiG9w0BAQsFAAOCAQEAEf6amwLOWX0B
SE1TbDnNeuQc8sv0s3PDHcNyVMx8/rCfcqDyaBItqQtmwkX+yjnttXnvTJo3DK+E
P5/ChtHJqjWiGi4rxKJ58LeUal+YIViDL3sMsPKcm5kVKLlI8ncW7iHShrV//5B1
qRi7tTphpdYdgQ+XkQ052nrwPy8xs7HQhaLFTVUNI0fLzu0MlFCcPWKc+Ox9jaxS
iraoYJPu5pfOqnUtEfRmvY82YCHmxPhjV1pBoOBG7tLF2d9vrP4OcSyOOK2DpY1x
TZ17EXMUDPp8Wenut9ZkXibr1Gyp+qG5Ze4mE+uz4cU9z83ftexr1yR9yxYshYR+
WyrVSn5AoQ==
-----END CERTIFICATE-----
Server-SH (config) #
Server-SH (config) #
```

4. Copy the server-side SteelHead certificate text and save it as a PEM file, naming it with the same hostname or IP address that you used with the **protocol ssl hsm safenet generate-cert** command.

   Enter this command on any host that can connect to the HSM.

   Figure 11-23 creates a PEM file that uses the server-side SteelHead IP address (10.33.198.11) as its name.

Figure 11-23. Generating a PEM file

```
C:\Users\sapmehta\Downloads>pscp ..\Desktop\HSM\10.33.198.11.pem admin@sv-safene
t1.lab.nbttech.com:
admin@sv-safenet1.lab.nbttech.com's password:

10.33.198.11.pem           | 1 kB |   1.2 kB/s | ETA: 00:00:00 | 100%

C:\Users\sapmehta\Downloads>
```

5. From the host, copy the certificate from the HSM to a host using the Linux **scp** command.

Figure 11-24. Using the scp command to copy the HSM certificate to a host

```
[root@oak-vcs406 ~]# scp admin@sv-
safenet1.lab.nbttech.com:server.pem .

admin@sv-safenet1.lab.nbttech.com's password:
```

6. From the host, open the server.pem file and copy the contents of the file using the Linux **cat** command.

Figure 11-25. Copying certificate contents from the host



7. From the server-side SteelHead, copy the certificate from the host to the server-side SteelHead using the **protocol ssl hsm safenet hsm-server name** <**hsm-device-name**> **import-cert "**<**cert-data**>**"** command.

   Note the quotation marks around the certificate data in Figure 11-26.

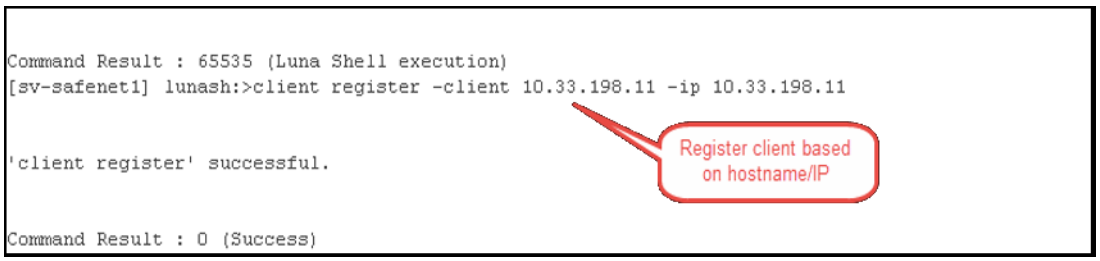
Figure 11-26. Importing the HSM certificate to the server-side SteelHead



8. From the HSM, enter the following command to register the server-side SteelHead with the HSM:

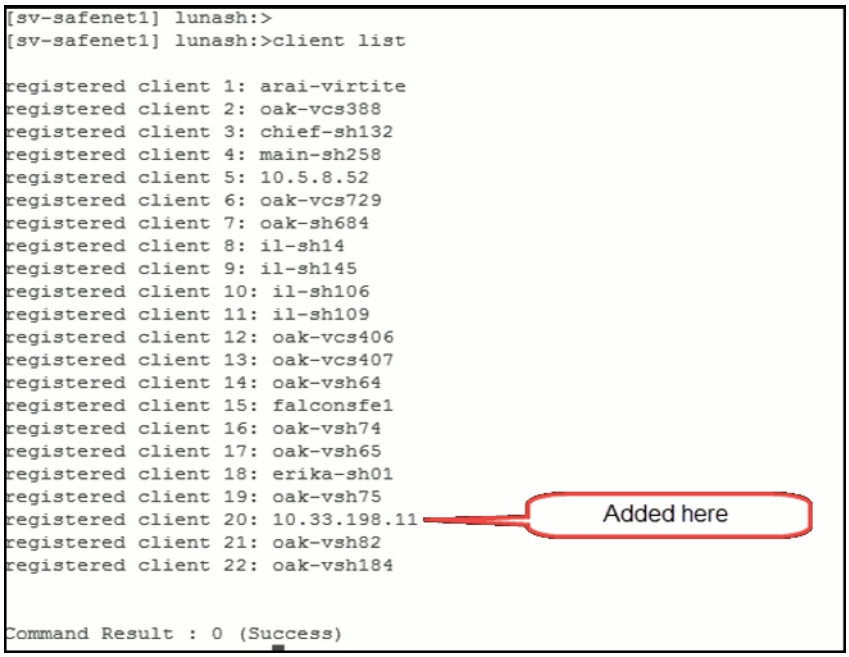
```
Client register -client <server-side-steelhead-nickname-or-ip-address> {-hostname <server-side steelhead hostname | -ip <server-side-steelhead-ip-address>}
```

Figure 11-27. Registering the server-side SteelHead with the HSM



9. From the HSM, verify that the server-side SteelHead has been registered by entering the **client list** command.

Figure 11-28. Verifying SteelHead registration on the HSM



10. From the HSM, assign a partition for the server-side SteelHead so that the SteelHead can access objects in that partition.

    Create a new partition slot, or use an existing slot. If you use an existing slot, specify the slot on which you stored the required private keys.

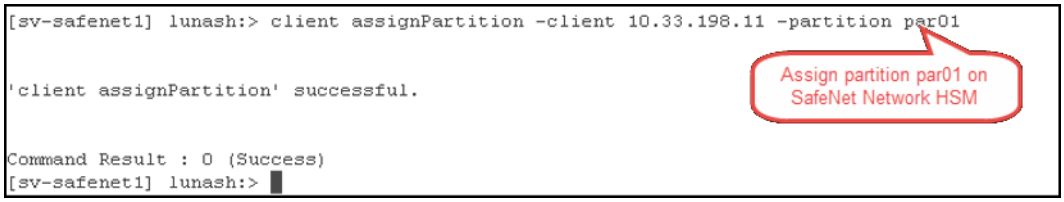
    Use the following command:

    ```
    client assignPartition -client <client-name> -partition <partition-number>
    ```

Note: A single SafeNet Network HSM can be separated into 100 cryptographically isolated partitions, with each partition functioning as if it was an independent HSM. Partitioning provides scalability and flexibility, as a single HSM can protect the cryptographic keys of hundreds of independent applications concurrently. In addition, the ability to assign a unique Partition Security Officer to each partition means the configurations of partitions and control over cryptographic keys can be strictly enforced, even in public cloud environments. For service providers, partitions can be offered as rentable services and your customers can maintain the trust and confidence that only they have access to their partition and sensitive cryptographic keys.

Figure 11-29 assigns a partition named par01 on the HSM.

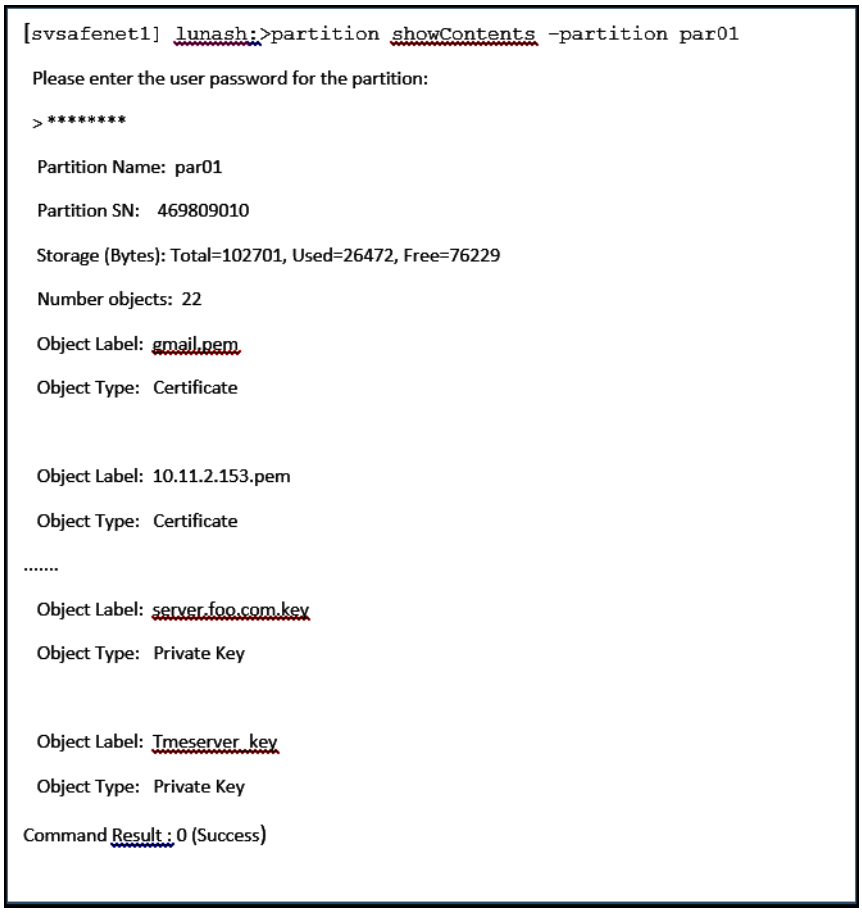
Figure 11-29. Assigning a partition for the server-side SteelHead



11. On the HSM, verify the presence of the private key by entering the **partition showContents -partition <partition-number>** command.

    You added and stored this private key before starting this procedure.

Figure 11-30. Verifying the private key



12. From the server-side SteelHead, determine the partition location on the HSM by entering the **show protocol ssl hsm safenet** command.

    The slot number and label name are shown in Figure 11-31.

Figure 11-31. Finding the partition for the SteelHead on the HSM

```
Server-SH # show protocol ssl hsm safenet
Registerd Safenet HSM device list:

Server: 10.34.232.95    HTL required: no


Connected SafeNet HSM device info:

The following Luna SA Slots/Partitions were found:

Slot    Serial #        Label
====    ========        =====
 1      469809010       par01
```

13. From the HSM, specify the slot number and slot PIN (password) for the partition that was created for the server-side SteelHead by entering the **protocol ssl hsm slot <slot-number> slot pin <password>** command.

Note: All slots assigned to the server-side SteelHead must use the same PIN.

Figure 11-32. Specifying the slot number and password for the partition

```
Server-SH (config) #
Server-SH (config) # protocol ssl hsm slot 1  slot-pin S@fenet1
Updated slot "1" pin successfully.
Server-SH (config) # no protocol ssl hsm server-cert name *.ad.riverbedtme.com
Server-SH (config) #
```

14. From the server-side SteelHead, configure a proxy certificate and corresponding private key object on the HSM by entering the **protocol ssl hsm server-cert name <name> import-cert "<proxy-certificate-text>" key-slot <slot-number>** [**key-label <key-label>**] [**key-id <key-id>**] command.

Figure 11-33 specifies a certificate name of “sapna-server”. Note the quotation marks around the certificate text, and note the key slot and key label specified after the certificate text.

Figure 11-33. Configuring a proxy certificate the private key object on the HSM



Verifying deployment of a SteelHead with an HSM

From the server-side SteelHead, enter the **show protocol ssl hsm server-certs** command to verify that the key is accessible and the deployment is correctly set up.

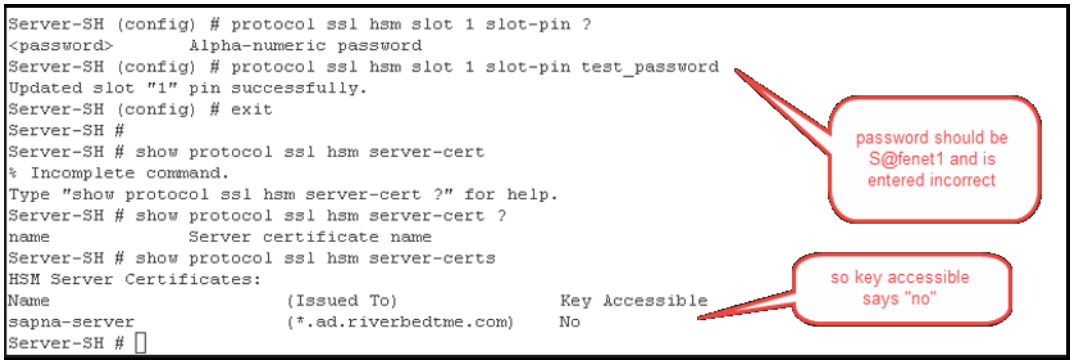
Figure 11-34. Verifying HSM-SteelHead deployment



Troubleshooting HSM and SteelHead deployments

Symptom: The output of the **show protocol ssl hsm server-certs** command shows that the key is not accessible.

Figure 11-35. Command output shows key as inaccessible



Condition: The PIN (password) for the HSM slot might be incorrect, the private key could not be loaded, or there might be a configuration error.

Workaround: Open the SteelHead Management Console and choose Reports > Diagnostics: System Logs and then perform one of the following steps:

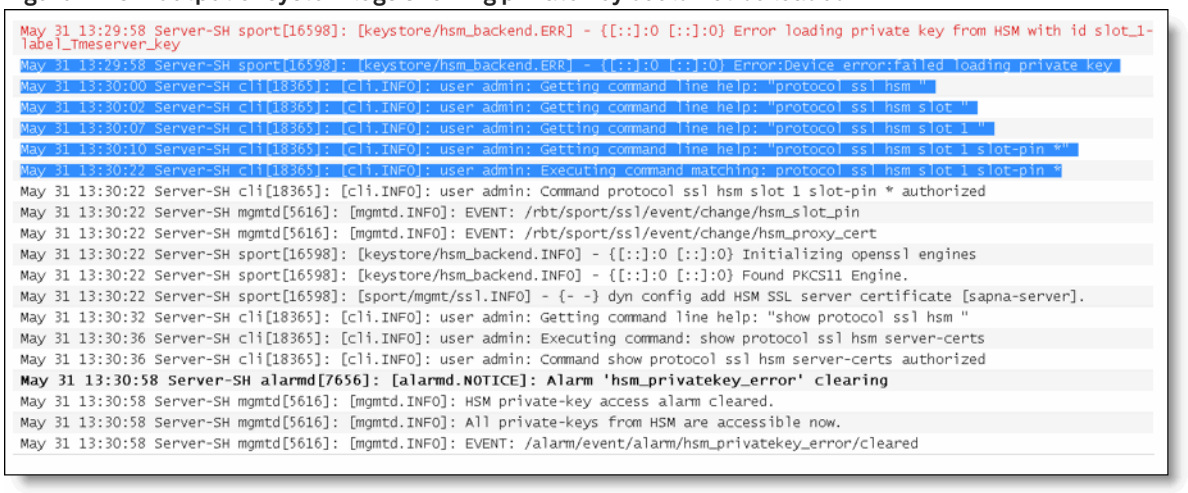
- If you see a “PIN incorrect” error, check that you entered the PIN for the HSM slot that is accessible to the SteelHead.

Figure 11-36. Output of system logs showing incorrect password



- Read the logs and check your steps to find the source of the problem. An “Error loading private key” error indicates that there is some problem with initial configuration.

Figure 11-37. Output of system logs showing private key could not be loaded



12

Configuring SCEP and Managing CRLs

This chapter describes how to configure the Simple Certificate Enrollment Protocol (SCEP) and how to manage Certification Revocation Lists (CRLs) using the Riverbed CLI. This chapter includes the following sections:

- "Using SCEP to configure on-demand and automatic reenrollment" on page 207
- "Managing Certificate Revocation Lists" on page 211

This section makes the following assumptions:

- You have configured SSL on the SteelHead (for details, see "SSL Deployments" on page 169).
- You have set up a SCEP server.

Using SCEP to configure on-demand and automatic reenrollment

SCEP is for securely issuing and revoking digital certificates in a simple, scalable manner on network devices. The SteelHead uses SCEP to configure on-demand enrollment and automatic reenrollment of SSL peering certificates.

Note: Currently, the SteelHead can only enroll peering certificates.

This section describes how to configure on-demand and automatic reenrollment of SSL peering certificates.

The following table summarizes the SCEP commands.

SCEP commands	Parameters	Definition
secure-peering scep auto-reenroll	enable	Enables automatic reenrollment of a certificate to be signed by a CA.
	exp-threshold <num-of-days>	Specify the amount of time (in days) to schedule reenrollment before the certificate expires.
	last-result clear-alarm	Clears the automatic reenrollment last-result alarm. The last result is the last completed enrollment attempt.

SCEP commands	Parameters		Definition
secure-peering scep max-num-polls	<max-number-polls>		Specify the maximum number of polls before the SteelHead cancels the enrollment. The peering certificate is not modified. The default value is 5. A poll is a request to the server for an enrolled certificate by the SteelHead. The SteelHead polls only if the server responds with pending. If the server responds with fail, then the SteelHead does not poll.
secure-peering scep on-demand cancel	None		Cancels any active on-demand enrollment.
secure-peering scep on-demand gen-key-and-csr	rsa		Generates a new private key and CSR for on-demand enrollment using the Rivest-Shamir-Adleman algorithm.
		state <string>	Specify the state. No abbreviations allowed.
		org-unit <string>	Specify the organizational unit (for example, the department).
		org <string>	Specify the organization name (for example, the company).
		locality <string>	Specify the city.
		email <email-address>	Specify an email address of the contact person.
		country <string>	Specify the country (two-letter code only).
		common-name <string>	Specify the hostname of the peer.
		key-size {512 | 1024 | 2048}	Specify the key size in bits (for example, 512, 1024, 2048).
secure-peering scep on-demand start			Starts an on-demand enrollment (in the background by default).
	foreground		Starts an on-demand enrollment in the foreground.
secure-peering scep passphrase	<pass-phrase>		Specify the challenge password phrase.
secure-peering scep poll-frequency	<minutes>		Specify the poll frequency in minutes. The default value is 5.
secure-peering scep trust	peering-ca <peer-ca>		Specify the name of the existing peering CA.
secure-peering scep url	<url>		Specify the URL of the SCEP responder. Use the following format: http://host[:port/path/to/service].

Configuring on-demand enrollment

The following example configures the most common on-demand enrollment SCEP settings.

Note: You can only perform one enrollment of a certificate at a time. You must stop enrollment before you begin the enrollment process for another certificate.

To configure on-demand enrollment of certificates

1. To configure SCEP settings, connect to the SteelHead CLI and enter the following commands:

```
enable
configure terminal
secure-peering scep url <http://host[:port/path/to/service>
secure-peering scep trust peering-ca <name>
secure-peering scep poll-frequency 10
secure-peering scep max-num-polls 6
secure-peering scep passphrase "<device-unique-passphrase>"
```

2. To perform an on-demand enrollment you must first generate a new key and Certificate Signing Request (CSR), at the system prompt enter the command:

```
secure-peering scep on-demand gen-key-and-csr rsa 1048 country us org mycompany org-unit
engineering
```

3. To display the CSR (including the fingerprint), at the system prompt enter the command:

```
show secure-peering scep peering on-demand csr
```

4. To start an on-demand enrollment, at the system prompt enter the command:

```
secure-peering scep on-demand start
```

5. To view current status and the result of the last attempt (since boot), at the system prompt enter the following commands:

```
show secure-peering scep enrollment status
show secure-peering scep on-demand last-result
```

6. To stop enrollment, at the system prompt enter the following commands:

```
secure-peering scep on-demand cancel
show secure-peering scep on-demand last-result
```

   You must stop enrollment before you can begin the enrollment process for another certificate.

Configuring automatic reenrollment

The following example configures the most common automatic reenrollment SCEP settings.

To configure automatic reenrollment of certificates

1. To configure SCEP settings, connect to the SteelHead CLI and enter the following commands:

```
enable
configure terminal
secure-peering scep url http://entrust-connector/cgi-bin/pkiclient.exe
secure-peering scep trust peering-ca <name>
secure-peering scep poll-frequency 10
```

```
secure-peering scep max-num-polls 6
secure-peering scep passphrase "<device-unique-passphrase>"
```

2. To configure automatic reenrollment, at the system prompt enter the following commands:

```
secure-peering scep auto-reenroll exp-threshold 30
secure-peering scep auto-reenroll enable
```

3. To view current automatic reenrollment settings, at the system prompt enter the following commands:

```
show secure-peering scep peering auto-reenroll csr
show secure-peering scep peering on-demand last-result
```

Viewing SCEP settings and alarms

This section describes how view SCEP settings and alarms.

The following table summarizes the commands for SCEP settings.

Command	Parameters	Definition
show secure-peering scep	None	Displays SCEP information.
show secure-peering scep auto-reenroll	csr	Displays the automatic reenrollment CSR.
	last-result	Displays the result of the last completed automatic reenrollment.
show secure-peering scep ca	<ca-name> certificate	Displays a specified SCEP peering CA certificate.
show secure-peering scep enrollment status	None	Displays enrollment status information.
show secure-peering scep on-demand	csr	Displays on-demand enrollment information.
	last-result	Displays result of the last completed on-demand enrollment.

An SCEP alarm is triggered when the SteelHead requests an SCEP server to dynamically reenroll an SSL peering certificate and the request fails. The SteelHead uses SCEP to dynamically reenroll a peering certificate to be signed by a certificate authority. The alarm clears automatically when the next automatic reenrollment succeeds.

To view SCEP alarm status

1. Connect to the SteelHead CLI and enter enable mode.

2. Enter the following the command:

```
show stats alarm ssl_peer_scep_auto_reenroll
Alarm ssl_peer_scep_auto_reenroll:
    Enabled:                  yes
    Alarm state:              ok
    Rising error threshold:   no
    Rising clear threshold:   no
    Falling error threshold:  no
```

```
Falling clear threshold:   no
Rate limit bucket counts:  { 5, 20, 50 }
Rate limit bucket windows: { 3600, 86400, 604800 }
Last checked at:           2009/07/30 17:43:07
Last checked value:        true
Last event at:
Last rising error at:
Last rising clear at:
Last falling error at:
Last falling clear at:
```

To clear the SCEP alarm

1. Connect to the SteelHead CLI and enter configuration mode.
2. Enter the following the command:

```
secure-peering scep auto-reenroll last-result clear-alarm
```

Managing Certificate Revocation Lists

Certificate Revocation Lists (CRLs) allow CAs to revoke issued certificates (for example, when the private key of the certificate is compromised).

Note: CRLs are not used by default in the SteelHead.

A CRL is a database that contains a list of digital certificates that have been invalidated before their expiration date, including the reasons for the revocation, and the names of the issuing certificate signing authorities. The CRL is issued by the CA that issues the corresponding certificates. All CRLs have a lifetime during which they are valid (often 24 hours or less).

CRLs are used when a:

- server-side SteelHead verifies the certificate presented by the server in the SSL handshake between the server-side SteelHead and the server.
- server-side SteelHead verifies the certificate presented by the client-side SteelHead in the handshake between the two SteelHeads for establishing a secure inner channel over the WAN.
- client-side SteelHead verifies the certificate presented by the server-side SteelHead in the handshake between the two SteelHeads for establishing a secure inner channel over the WAN.

Note: Currently, the SteelHead only supports downloading CRLs from Lightweight Directory Access Protocol (LDAP) servers.

The following table summarizes CRL management commands.

CRL commands	Parameters		Definition
protocol ssl crl ca			Configures CRL for automatically discovered CAs. You can update automatically discovered CRLs using this command.
	<ca-name>		Specify the name of an SSL CA certificate.
	cdp <integer>		Specify an integer index of a Cisco Distribution Point (CDP) in a CA certificate. The no protocol ssl crl ca * cdp * command option removes the update.
		ldap server {<ip-address> | <hostname>}	Specify the LDAP server IP address or hostname to modify a CDP URI.
		port <port>	Optionally, specify the LDAP service port.
		crl-attr-name <attr-name>	Optionally, specify the attribute name of CRL in a LDAP entry.
protocol ssl crl cas enable			Enables CRL polling and use of CRL in handshake verifications of CA certificates. Enabling CRL allows the CA to revoke a certificate. For example, when the private key of the certificate is compromised, the CA can issue a CRL that revokes the certificate.
protocol ssl crl handshake			Configures handshake behavior for a CRL.
	fail-if-missing		If a relevant CRL cannot be found the handshake fails.
[no] protocol ssl crl manual	ca		Specify the CA name to manually configure the CDP. The no protocol ssl crl manual command removes manually configured CDPs.
		uri <uri>	Specify the complete CDP URI to manually configure the CDP for the CA.
	peering ca		Specify the CA name to manually configure the CDP for the peering CA.
		uri <uri>	Specify the complete CDP URI to manually configure the CDP for the peering CA.

CRL commands	Parameters		Definition
protocol ssl crl peering	ca <ca-name>		Configures a CRL for an automatically discovered peering CA.
		cdp <integer>	Specify an integer index of a cdp in a peering CA certificate. The no protocol ssl crl peering ca * cdp * command removes the update.
		ldap server {<ip-address> | <hostname>}	Specify the IP address or hostname of a LDAP server.
		crl-attr-name {<string> | port <port-number>}	Optionally, specify an attribute name of CRL in a LDAP entry.
		port <port-number>	Optionally, specify the LDAP service port.
	cas enable		Enables CRL polling and use of CRL in handshake verification.
protocol ssl crl query-now	ca <string> cdp <integer>		Download CRL issued by SSL CA. Specify the CA name and CDP integer.
	peering ca <ca-name> cdp <integer>		Download CRL issued by SSL peering CA. Specify the CA name and CDP integer.
show protocol ssl crl	ca <ca-name>		Display current state of CRL polling of a CA.
	cas crl-file <string> text		Display information about the specified CRL file in text format.
	report ca <ca-name>		Display reports of CRL polling from the CA.

Managing CRLs

This section describes how to manage CRLs using the CLI.

To update an incomplete CDP

1. To enable CRL polling and handshakes, connect to the SteelHead CLI and enter configuration mode.
2. Enter the following set commands:

```
protocol ssl crl cas enable
protocol ssl crl peering cas enable
```

3. To view the CRL polling status of all CAs, enter the following command:

```
show protocol ssl crl ca cas
<<This example lists two CDPs: one complete CDP and one incomplete CDP.>>
  CA: Comodo_Trusted_Services
   CDP Index: 1
     DP Name 1: URI:http://crl.comodoca.com/TrustedCertificateServices.crl
```

```
      Last Query Status: unavailable
    CDP Index: 2
      DP Name 1: URI:http://crl.comodo.net/TrustedCertificateServices.crl
      Last Query Status: unavailable
<<An incomplete CDP is indicated by the DirName format.>>
  CA: Entrust_Client
    CDP Index: 1
      DP Name 1: DirName:/C=US/O=Entrust.net/OU=www.entrust.net/Client_CA_Info/CPS incorp. by
  ref.limits liab./OU=(c) 1999 Entrust.net Limited/CN=Entrust.net Client Certification
Authority
  CN=CRL1
      Last Query Status: unavailable
    CDP Index: 2
      DP Name 1: URI:http://www.entrust.net/CRL/Client1.crl
      Last Query Status: unavailable
```

In this case, the Entrust Client is an incomplete CDP as indicated by **DirName** format. Currently, the SteelHead only supports updates in the **DirName** format.

4. To update the incomplete CDP URI, enter the following commands:

```
protocol ssl crl ca Entrust_Client cdp 1 ldap-server 192.168.172.1
protocol ssl crl peering ca Entrust_Client cdp 1 ldap-server 192.168.172.1
```

5. To view the status of the updated CDP, enter the following command:

```
show protocol ssl crl ca Entrust_Client
```

The status of CRL polling can be either **pending**, **success**, or **error.**

6. To check CRL polling status of all CAs, enter the following command:

```
show protocol ssl crl cas
```

Viewing CRL alarm status

This section describes how to view a CRL alarm and how to clear a CRL alarm.

To view CRL alarm status

1. Connect to the SteelHead CLI and enter enable mode.
2. Enter the following the command:

```
show stats alarm crl_error
Alarm crl_error:
    Enabled:                   yes
    Alarm state:               ok
    Rising error threshold:    1
    Rising clear threshold:    1
    Falling error threshold:   no
    Falling clear threshold:   no
    Rate limit bucket counts:  { 5, 20, 50 }
    Rate limit bucket windows: { 3600, 86400, 604800 }
    Last checked at:           2009/07/30 17:40:34
    Last checked value:        0
    Last event at:
Last rising error at:
    Last rising clear at:
    Last falling error at:
    Last falling clear at:
```

To clear a CRL alarm, you must either rectify the problem by updating the incomplete CDP or you must disable CRL polling.

To disable CRL polling and clear a CRL alarm

1. Connect to the SteelHead CLI and enter configuration mode.
2. Enter the following the command:

```
no protocol ssl crl cas enable
```