



CYREN

Outbound Anti-Spam Daemon (ctasd)

Integration Manual

Version 5.00, August 2017

© 2017 CYREN Inc.

All rights reserved

Trademark and Copyright Statement

CTT20-800-112-088-R2

© CYREN Inc. 2017 All rights reserved.

The information contained in this document is subject to change without notice. CYREN makes no warranty of any kind. CYREN will not be liable for any direct, indirect, incidental, consequential or other damage alleged in connection with the furnishing or use of this information. Except as allowed by copyright laws or herein, reproduction, adaptation or translation without prior written permission is prohibited. RPD™, ctasd™, and ctengine™ are trademarks of CYREN Inc. All other trade/service marks or names that may be referenced and/or mentioned in this document belong to their respective owners. Microsoft is a trademark and/or registered trademark of Microsoft Corp. Linux is a trademark of Linus Torvalds. Red Hat is a trademark of Red Hat, Inc. in the United States and other countries. Debian is a registered trademark of Software in the Public Interest, Inc.

COPYRIGHT AND PERMISSION NOTICE Copyright (c) 1996 - 2017, Daniel Stenberg, <daniel@haxx.se>. All rights reserved. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

Contacts

Any technical questions you or your developers have about using the ctasd should be addressed to support@cyren.com.

About CYREN

CYREN™ provides proven Internet security technology to more than 150 security companies and service providers including 1&1, Check Point, F-Secure, Google, Microsoft, Panda Security, Rackspace, US Internet, and WatchGuard, for integration into their solutions. CYREN's GlobalView™ and patented Recurrent Pattern Detection™ (RPD™) technologies are founded on a unique cloud-based approach, and protect effectively in all languages and formats. CYREN Antivirus utilizes a multi-layered approach to provide award winning malware detection and industry-leading performance.

CYREN technology automatically analyzes billions of Internet transactions in real-time in its global data centers to identify new threats as they are initiated, enabling our partners to protect end-users from spam and malware, and ensure safe, compliant browsing. The company's expertise in building efficient, mass-scale security services mitigate Internet threats for thousands of organizations and hundreds of millions of users in 190 countries.

CYREN, formerly known as Commtouch, was founded in 1991, is headquartered in the US in McLean, Virginia, with offices in Palo Alto, California, Herzliya, Israel, Berlin, Germany, and Reykjavik, Iceland.

For more information about enhancing security offerings with CYREN technology, visit our website at www.cyren.com, see our blog at <http://blog.cyren.com> or write to info@cyren.com.

Table of Contents

1	INTRODUCTION.....	1
1.1	Working with ctasd.....	2
1.2	ctasd Solution Components.....	4
1.3	System Architecture and Data Flow	5
1.3.1	White List and Blue List	7
1.4	Spam Classifications	7
1.5	Virus Outbreak Detection.....	8
1.6	Command Antivirus Protection	9
1.6.1	Command Antivirus Detection Types.....	9
1.6.2	Detection Accuracy Values.....	12
1.7	Synchronous vs. Asynchronous Communication.....	13
1.8	SpamAssassin-Compatible Network Protocol Option	13
1.9	SpamAssassin-Compatible Workflow	15
1.10	ctasd Deployment Options.....	16
1.11	ctasd Minimum Requirements	16
1.12	ctasd Package	17
1.13	Internal Directory Structure	17
1.13.1	RPM Structure	17
2	CTASD CONFIGURATION.....	19
2.1	Sample Configuration File.....	19
2.1.1	[General].....	27
2.1.2	[Connectivity]	28
2.1.3	[Security]	29
2.1.4	[HttpServer].....	29
2.1.5	[Spamd]	30
2.1.6	[AV].....	32
2.1.7	[Stats]	33
2.1.8	[Outbound].....	33
2.2	Optional Configuration Parameters.....	37
2.2.1	[Connectivity]	37
3	RUNNING CTASD	38
3.1	Command Line Options	38
3.2	Stopping ctasd	39
4	CTASD PROTOCOL	40
4.1	Request and Response Conventions	40
4.2	ClassifyMessage_File	41
4.2.1	Method: ClassifyMessage_File	41
4.2.2	ClassifyMessage_File Response.....	45
4.3	Method: ClassifyMessage_Inline.....	47

4.3.2	Sample HTTP Request with Streaming	48
4.4	Response to ClassifyMessage_Inline Request	49
4.5	Method: UpdateSenderIDList Request	52
4.6	UpdateSenderIDList Response	52
4.7	Method: ReportFP Request	53
4.8	Response to ReportFP Request	53
4.9	Method: ReportFN Request	54
4.10	Response to ReportFN Request	54
4.11	Method: GetServices Request	55
4.12	Response to GetServices Request	55
4.13	Method: GetStatus Request	55
4.14	Response to GetStatus Request	56
4.15	Error Handling	56
4.15.1	Sample HTTP Error	56
4.16	SenderID	57
4.17	SenderID Definition and Threshold Crossing Notifications	57
5	THE SPAMASSASSIN NETWORK PROTOCOL	59
5.1	Request and Response Conventions	59
5.2	spamc Commands	60
5.3	SpamAssassin Network Protocol Headers Description	60
5.4	Method: spamc command	62
5.5	Response to spamc Command	64
5.6	Outbound Spam Detection Exim Response Sample	64
5.7	Exim for the SpamAssassin Network Protocol	65
6	SNMP COUNTERS	66
6.1	General SNMP Counters	66
6.2	Spam Classification Counters	67
6.3	HTTP Protocol Counters	68
6.4	Outbound Spam SNMP Counters	69
6.5	Outbound SenderID Threshold SNMP Counters	69
6.6	Spamd Protocol Counters	70
6.7	Virus Outbreak Detection SNMP Counters	70
6.8	Command Antivirus SNMP Counters	71
7	DEPLOYING CTASD OVER WAN	72
7.1	Implementing a Failover Mechanism	72
8	CTASD TESTING AND VERIFICATION	74
8.1	Connectivity Test	74
8.2	Acceptance Test	75
8.3	Corpus of Messages for Evaluation	75
8.3.1	For Spam classification	76
8.3.2	For Virus Outbreak Detection	76
8.3.3	For Command Anti-Virus	77
8.3.4	For SenderID Notifications	77
8.4	Running the Sample Client	77

8.5	Sample Response from ctasd.....	79
9	INDEX.....	81

1 Introduction

Every massive email-borne outbreak, spam or malware, can be identified by one or more recurrent patterns, even when almost every message within the attack is intentionally composed differently in an attempt to evade lexical analysis-based filters of the message-content or as an attempt to detect predefined virus signatures. CYREN RPD™ technology detects these patterns as it analyzes data culled from characteristics of Internet email traffic. By identifying these message patterns, RPD is able to identify and detect massive email-borne threat attacks with a high level of accuracy. CYREN RPD™ technology was designed to detect spam or malware that is written in every language and in all message-formats (text, HTML, images, etc.). RPD is hosted at the CYREN Datacenter, a global network of servers that provide unparalleled email, web, and antivirus security.

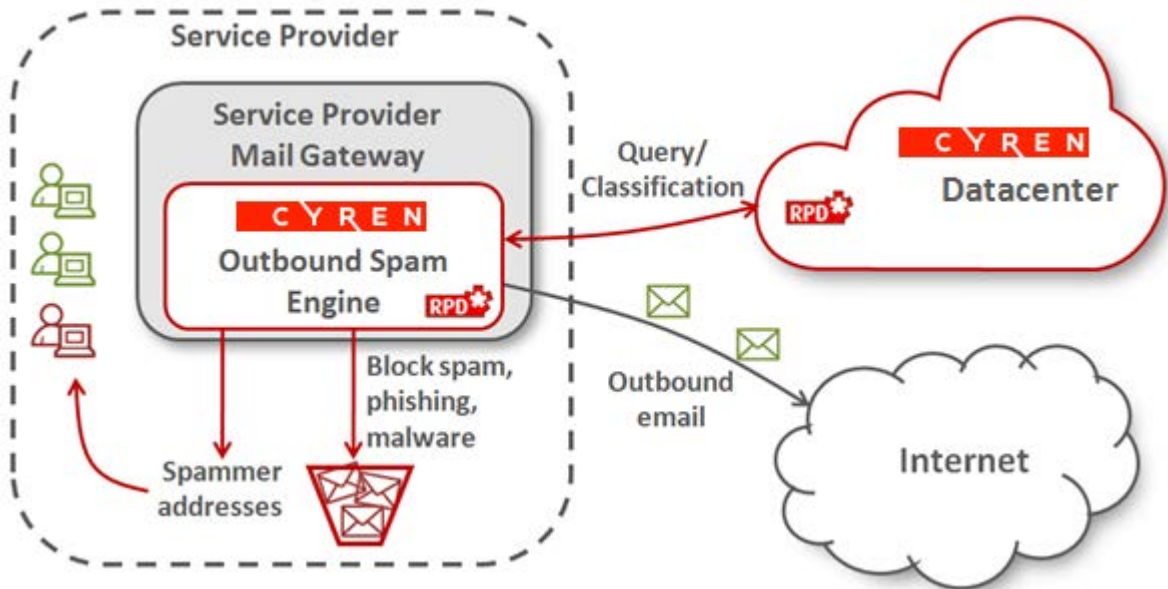
The CYREN Datacenter spans several separate sites for failover and load-balancing purposes. ctasd is designed with a built-in automatic failover mechanism in case the last-used Datacenter is not available.

Although highly unlikely, if the connectivity with all the Datacenters is interrupted for some reason, ctasd continues to retrieve new incoming messages and matches these new messages against a local classification cache, which is generally credited with detecting and classifying almost 70% of the spam messages. If a connection to all Datacenters is unavailable, messages that do not match the local cache are considered Uncategorized and classified accordingly and are not held until the connectivity is restored. In addition, a connectivity error message will be logged to the syslog file. When connectivity is resumed, the standard filter flow will resume.

CYREN Advanced Security Daemon (a.k.a. ctasd™) is a plug-n-play email-borne spam and malware outbreak detection daemon that combines your current core messaging network infrastructure with advanced detection and classification capabilities. The daemon adds a layer of email detection to your mail delivery system in order to provide real-time classification, already in the first minutes after a new outbreak is launched.

Service Providers or OEM partners involved in applications that involve mail delivery systems often unwittingly transport spam messages and therefore run the risk of being blacklisted on a daily basis, a situation which immediately affects subscribers' satisfaction. The ramification of one infected server or PC used in spam attacks can potentially stop every subscriber from sending email as they share the same public IP address range.

CYREN has specifically tailored a solution to properly address outbound spam, malware or phishing. CYREN's Outbound Spam Protection solution not only enables Service Providers and OEM partners to stop these unwanted emails but also provides immediate identification of the sender. The solution detects any type of outbound spam source, including zombie computers, compromised accounts, spammer accounts and webmail spammers.



ctasd supporting Outbound Spam protection not only detects spam but also identifies the spammer, giving Service Providers and OEM partners the power and tools to find and stop spammers from using their networks as a means of spamming others.

ctasd can be configured to support both the inbound Anti-Spam and the Outbound Spam Protection service. This manual will describe how to configure ctasd for the **Outbound Spam Protection Service**.

1.1 Working with ctasd

To work with ctasd, you need to have the following:

- The ctasd daemon
- Messages requiring detection
- A client application that connects to ctasd

ctasd is an HTTP server listening to HTTP requests on a predefined port. When it receives input about messages requiring classifications, it passes the data to an embedded Detection Engine (ctengine™) and then it is responsible for responding to the HTTP client with the specific classifications of the messages and notifications about the message sender.

The ctasd package includes a client application, named **http_client.pl**. This is a sample application that can be used for demonstration purposes and does not use the full functionality of ctasd. For a production environment, you should develop your own client.

The client application is responsible for passing information about the messages to ctasd. It will then receive classifications that can be translated afterwards to applied actions on the message senders.

The client connects to a predefined TCP port on the ctasd daemon and passes information about the messages that require classification in one of two ways:

- File reference
- Streaming data

In 'file reference' mode, the client passes the absolute path of a file to ctasd, which then loads the file. In 'streaming data' mode, the client passes the message headers and body to ctasd. In both cases, ctasd classifies the message and returns a series of relevant classifications to the client (Spam and/or malware), including notifications regarding the specific behavior of the sender.

The client sends HTTP requests to ctasd and receives HTTP responses. Communication is based on the CYREN-proprietary API using HTTP 1.0 conventions, as described later in this document.

The following steps demonstrate how to prepare and work with ctasd to detect outbound spam and spammers during evaluation and testing:

1. Modify the default configuration file (ctasd.conf) or create a new version of the configuration file.
2. Run the daemon, specifying the path of the configuration file to use.
3. Execute the http_client.pl sample code to connect and point to messages requiring filtering.

For more details how to make effective evaluation and testing see the section [ctasd Testing and Verification](#).

The daemon extracts some message characteristics, otherwise referred to as 'message-patterns', and then prepares and sends out a small query of few hundred bytes to the CYREN Datacenter. The Datacenter is responsible for warehousing a vast, constantly-expanding Spam and virus pattern repository. The Datacenter responds with classification to message patterns found in the original query. Full round-trip time for query/response is less than 300 ms, excluding Internet latency.

The global Spam and virus pattern repository on the CYREN Datacenter is a result of the fully-automated analysis center adjacent to the Datacenter. The Datacenter receives real-time information about new spam and virus attacks that emerge over the Internet, globally and regionally. It analyzes the information, updates the pattern repository with appropriate classification, and replies simultaneously to real-time outstanding queries from CYREN systems, such as ctasd, that are deployed at customer sites.

Every massive email-borne outbreak, Spam or malware, can be identified by one or more recurrent patterns, even when almost every message within the attack is intentionally composed differently in an attempt to evade lexical analysis-based filters of the message-content or as an attempt to detect predefined virus signatures. CYREN RPD technology was designed to detect spam or

malware that is written in every language and in all message-formats (text, HTML, images, etc.) by identifying these recurrent patterns as it analyzes data culled from characteristics of local and remote Internet email traffic. By identifying these patterns, the technology is able to identify and detect massive email-borne threat attacks with a high level of accuracy.

The Outbound Spam Protection service uses the analysis of regional traffic only, as it analyzes and detects suspected local patterns. This is in addition to having access to all globally detected patterns.

One of the key benefits of the Outbound Spam Protection service is that it not only detects spam messages but also identifies the spammers. To achieve this, ctasd manages counters on senders, notifying customers when specific sender threshold values have been crossed.

The CYREN Datacenter spans several separate sites for failover and load-balancing purposes. ctasd is designed with a built-in automatic failover mechanism in case the last-used Datacenter is not available.

Although highly unlikely, if the connectivity with all the Datacenters is interrupted for some reason, ctasd continues to retrieve new outbound messages and match these new messages against a local classification cache, which is generally credited with detecting and classifying almost 70% of the spam messages. If a connection to all Datacenters is unavailable, messages that do not match the local cache are considered Uncategorized and classified accordingly and are not held until the connectivity is restored. In addition, a connectivity error message will be logged to the syslog file. When connectivity is resumed, the standard filter flow will resume.

1.2 ctasd Solution Components

The overall ctasd solution involves the following components:

- CYREN Datacenter
- Querying devices or Mail Server
- ctasd daemon
- ctasd protocol
- ctasd for SpamAssassin Network protocol (if enabled)

CYREN Datacenter

The CYREN Datacenter consists of a series of global data servers located around the world that monitor global email traffic in real-time (24*7*365) from various sources on an ongoing basis. Collectively referred to as the CYREN Datacenter, these advanced data centers maintain a vast database of message patterns and classifications that are determined based on numerous dynamically changing parameters.

Querying Device or Mail Server

The term “querying device” is used as a generic term for OEM applications responsible for email filtering. These applications are integrated with ctasd to provide anti-spam and virus outbreak detection by sending queries to ctasd over HTTP. Once a response is received from ctasd, the querying device is responsible for applying the appropriate policy/action on the filtered messages.

In a configuration using a SpamAssassin-compatible protocol, ctasd communicates with the “Mail Server,” a generic term used to describe an OEM or Service Provider’s MTA, which is responsible for email filtering. These mail servers are integrated with ctasd to provide anti-spam and virus outbreak detection by sending queries to ctasd using their current SpamAssassin-compatible methodology. Once a response is received from ctasd, the Mail Server is responsible for applying the appropriate policy/action on the filtered messages.

ctasd

The CYREN Advanced Security daemon (ctasd) performs various functions to provide the most advanced anti-spam, anti-virus protection available. It is responsible for receiving and processing incoming queries from query devices to determining the spam and/or virus classifications of outbound messages. In addition, ctasd notifies querying devices of detected spammers, zombies or compromised accounts.

ctasd Protocol

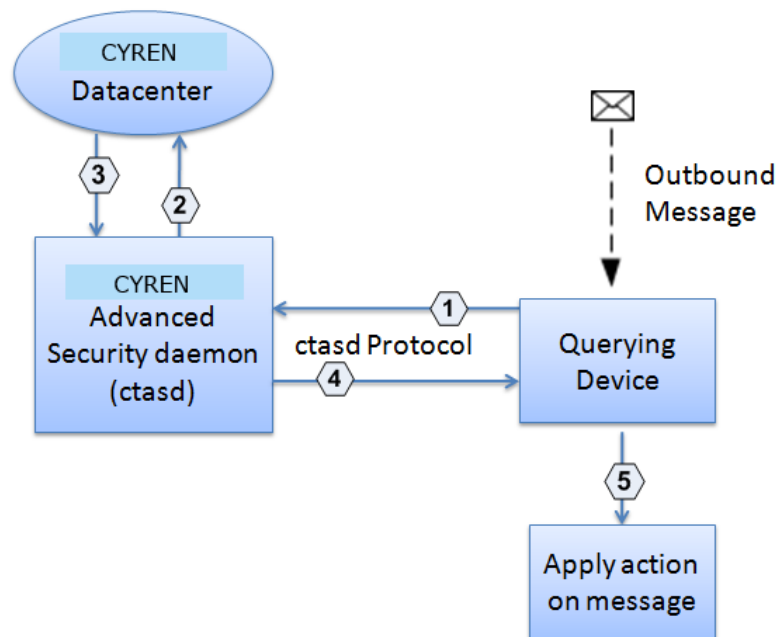
In order to enable communication between a querying device and ctasd, CYREN has developed a simple protocol for Service Providers and OEM partners using ctasd. This enables Service Providers and OEM partners to provide advanced anti-spam and virus outbreak detection services to their users. Communication between the ctasd and the querying device is accomplished over HTTP. For more information, see [ctasd Protocol](#).

ctasd incompatibility

The cache file has an incompatibility between 32-64 bits, and you cannot reuse the cache file.

1.3 System Architecture and Data Flow

Although the data flow of ctasd can vary depending on configuration settings and deployment scenarios, a typical data flow is detailed below:



1. An outbound message is received by the querying device. The querying device uses the CYREN ctasd protocol to generate a query to ctasd requesting spam and virus classifications. The SenderID is included in the query to detect spamming SenderID activities.
2. ctasd prepares and forwards a query to a CYREN Datacenter to retrieve the most up-to-date classification for each of the services.
3. The CYREN Datacenter responds to ctasd with current information regarding the message patterns in the query.
4. ctasd then prepares a response, collating all current information into a pre-determined format and sends the response back to the querying device. The information includes notifications regarding spamming behavior of the SenderID. The ctasd counters are also updated to help track spamming activities of each SenderID.
5. The querying device, upon receiving the response from ctasd, may apply a predefined action to the sender (i.e., place behind a walled garden, send warning mail etc).

ctasd automatically stores message pattern classifications received in responses from the CYREN Datacenter to a local cache to optimize the spam and malware detection and classification process. ctasd analyzes message patterns against this local cache as the first step in detection and filtering.

If a match is found based on previous queries, a similar classification is assigned and the querying device can then apply an appropriate action without sending a new query to a CYREN Datacenter.

If no match is found, ctasd will then prepare and send a query to a CYREN Datacenter. The local cache is updated regularly each time a response is received from the Datacenter and older or expired classifications are deleted.

In addition to the local cache, ctasd maintains a persistent cache, which is automatically reloaded when ctasd is stopped and restarted. This helps improve overall response time because it restores the local cache with the most up-to-date classifications, as well as previously stored classifications that are still relevant.

Depending on the service provisioned and in progress, ctasd can operate either in synchronous mode or in asynchronous mode.

Note: When integrating both anti-spam email service and CYREN's Command Antivirus protection, you should operate ctasd in synchronous communication. When operating in asynchronous communication, ctasd will only return spam classifications.

1.3.1 White List and Blue List

For Outbound Spam Protection, OEM partners can maintain a White list and a Blue list.

- **A white list:** listing of senders who have been cleared for any activity in the system including sending out suspected and spam message. No notifications will be triggered for white-listed senders. Those on this list are given complete freedom to send messages regardless of any defined thresholds.
- **A blue list:** listing of senders who have been cleared only for sending out suspected messages but not spam messages. Notifications will be triggered for blue-listed senders if they attempt to send out spam messages but any messages that are not confirmed spam.

The functions of these lists differs slightly when used for the Outbound services as compared to Inbound services. In Outbound, the White and Blue lists refer to the SenderIDs and indicate whether a specific SenderID can send messages based on a specified classification. Those on the White list can send any messages, even those considered spam or suspected while those on the blue list can only send suspected messages but not confirmed spam.

In addition, the LocalView Custom Rules engine for Inbound services also allows for the use of whitelists and blacklists. These lists refer to lists that contain any IP or From address. All messages that arrive from an entry in a white lists is given a NonSpam classification and any message received from an entry on the blacklist received a Confirmed Spam classification.

1.4 Spam Classifications

The following table describes all the optional Spam classifications that ctasd can return in response to a `ClassifyMessage_File/Inline` request from the querying devices. (X-CTCH-Spam)

Additionally, this table outlines some guidelines for how to handle messages of each type.

Classification	Explanation
Confirmed	Spam messages from known spam sources (e.g. zombies).
Bulk	Spam messages from sources that are not confirmed spammers.
Suspect	Legitimate messages that are sent to slightly larger than average distribution or are unidentified spam messages in the first few seconds of a massive spam outbreak.
Unknown	Messages for which ctasd does not have any incriminating information, and are therefore assumed to represent legitimate correspondence.
NonSpam	Messages that are confirmed, without doubt, as coming from trusted sources. This classification is very rarely used.
Valid-Bulk	Messages that are determined by the Datacenter to be valid bulk (e.g. solicited bulk messages such as newsletters).

1.5 Virus Outbreak Detection

Because CYREN's Virus Outbreak Detection services (VOD™) are designed to detect new virus outbreaks as they are being launched, it is highly recommended that you enable CYREN's Command Antivirus protection (or another anti-virus application). By scanning messages or objects with the Command Antivirus service, known viruses will be quickly identified and blocked. This improves overall performance because it lowers the number of messages that need to be scanned and classified by ctasd's other email security services.

All messages detected with a virus are included in the Sender Spam counter. Threat Level Classifications (X-CTCH-VOD)

Classification	Explanation
Virus	The message contains characteristics of confirmed malware.
High	High likelihood of the message presenting a malware threat.
Medium	Probable threat of malware in the message has

Classification	Explanation
	been detected.
Unknown	Threat for malware could not be determined at this time.
NonVirus	Confirmed that message does not contain a malware.

1.6 Command Antivirus Protection

When Command Antivirus functionality is enabled and integrated, each query sent by the querying device to the CYREN Datacenter (by ctasd) will result in an additional classification in the response. For instances in which a virus is detected, the response will include the virus type, accuracy, virus name and scan result.

An important element of the antivirus solution is the creation and maintenance of definition files that contain information about known viruses. ctasd downloads updated definition files on a predetermined schedule. Initially, the unified daemon tries to update the most updated def files from the Datacenter. If it fails to download the most recent updates, it will continue to try to download in the background.

The following table describes the possible Command Anti-Virus Protection classifications ctasd can return for a query:

Classification	Explanation
vseScanResultNone	Nothing was found.
vseScanResultFound	<p>An object of concern was detected. In this case, the following information will also be returned:</p> <ul style="list-style-type: none"> Detection Type (see Command Antivirus Detection Types) Detection Accuracy (see Detection Accuracy Values) Virus Name

1.6.1 Command Antivirus Detection Types

The following table details possible detection type values and an explanation for each.

Header	Explanation
vseDetectionTypeNone	Nothing was detected.
vseDetectionTypeVirus	A virus detected.
vseDetectionTypeAdware	Adware was detected. There may be legal implications in reporting this. Please refer to Legal Issues Surrounding Detection
vseDetectionTypeApplication	Application was detected. There may be legal implications in reporting this.
vseDetectionTypeBackdoor	Backdoor Trojan was detected. A backdoor is a type of malware that allows an infected machine to be remotely controlled.
vseDetectionTypeBomb	Archive Bomb or something similar detected. An archive bomb is an archive that when scanned or extracted will exhaust the resources of the machine doing those actions.
vseDetectionTypeBootVirus	Boot Sector Virus detected.
vseDetectionTypeDenial	Tool used for Denial of Service attacks detected.
vseDetectionTypeDialer	Dialer was detected.
vseDetectionTypeDownloader	Downloader for other types of malware detected. A downloader is a piece of malware that will download other malware.
vseDetectionTypeExploit	Exploit detected. An exploit for some operating system or common application was detected. The exploit can be a vector to introduce other malware or to stop the application or operating system from working as expected.
vseDetectionTypeGarbage	Non-virus we have to detect. This is most likely something that this is not a virus/ malware but we have to test as malware testers use it in their collections.

Header	Explanation
vseDetectionTypeJoke	Joke application was detected.
vseDetectionTypeMacro	Macro has been detected. This will be reported with paranoid heuristics if any macros are detected in a document.
vseDetectionTypeMassMailer	Mass mailer detected. This is a worm that spreads using email as its primary distribution technique.
vseDetectionTypeMisDisinfection	Mis-disinfected virus detected. A mis-disinfected virus is one that another product tried to disinfect but could not do so.
vseDetectionTypeNetWorm	Network worm detected. This is a worm that spreads using network vulnerabilities or poor network security as its primary distribution technique.
vseDetectionTypeP2PWorm	Peer to peer networking worm detected. This is a worm that spreads using peer to peer file sharing networks.
vseDetectionTypeProxy	Backdoor Proxy detected. This is malware that will allow unauthorized connections to be made through this machine. This allows a malicious person to use the machine infected with this to attack other machines, send spam or impersonate your computer.
vseDetectionTypePasswordStealer	Password Stealer detected. This is malware that will attempt to steal passwords. It will do it by stealing files and registry keys that are normally used to store passwords, doing key logging or taking screen shots.
vseDetectionTypeRemote	Remote template detected. This is a document that contains a reference to a template that can potentially include exploits or viruses and is accessed over the Internet.
vseDetectionTypeRisk	Security risk detected. This is a risk that we could not accurately categorize into one of the other risks due to either its unique abilities or due to it using a wide

Header	Explanation
	variety of techniques.
vseDetectionTypeSpyware	Spyware was detected.
vseDetectionTypeTool	Virus generation tool detected. This is a tool used to generate viruses or exploits.
vseDetectionTypeTrojan	Trojan detected. This is an application that will pretend to have a useful purpose and then do damage when the user does not expect it.
vseDetectionTypeHiddenProcess	Hidden process detected. This is a hidden process as a result of rootkit activity.
vseDetectionTypeInjectedCode	Injected code detected. This is (possibly) a legitimate process with malicious code injected in it.
vseDetectionTypePacker	A Packer was detected. Packers/Obfuscators are tools that are used on an executable to make it smaller or harder to reverse engineer. It should be considered a warning.

1.6.2 Detection Accuracy Values

The following table details possible accuracy type values and an explanation for each.

Header	Explanation
vseDetectionAccuracyNone	Not specified
vseDetectionAccuracyExact	Exact signature detection.
vseDetectionAccuracyHeuristic	Heuristics were used for this detection.
vseDetectionAccuracyNewOrModified	New or modified version of a known virus.
vseDetectionAccuracyLow	Low detection certainty. May be an exploit.

1.7 Synchronous vs. Asynchronous Communication

In Synchronous communication mode, a thread is allocated per each classification request. The query device initiates a query and then waits for the response. In this mode, higher memory resources are required. To ensure backwards compatibility, ctasd continues to work, by default, in Synchronous mode.

In Asynchronous communication mode, a single thread can handle multiple queries. The query device sends multiple queries, each query identified by its own unique context. Once a response is received, ctasd performs a callback to the query device with the specific query's context. The Asynchronous mode supports a much higher classify transaction rates. This saves additional resources because the query device does not have to maintain a thread per classify request. CYREN's Callback functionality runs a callback API function with the classification response. When enabled, the classification response will be transmitted back to the query device asynchronously.

When implementing ctasd with CYREN's Command Antivirus protection, a ClassifyMessage request will only return a Spam classification. Therefore, only Synchronous communication should be used when implementing ctasd's both spam detection and virus protection services. Alternatively, consider using the ClassifyObject function, which will return both spam and virus classifications.

1.8 SpamAssassin-Compatible Network Protocol Option

SpamAssassin is a leading open source email spam filtering solution. It is supported by a wide variety of email systems (e.g. Exim). Though SpamAssassin can be run as a standalone application, or as a subprogram of another application (e.g. Mail Scanner), it also can be run as a client (spamc) that communicates with a daemon (spamd). The client/server mode of operation offers major performance benefits. The SpamAssassin Network Protocol is the communication protocol between spamc and spamd.

CYREN has enabled its customers with two ways to integrate ctasd using the SpamAssassin protocol:

- ctasd SpamAssassin Network Protocol Daemon
- CYREN Plug-In for SpamAssassin

The ctasd daemon can use standard SpamAssassin Network Protocol to quickly and easily integrate with a Mail Transfer Agent (MTA) that supports this protocol. ctasd SpamAssassin Network Protocol Daemon offers the following benefits:

- Increasing security and performance.
- Saving valuable time and resources.
- Leveraging existing SpamAssassin infrastructure to improve competitive offering.
- Increasing productivity by relying on a full commercial service, including SLA and support.

- Blocking any Spam outbreak with 98% detection and near zero false positives.
- Blocking viruses within seconds of any new outbreak.

ctasd SpamAssassin Network Protocol integration features:

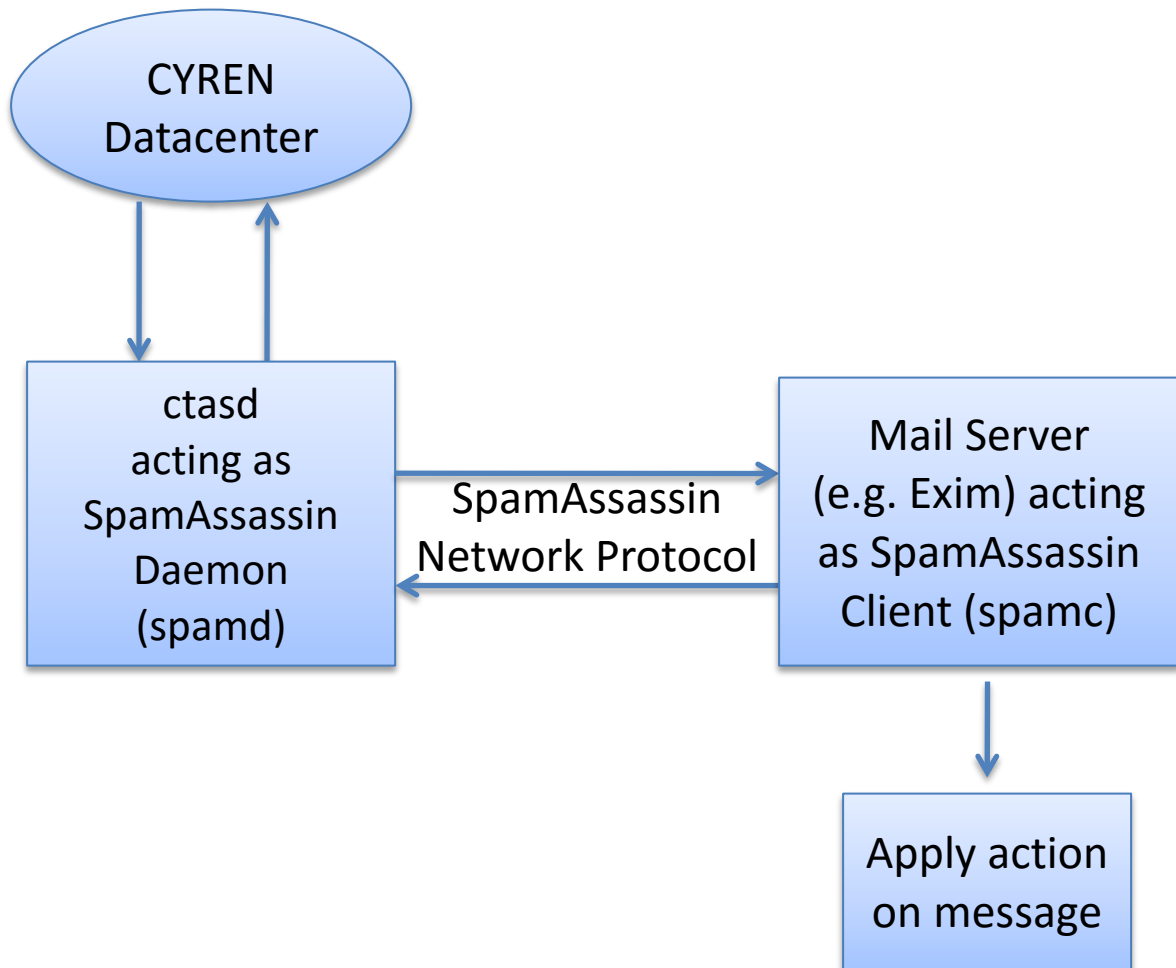
- Fully automated system with no administration or maintenance required.
- High accuracy in any format or language, including image and PDF spam.
- In-the-cloud architecture that scales to the needs of the most demanding environment.
- Fast and easy integration that allows you to gain immediate competitive edge with almost no effort.
- Increase throughput and performance by dramatically reduced message scanning time.
- Cross platform support works with virtually any SpamAssassin environment.

1.8.1.1 CYREN Plug-In for SpamAssassin

For those customers who are currently using SpamAssassin and wish to integrate CYREN's spam and virus protection into its existing infrastructure to improve detection accuracy, CYREN also has a Plug-In for SpamAssassin that queries and retrieves spam and malware classifications from the CYREN Datacenter in real-time. For more information, contact your local CYREN Sales Representative.

1.9 SpamAssassin-Compatible Workflow

A typical ctasd for SpamAssassin Network Protocol daemon data flow is detailed below:



- An outbound message is received by the Mail Server. The Mail Server, acting as a SpamAssassin client (spamc) uses the SpamAssassin Network protocol to generate a query to ctasd requesting spam and VOD classifications.
- ctasd, acting as a SpamAssassin daemon (spamd) receives the query, checks its local cache for classification results and, where necessary, prepares and forwards a query to the CYREN Datacenter to retrieve the most up-to-date classification.
- The CYREN Datacenter responds to ctasd with current information regarding the message patterns in the query.

- ctasd then prepares a response using the SpamAssassin Network Protocol with all current information into a pre-determined format and sends the response back to the Mail Server.
- The Mail Server, upon receiving the response from ctasd, may apply a predefined action to the message (i.e., reject the message, approve the message and pass it to the recipient, etc.).

As explained in the previous section, ctasd message patterns are saved in a local cache, and uses a persistent cache to improve response time.

1.10 ctasd Deployment Options

ctasd can integrate with a wide variety of applications and devices to enable outbound spam detection services. Each deployment option is adaptable to the individual requirements and infrastructure of the CYREN customer and its end users or specific Service Provider/OEM partner architecture.

Following is a partial list of some of the ways in which ctasd can be deployed:

- A single ctasd standalone daemon (running on either a dedicated box or one of the existing machines within the organization) can be used to serve one or more querying devices simultaneously.
- Multiple ctasd daemons running on multiple machines can serve one or more querying devices.
- One or more ctasd daemons can run on the same machine.
- ctasd may also run as a fully-embedded daemon that is tightly integrated with the Service Provider or OEM partner solution.

ctasd can be deployed on the customer's premises, thus requiring no authentication between ctasd and the querying devices. Nonetheless, ctasd will require authentication of communication between ctasd and the CYREN Datacenter. Alternatively, ctasd can be deployed over WAN and receive queries remotely from the querying devices. To learn more about this deployment option, review [Deploying ctasd Over WAN](#).

1.11 ctasd Minimum Requirements

Following is a list of recommended hardware requirements:

- Single CPU, 2.8 GHz
- 1 GB RAM
- 100 MB free disk space
- 100 Mbps Network interface

1.12 ctasd Package

The ctasd package contains the following:

- ctasd daemon and associated binary
- ctasd documentation
- Sample files for quick evaluation and testing
- SNMP script for counters

1.13 Internal Directory Structure

Depending on which platform your application uses, you may need to download and use different package of ctasd.

Following is a list of the internal directories and files:

<code>./ctasd-<version>-<platform>-<compiler>/</code>	
<code>ReleaseNotes.txt</code>	
	<code>/bin</code> Binary files, the configuration file, Command SDK engine + SDK files, and definition files
	<code>/bin/oslib</code> Operating system libraries
<code>/bin/snmp</code>	SNMP files, including sample scripts for evaluating and testing
<code>/bin/snmp/mibs</code>	SNMP MIB files
	<code>/corpus</code> Files used for testing ctasd integration
	<code>/docs</code> ctasd documentation
	<code>/samples</code> Sample scripts for evaluation and testing

1.13.1 RPM Structure

Following is the structure when installing ctasd via the RPM file:

<code>/usr/lib/ctasd</code>	Stores all the binaries in the same structure as in the bin directory of the ctasd package
<code>/usr/lib/ctasd/snmp</code>	Stores all SNMP counters scripts
<code>/etc/ctasd/</code>	Stores configuration file (ctasd.conf)

`/etc/init.d/`

Contains the `ctasd_initd`. To assist you in identifying the folder, you may want to rename it to `ctasd`.

2 ctasd Configuration

A default `ctasd.conf` file is provided with the `ctasd` package containing the necessary configuration parameters for the daemon. The administrator can configure a single `ctasd.conf` file and then copy it to other locations to be used by other daemons, but each instance of `ctasd` must have its own, unique `ctasd.conf` file.

By default, the `ctasd.conf` file is located in the default `/etc/ctasd` directory. However, this can be changed using the `-c` switch.

Most of the parameters in the configuration are optional, and have default values that are enforced if another value is not specified. However, in order for connectivity with the Datacenter to be established, valid `License_key_code` and the `Server_address` must be manually set.

Note: *If changes are made to the `ctasd.conf` file while the daemon is running, `ctasd` must be stopped and restarted for the changes to take effect.*

2.1 Sample Configuration File

Following is a copy of the default configuration file. The next section provides a detailed description of the parameters.

```
#                                     ctasd Configuration File
#
#-----
#
#
#           Note:      If you change this file, you must restart
#                       CYREN ctasd in order to have your
#                       changes take effect.
#
#-----
#-----

[General]
```



```
PersistentCacheEnabled=1
UseAuthMode=0

#ValidBulkEnabled- This parameter defines whether the Valid Bulk
#classification should be enabled. By default Valid Bulk is
disabled.
#ValidBulkEnabled=0

#SpamdServerEnabled - defines the spamd as the protocol by
which the
#Mail Server will be communication with ctasd
#SpamdServerEnabled=1

# Outbound Spam - configures the ctasd daemon to detect
outbound
# spam as part of the CYREN Outbound Spam Protection
Service.
# The default value is disabled.
#OutboundEnabled=0

#Defines whether communication with the CYREN Datacenter is
Asynchronous or Synchronous.
#For backward compatibility, the default is disabled.
#For improved performance, it is recommended that you
enable this parameter.
#AsyncResolverRequests=0

#IP Ignore List for IP addresses of all local mail servers.
#IP_ignore_list =

# Connectivity Section

# * Your license key code (mandatory)
# * Server address (mandatory)
# * Maximum cache records value (optional)
# * Proxy Server settings (only if using proxy server)
```

```
[Connectivity]
License_key_code = xxxxxxxxxxxxxxxxxxxxxxxx
Server_address = xxxxxxxxxxxxxxxxxxxxxxxx

# This is the maximum number of records that will be
# stored in the local spam detection cache.

Cache_max_records = 100000

# If you connect to the Internet through a proxy server, you
# should uncomment the following parameters and assign
# appropriate
# values.

#ProxyPort =
#ProxyServerAddress =
#ProxyAuth =
#ProxyUserName =
#ProxyPassword =
#ProxyAccess =

# Security Section
#
# Associates the user and group names for the daemon.
#

[Security]
User=root
Group=root

# HttpServer Section
#
# Specify the TCP port on the daemon to which the client connects,
# and the relevant connectivity and performance parameters.
#

[HttpServer]

Port=8088

ListenBackLog=100
```

```
InitialThreads=1

MaxThreads=50

Concurrency=50

#TempFolder= OS default temp folder

#If BindingAddress is empty (or commented out). BindingAddress is
set to INADDR_ANY.

#BindingAddress=<IP Address>

# Spamd Section

#

#This section is applicable only if the SpamdEnabled flag is
enabled.

#Specify the spamd port on the daemon to which the client
connects, and the relevant connectivity and performance
paramters.

#You may change the spamd default scores - however please refer
to the ctasd Integration Guide for guidelines.

#

[Spamd]
#ConfirmedScore = 100
#BulkScore = 50
#SuspectedScore = 2
#NonSpamScore = -100
#VirusScore = 200
#HighScore = 150
#MediumScore =4
#SpamThreshold=50
#Port = 7830
#ReceiveTimeout=5000
#ListenBacklog=100
#InitialThreads=5
#MaxThreads=100
#Concurrency=30
```

```
#TempFolder= OS default temp folder
#BindingAddress =

[AV]

# AVDefPath defines the local path where the virus Definitions
File is saved and
# the path from which it is read.
# The default value is the current directory.
#AVDefPath

# AVScanMode defines the scan mode. The mode setting has a
significant performance
# impact.
# The higher the scan mode setting, the more resources are
required to perform the
# scan.
# Default value: medium.
#AVScanMode=medium

# AVWaitForUpdatedDefFiles defines if to wait for updated
definition files or not.
# By default, ctasd will not wait for updated definition files.
# Default value: 0 (disabled).
#AVWaitForUpdatedDefFiles=0

[Stats]
#Port=/tmp/ctasd.stats
# If BindingAddress is empty (or commented), BindingAddress is
set to INADDR_ANY.
#BindingAddress=<IP Address>

# Outbound Spam section
# * Define SenderID counter time durations.
# * Define which SenderID counters to enable
# * Define SenderID counter threshold values
# A threshold with no value indicates that this threshold
is not in use
# * Define max cache values for Outbound Spam counters
```

```
# * Define policy of how to determine the SenderID of a
message
# * Define white and grey list definitions
# * Define reporting time intervals for alerting of crossed
thresholds
# * Define if to report counter values in reply messages

[Outbound]

#CountersMask is a bit-wise flag which defines which SenderID
counters to enable
#By default the SuspectedCounter, TotalMessagesCounter and
BulkCounter are enabled
#The following displays the flag values per each SenderID
counter
#           SuspectedCounter = 1
#           TotalMessagesCounter = 2
#           SpamCounter = 4
#           BulkCounter = 8
#           ConfirmedCounter = 16
#           RecipientsCounter = 32
#           VirusCounter = 64
#CountersMask=7

#The size/time duration of each time window managed for each
SenderID Counter.
#The parameter is measured in seconds.
#SenderIDWindowSize=60

#The number of time windows to be managed for each senderID
counter.
#SenderIDWindows=5

#Maximum cache values for outbound spam SenderIDs
#CacheMaxEntries=1000000

#Policy definition of how to determine the SenderID of a message
#A customer may decide to either explicitly pass the SenderID in
each message
# Or alternately define here how to extract the SenderID
```

```
#SenderIdHeaderName: Defines the name of the message header from
which to extract the SenderID
#   Example message headers are: From, Reply-To headers
SenderIdHeaderName=From

#   SenderIDHeaderFormat specifies if the value of the header
will be taken "as is"
#   Or the email address only will be extracted from the header.
#   Place the value "raw" when taking the value as-is; Place the
value "email" when
#   taking the email only.
SenderIdHeaderFormat=email

#   SenderIdIgnoreList is applicable only if the "Received"
header was defined in
#   SenderIDHeaderName
#   The SenderID is extracted from the last Received header.
#   The system will ignore all SenderIds appearing in the
SenderIdIgnoreList
SenderIdIgnoreList=

# Suspected counter thresholds per each SenderID
#SuspectedThreshold1=

# Spam counter thresholds per each SenderID
#SpamThreshold1=

# Bulk counter thresholds per each SenderID
#BulkThreshold1=

# Confirmed counter thresholds per each SenderID
#ConfirmedThreshold1=

# Virus counter thresholds per each SenderID
#VirusThreshold1=

# Total message counter thresholds per each SenderID
#TotalThreshold1=

# Recipients counter thresholds per each SenderID
#RecipientsThreshold1=
```

```
#White and Blue listing file name settings.
#   If no path is defined, file will be created in ctasd local
directory.
#SenderIDWhiteListFile=senderid_white
#SenderIDBlueListFile=senderid_blue

# The reporting time interval in seconds for alerting repeating
crossed thresholds
#SenderIDReportingInterval=600

# Flag if the ctasd ClassifyMessage response message should
include also counter
# values.
#ReportCounters=0

[LocalView]

# The location that the custom rules are stored.
# This is a mandatory field if LocalView is enabled.
#CustomRulesFilePath=

#The URL of a local static content server used for Local Custom
Rules distribution.
#This parameter should be defined only if a local distribution
server is defined.
#The Local Custom Rules will be copied and placed in
CustomRulesFilePath location.
#LocalCustomRulesDistributionURL

#The LocalView Bulk threshold.
#LocalView_BulkThreshold=5

#The LocalView Confirmed threshold.
#LocalView_ConfirmedThreshold=10

#Flag to enable/disable shortcircuit functionality.
#ShortCircuitEnabled=0

#The short circuit threshold; if crossed, message scanning will
be short-circuited.
```

```
#ShortCircuitThreshold=100

#Defines CT IPRep RBL rule scores
#CTIPRepRBL_Tags= "IP-Black=5 IP-Dark-Grey=4 IP-Grey=3.5 IP-
White=-0.3 IP-Very-White=-0.6"

#The entries are matched against these headers.
#WBLHeaderListFrom=Envelope-Sender,Resent-Sender,X-Envelope-
From,From,list-unsubscribe,Sender,Mail-From
```

2.1.1 [General]

PersistentCacheEnabled

The PersistentCacheEnabled option defines whether ctasd will operate with a persistent cache file. When enabled, the persistent cache file is automatically reloaded when ctasd is restarted. This helps improve overall response time because it restores the local cache with the most up-to-date classifications, as well as previously stored classifications that are still relevant. Default value: 1 (enabled)

UseAuthMode

The UseAuthMode option defines whether ctasd is deployed over LAN (0) or over WAN (1). For more information, review the relevant section: [Deploying ctasd Over WAN](#). If ctasd is deployed over WAN you need to ensure that HTTP requests are sent with the X-CTCH-Key header as explained later in the section [ctasd protocol](#). Default value: 0 (disabled).

ValidBulkEnabled

The ValidBulkEnabled option defines whether ctasd will return classifications of Valid Bulk. Default value = 0.

Note: The Valid-Bulk feature is not relevant for Outbound spam deployment.

AsyncResolverRequests

Use the AsyncResolverRequests value to enable or disable the Asynchronous Mode for ctasd. By default, the option is disabled and ctasd operates in Synchronous mode, which is backwards compatible with previous versions of ctasd. To work in Asynchronous mode, change the value to 1. However, if you are integrating ctasd to provide both email anti-spam services and Command Antivirus protection, it is recommended that you run ctasd in synchronous mode only. Default value: 0

SpamServerEnabled

The SpamServerEnabled option defines whether the spamd protocol will be used. By default, the option is enabled, allowing the mail server to communicate with ctasd using a SpamAssassin-compatible networking protocol. Default value: 1 (enabled).

OutboundEnabled

The OutboundEnabled option defines whether to allocate resources in ctasd for Outbound Spam Protection service specific features. This parameter must be enabled if ctasd is used to run the Outbound Spam Protection service. Default value: 0 (disabled).

Note: The OutboundEnabled option must be changed from its default value to 1 (enabled) and a unique license key for this service should be used.

2.1.2 [Connectivity]

License_key_code = <license key code>

Enter the license key code for ctasd. If an incorrect number is entered, the CYREN Datacenter will be unable to authenticate the organization and therefore decline detection services. This is a mandatory parameter in the Connection String, and consists of the following:

- **CYREN token:** 20-character unique identifier provided by CYREN to identify the Service Provider/OEM partner.
- **Service Provider token:** A unique identifier (up to 35 alphanumeric characters) provided by the Service Provider/OEM partner.

The Service Provider or OEM partner identifier should distinguish between each user, device, or installation. In cases where more than one CYREN product or service is installed on the same device, a unique token should be created per instance. The token should be unique for the lifetime of the host application and should not be changed so that the same Service Provider/OEM partner token is used each time the application is initiated. It can be based on hardware or software-specific data. CYREN needs this full license key format to offer the highest level of customer support and service.

The format for this concatenated parameter uses a colon delimiter, as follows:

LicenseKey=<CYREN token>:<unique Service Provider/OEM partner token>

Example: LicenseKey=0001K032B1010W167E2B:12345-1234A-55555

Server_address = <DNS string>

The DNS string is the server address at CYREN Datacenter. Contact your CYREN sales representative to obtain a valid DNS string that will uniquely identify your queries.

Cache_max_records = <value>

The maximum size of the local spam classification cache. There is no specific upper limit to this value. Once the cache reaches this amount, the oldest records are overwritten with newer ones, thus limiting the cache records to the specified value set in the configuration file. Default value: 100,000.

2.1.3 [Security]**User = <user name>**

User name associated with ctasd.

Group = <group name>

Group associated with ctasd.

2.1.4 [HttpServer]**Port = <number>**

Listening port number. Default value: 8088.

ListenBacklog = <number>

Maximum number of outstanding connection requests in listen()'s input queue associated with SOCK_STREAM or SOCK_SEQPACKET type sockets.
Default value: 100.

InitialThreads = <number>

Initial amount of threads waiting for client requests when starting ctasd.
Default value: 1.

MaxThreads = <number>

Maximum amount of threads waiting for client requests during ongoing operation.
Default value: 50

Note: When using the daemon with proxy (i.e. `ProxyAccess=1`), it is recommended to set the `MaxThreads` for the HTTP server to **35 or below** in order to avoid ephemeral port starvation.

Concurrency = <number>

Maximum concurrent sessions handled by the ctasd daemon.
Default value: 50

BindingAddress=<IP Address>

The BindingAddress option defines the IP address to monitor for traffic when ctasd is deployed on a machine with multiple network cards. If a specific IP address is not specified, then ctasd will monitor traffic on all available IP addresses.

Default value: INADDR_ANY

TempFolder= OS default temp folder

Directory to be used for short term operations. The directory should have read/write permissions.

Default value: OS default temp folder

2.1.5 [Spamd]

SpamAssassin is a score-based engine, therefore some of the SpamAssassin Network Protocol Command results include spam scores. In this section, it is defined how the CYREN spam classifications are translated to spam scores. The Spam Threshold defines the threshold that needs to be crossed so that the Spamd response will define the message as Spam. The default configuration defines that both Confirmed and Bulk spam classifications, will result in a Spam classification by the spamd protocol.

Note: *It is not recommended that you change the Scores parameters.*

ConfirmedScore

The ConfirmedScore value indicates the current value used to define a message as Confirmed spam. Default value: 100.

BulkScore

The BulkScore value indicates the current value used to define a message as Bulk. Default value: 50.

SuspectedScore

The SuspectedScore value indicates the current value used to define a message as Suspected spam. Default value: 2.

NonSpamScore

The NonSpamScore value indicates the current value used to define a message as NonSpam spam. Default value: -100.

VirusScore

The VirusScore value indicates the current value used to define a message as a virus. Default value: 200.

HighScore

The HighScore value indicates the current value used to define a message as having a high likelihood of containing a virus. Default value: 150.

MediumScore

The MediumScore value indicates the current value used to define a message as having a medium likelihood of containing a virus. Default value: 4.

SpamThreshold

By default, the SpamThreshold value is set at 50, as is the BulkScore (default value of 50). If you wish to differentiate between Confirmed Spam and Bulk, you can change the default value of the SpamThreshold to a number that is higher than the bulk default setting (50), but lower than the Confirmed Spam default value (100), meaning anything from 51-99. In this case, Bulk messages will not be classified as spam, while Confirmed Spam will still be flagged as spam. Default value: 50.

Port

The Port value defines the Listening port number. Default value: 7830.

ReceiveTimeout

The ReceiveTimeout value indicates the interval of time that a connection can remain inactive, during which no application messages are received, before it is dropped. It is measured in milliseconds. Default value: 5,000 ms.

ListenBacklog

Maximum number of outstanding connection requests in listen()'s input queue associated with SOCK_STREAM or SOCK_SEQPACKET type sockets.
Default value: 100.

InitialThreads

Initial amount of threads waiting for client requests when starting ctasd.
Default value: 5.

MaxThreads

Maximum amount of threads waiting for client requests during ongoing operation.
Default value: 100

Concurrency

Maximum concurrent sessions handled by the ctasd daemon.
Default value: 30

TempFolder= OS default temp folder

Directory to be used for short term operations. The directory should have read/write permissions.

Default value: OS default temp folder

BindingAddress

The BindingAddress option defines the IP address to monitor for traffic when ctasd is deployed on a machine with multiple network cards. If a specific IP address is not specified, then ctasd will monitor traffic on all available IP addresses.

2.1.6 [AV]**AVDePath**

Defines the local path where the virus Definitions File is saved and the path from which it is read. The AVDefPath parameter defines the local path for virus definition file storage and reading. CYREN recommends placing these files in a /var/<prod_name> directory, where the host application has Read/Write permissions. To enable the Anti-Virus engine to run, Definition files must be located in the defined AVDefPath. Therefore, it is required that you copy the packaged Definition files (located in the root of the package) to the designated location prior to starting up the engine.

It is important that these files not be placed in a directory with a defined clean-up mechanism. The following files should be copied to the directory:

- aivsecon.def
- antiviri.def

Default value: the current directory

AVScanMode

Defines the scan mode. Possible values: Low, Medium, High, VeryHigh. This mode has a significant performance impact and may be useful when scanning virus collections or other atypical files.

Default value: medium.

AVWaitForUpdatedDefFiles

When ctasd is started, by default it does not wait for the updated Definition files before scanning any messages or files. Alternatively, if this is enabled, ctasd will wait while the latest Definition files are downloaded.

Default value: 0.

2.1.7 [Stats]

StatusPort

The Port option defines the server port for the statistics information collected from the SNMP counters. Default value: /var/run/ctasd/ctasd.stats

BindingAddress=<IP Address>

The BindingAddress option defines the IP address to monitor for traffic when ctasd is deployed on a machine with multiple network cards. If a specific IP address is not specified, then ctasd will monitor traffic on all available IP addresses.
Default value: commented setting (monitor all IP addresses)

2.1.8 [Outbound]

The [Outbound] section of the ctasd.conf defines the parameters needed to configure the Outbound Spam Protection Service. By default, this is not enabled. To enable this service, you must receive a new license key from CYREN, which will provision this service on the Datacenter, and enable the service in the OutboundEnabled option in the [General] section of the ctasd.conf. Many of the parameters in this section relate to individual senders, as determined by the SenderID value. Once identified, counters are maintained for each SenderID. Policies that are set by external management applications (of the partner or OEM), can use these values to determine policies.

CountersMask=<integer>

There are several SenderID counters. However, not all customers may want to set thresholds or counters for each option. Therefore, there are Counter Masks created to enable the customer to mask, or hide, counters that are not being used. The CountersMask parameter enables you to define which SenderID counters should be enabled.

The CountersMask is a bit-wise parameter that represents the total of all enabled counters. In order to enable a counter, add the value of the counter to the total CountersMask value. By default, the SuspectedCounter, TotalMessagesCounter and SpamCounter are enabled, while the others are disabled. So the default value of the CountersMask parameter will be 7 (1+2+4). The full list of available flag values per SenderID counter is available in the configuration file.

The following displays the flag values per each SenderID counter:

- SuspectedCounter = 1
- TotalMessagesCounter = 2
- SpamCounter = 4
- BulkCounter = 8
- ConfirmedCounter = 16
- RecipientsCounter = 32

- VirusCounter = 64

Counter mask options include masking or unmasking the following counter options:

Suspected	The number of suspected messages by a SenderID within a given time window.
TotalMessages	Total number of messages a SenderID can send within a given time window.
Spam	The combined number of confirmed spam and bulk messages by a SenderID within a given time window.
Bulk	The number of bulk messages by a SenderID within a given time window.
Confirmed	The number of confirmed spam messages by a SenderID within a given time window.
Recipients	The number of recipients that can be specified by a SenderID within a given time window.
Virus	The number of virus messages by a SenderID within a given time window.

SenderIDWindowSize = <integer>

The SenderIDWindowSize option defines the duration of each time window managed per SenderID counter. This parameter indicates the update rate of the SenderID counter values. The time window is defined in seconds.

Default value: 60

SenderIDWindows = <integer>

The SenderIDWindows option defines the number of time windows to be managed for each SenderID. The SenderIDWindowSize multiplied by the number of SenderIDWindows totals to the timeframe that each SenderID counter value is maintained. The default value is 5 minutes (60 Sec * 5 = 5 min).

Default value: 5

CacheMaxEntries= <value>

The CacheMaxEntries parameter defines the maximum cache values for Outbound Spam counters. Once the cache reaches this amount, the least recently used counters are overwritten with newer ones, thus limiting the cache counter records to the specified value set by this parameter.

Default value: 1,000,000

2.1.1.8.1 SenderID-Related Parameters

The following parameters in the ctasd.conf file relate to each

SenderIdHeaderName = <string>

The SenderIDHeaderName parameter defines the name of the message header from which to extract the SenderID. Example message headers are: From, Reply-To headers. If the SenderID is not sent explicitly in each message, then this parameter value must be defined. Default value: From header.

SenderIdHeaderFormat = <string>

SenderIdHeaderFormat specifies if the value of the header will be taken "as is" or only the email address will be extracted from the header. Place the value "raw" when taking the value as-is. Place the value "email" when taking the email only.

Default value: email

SenderIdIgnoreList = <IP address:mask>, <IP address:mask>...

SenderIdIgnoreList is applicable only if the "Received" header was defined in SenderIDHeaderName. The SenderID is extracted from the last Received header. The system will ignore all SenderIDs appearing in the SenderIdIgnoreList

SenderIdWhiteListFile = <filename>

SenderIdWhiteListFile is the file name to manage white-listed SenderIDs. If no path is defined, the file will be created in the ctasd local directory.

Default value: senderid_white

SenderIdBlueListFile = <filename>

SenderIdBlueListFile is the file name to manage blue-listed SenderIDs. If no path is defined, the file will be created in the ctasd local directory. For more information, see [White List and Blue List](#).

Default value: senderid_blue

SenderIdReportingInterval = <number>

The reporting time interval in seconds for alerting repeating crossed thresholds.

Default value: 600

ReportCounters = 0

The ReportCounters parameter is a Flag that defines if the ctasd ClassifyMessage response message should include the counter values. If the value of "0" is defined, then the counter values will not be included in the ClassifyMessage response. If the value of "1" is defined, then the counter values will be included in the ClassifyMessage response. It should be noted that only enabled counters are reported in the ClassifyMessage response.

Default value: 0

2.1.8.2 Threshold Configuration

The following parameters, which are included in the Outbound section of the configuration file, enable you to set thresholds for the SenderID counters. You have the option of setting up to three thresholds per counter. These should be named with consecutive integers. For example: SuspectedThreshold1, SuspectedThreshold2, and SuspectedThreshold3.

You can then set policies based on which of the thresholds are crossed. There are no defaults set for thresholds. A threshold with no value indicates that this threshold is not in use.

Note: *ctasd will report on the highest threshold crossed per time window only. This means that if SuspectedThreshold3 is defined and crossed, it is reasonable to assume that both SuspectedThreshold1 and SuspectedThreshold2 have also been crossed.*

SuspectedThreshold1 = <number>

The first threshold of the SenderID Suspected counter.

SpamThreshold1 = <number>

The first threshold of the SenderID Spam counter.

BulkThreshold1 = <number>

The first threshold of the SenderID Bulk counter.

ConfirmedThreshold1 = <number>

The first threshold of the SenderID Confirmed counter.

VirusThreshold1 = <number>

The first threshold of the SenderID Virus counter.

TotalThreshold1 = <number>

The first threshold of the SenderID Total counter.

RecipientsThreshold1 = <number>

The first threshold of the SenderID Recipients counter.

[LocalView]

The [LocalView] section of the ctasd.conf defines the parameters needed to configure the LocalView rules-based engine. By default, this is not enabled and it is not relevant for Outbound traffic. This section of the ctasd.conf can be ignored when provisioning for the Outbound Spam Protection services.

2.2 Optional Configuration Parameters

The following parameters are optional. They are commented out and need to be manually enabled, if used.

2.2.1 [Connectivity]

ProxyPort = <port number>

Specifies the port number used for connectivity with the proxy server.

ProxyServerAddress = <address>

Specifies the host name or IP address of the proxy server.

ProxyAuth = <Authentication mode>

Specifies the authentication mode for connectivity with the proxy server. Options include: `Basic`, or `NoAuth`.

ProxyUserName = <user name>

The name of an authorized user.

ProxyPassword = <password>

The password of the authorized user.

ProxyAccess =

When using a proxy server, this value should be set to 1. This indicates that connectivity with the Internet is via a proxy server. This number should not be changed. A value of 0 indicates that a proxy server is not being used in this network topology.

Note: *If changes are made to the `ctasd.conf` file while the daemon is running, `ctasd` must be stopped and restarted. See the following section for more details. This is not relevant for the `ctasd` Windows version.*

Note: *When using the daemon with proxy (i.e. `ProxyAccess=1`), it is recommended to set the `MaxThreads` for the HTTP server to **35 or below** in order to avoid ephemeral port starvation.*

3 Running ctasd

ctasd may be run as a service from init.d or interactively. ctasd tries to load libasapsdk.so from the local folder and if it cannot find it there, it looks in the system search path (LD_LIBRARY_PATH).

3.1 Command Line Options

Usage: ctasd [OPTIONS]

-r	For Windows-only platforms: name register service
-p <name>	For Windows-only platforms: display name for registration (default: ctasd). This option must be defined together with the -r option.
-p <path>	For non-Windows platforms: creates a PID file and place it in the specified path location. By default, the path and PID file name is: [/var/run/ctasd.pid]. This option is not applicable for Windows platforms.
-u <name>	For Windows-only platforms: un-register service
-c <path>	Changes the location of the configuration file by specifying -c and the new path. The default location of the configuration file is etc/ctasd.conf
-l	Runs ctasd in interactive mode
-l <port>	For Windows-only platforms: udp log port
-s or -logfac	Determines the numerical value of the syslog facility to use. Default is local0 facility, whose numerical value is 16. A negative facility number will disable syslog event notifications.
-v	Prints version information and exits.
-? or -h	Displays the help screen.

3.2 Stopping ctasd

If you are running ctasd interactively, you can kill the daemon using Ctrl-C. Alternatively, if you are not running it in interactive mode, you can send a SIGTERM to the process identified in ctasd.pid file or in the file created with the `-p` option above.

4 ctasd Protocol

The querying device is developed and implemented by the Service Provider according to the protocol detailed in the following sections. It is then integrated with ctasd and the messaging network according to one or more of the scenarios in [ctasd Deployment Options](#).

The querying device receives outbound messages from the messaging network and for each message it generates and posts a request to ctasd to classify the message. The protocol of ctasd has been extended in the Outbound Spam Protection service to include detection and reporting of any spamming or suspected spamming behavior by a sender in the classify message response notifications.

In addition, the ctasd protocol has been extended for the Outbound Spam Protection service to manage SenderID lists to track sender spamming activity.

Communication between the querying device and ctasd is made over HTTP 1.0. The querying device connects to a TCP port on the ctasd daemon as prescribed in ctasd.conf.

4.1 Request and Response Conventions

Both the requests and the responses contain information with the following conventions:

- Requests and responses consist of data blocks such as a series of headers or the message body, etc.
- Data blocks are separated by CRLF.
- Headers support multiple fields, one line per-field.
- Fields support multiple values with the following syntax: field:value;value;... the semi-colon ';' delimiter is used to separate between values in the field.
- Values may span multiple lines. Each line begins with a white space.

The request envelope includes CYREN-related data. The structure of the header is extensible, meaning that the order of the headers is not mandatory and not all headers are required to be included in all requests and responses. CYREN uses and requires standard rfc822 headers structure.

4.2 ClassifyMessage_File

4.2.1 Method: ClassifyMessage_File

The ClassifyMessage_File request references a file. In this case, the <path> is conveyed to ctasd by the querying device and ctasd opens the file from the specified location. When using this option make sure that ctasd has sufficient access permissions to the files' location. For an example of the syntax and contents, see [Sample HTTP Request with File Reference](#).

Note: When integrating both anti-spam email service and CYREN's Command Antivirus protection, you should operate ctasd in synchronous communication. When operating in asynchronous communication, ctasd will only return spam classifications. Alternatively, consider using the ClassifyObject function, which will return both spam and virus classifications.

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	An optional value that holds the CYREN license key. This header is used in conjunction with the UseAuthMode=1 option in ctasd.conf and only when deploying ctasd over WAN.
X-CTCH-SenderIP	The IP address of the SMTP that sent the message. This header is not always available to be included in the request and therefore is not mandatory.
X-CTCH-MailFrom	The mailfrom of the message envelope. This header is also not always available to be included in the request and therefore is not mandatory.
X-CTCH-SenderID	This is the SenderID of the outbound message. This additional header is required only if the outbound spam service was configured in ctasd.conf to explicitly receive the SenderID.
X-CTCH-FileName	The path to the message file requires filtering. Verify that ctasd has access permission to the files.

X-CTCH-RcptCount	The recipient's count of the message. This header is optional.
------------------	--

X-CTCH-SenderIP

The IP address of the sending SMTP machine is an important value, as it is a valuable indicator of spammer sources and is very hard to fake.

After the message is received, it is stamped with the IP address and host name of the sending SMTP machine in the **'Received:'** message header.

- If the message "hopped" a few times on its way to the recipient(s), each time it completed a hop the address of the sending SMTP machine is stamped, or added to the header.
- Therefore, the information in this header describes the message's entire journey from the sender to the recipient's mail server.

Some ISP and Enterprise organizations route messages through a series of internal mail servers before redirecting the messages to the **Detection Client** for filtering. The X-CTCH-SenderIP is an optional header in the request-envelope data block.

While you may want to assume that the address at the bottom of the chain in the **'Received:'** message header representing the actual SMTP machine that was used by the spammer, this is unfortunately not possible. Smart spammers are aware of this assumption and try to circumvent it to disguise their origin by inserting faked addresses at the beginning of the list.

Nevertheless, the address at the top of the list contains valuable hints to CYREN because it may expose SMTP machines that spammers use or abuse frequently (i.e. 'zombies'). This address is compared at the CYREN Datacenter against dynamically-generated lists of both 'bad' and 'good' IP addresses and is used for a host of applications.

Although not mandatory, it is highly important that you will setup the IP Ignore List parameter in ctasd.conf to have ctasd effectively clear your internal mail servers from suspicious and strip them from the top of the **'Received:'** message headers down the list to the first real external sending SMTP server.

X-CTCH-MailFrom

The 'mailfrom' email address can be used to determine the likelihood of a message being spam. However, this is an optional header in the request-envelope data block.

X-CTCH-SenderID

This additional header is required only if the outbound spam service was configured in ctasd.conf to explicitly receive the SenderID from the ClassifyMessage request. If you leave the SenderIDHeaderName value in the ctasd.conf file empty, by default the From header value will be used.

X-CTCH-RcptCount

This additional header is required only if the Service Provider implements any Recipients based outbound spam protection policies. The host application is required.

Sample HTTP Request with File Reference

URL: http://host:port/ctasd/ClassifyMessage_File, method: POST

HTTP Headers	POST /ctasd/ClassifyMessage_File HTTP/1.0 Accept-Language: en-us Accept: */* Content-Length: 3867 Host: 150.215.71.23 User-Agent: CYREN HTTP Client
POST data: request envelope >>	X-CTCH-PVer: 0000001 X-CTCH-SenderIP: 150.215.71.29 X-CTCH-MailFrom: uxg4pbb6y@yahoo.com X-CTCH-SenderID: bary008@bxyxy.com X-CTCH-FileName: /var/spool/incoming/00000E44E1.eml X-CTCH-Rcptcount: 10

4.2.1.1 Response to ClassifyMessage_File Request

The response to ClassifyMessage_File consists of the following data blocks:

- HTTP response header
- Response envelope
- Errors (if any occur)

The HTTP response headers are standard HTTP 1.0.

Header	Explanation
X-CTCH-Pver	The CYREN protocol version. The protocol of this version is 0000001.
X-CTCH-Spam	This is the Spam classification of the message

Header	Explanation
	for which a query was sent by ctasd to the Datacenter. Options are: Confirmed, Bulk, Suspected, Unknown and NonSpam. For more explanation, see Spam Classifications .
X-CTCH-VOD	This is the VOD classification of the message for which a query was sent by ctasd to the Datacenter. Options are: Virus, High, Medium, Unknown and NonVirus. For more explanation, see Virus Threat Level Classifications .
X-CTCH-Flags	This is the value of the classification flags of the message for which a query was sent by ctasd to the Datacenter. This is a bitwise value.
X-CTCH-RefID	This is a value representing the transaction between ctasd and the Datacenter on behalf of the message and is used for various technical support diagnostics by CYREN. It is very important that you keep the RefID with the filtered message (e.g., as an X-header).
X-CTCH-SenderID	This is the SenderID that was included in the ClassifyMessage request message or alternately extracted from the outbound message using the defined <i>SenderIDHeaderName</i> in ctasd.conf.
X-CTCH-SenderID-Flags	The Sender ID flags provide notifications regarding SenderID counter thresholds crossing. The SenderID-Flags is a bitwise flag with a predefined value set.
X-CTCH-SenderID-TotalMessages	This is the value of the Total Messages counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalSpam	This is the value of the Spam counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled

Header	Explanation
	(value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalSuspected	This is the counter value of the Suspected counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalBulk	This is the counter value of the Bulk counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalConfirmed	This is the counter value of the Confirmed counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalRecipients	This is the counter value of the Recipients counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalVirus	This is the counter value of the Virus counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.

4.2.2 ClassifyMessage_File Response

X-CTCH-SenderID

This is the SenderID that was included in the ClassifyMessage request message or alternately extracted from the outbound message using the defined *SenderIDHeaderName* (in ctasd.conf).

X-CTCH-SenderID-Flags

The SenderID-Flags provide notifications regarding SenderID counter threshold crossings. The SenderID-Flags is a bitwise flag with a predefined value set. The following table describes the value received for each threshold crossing:

Note: *Where more than one threshold is crossed during a time window, ctasd notifies for the highest threshold value only, though the notification itself indicates previous thresholds were also crossed.*

bit #	Value	Description
0	1	Reserved.
1-3	2, 4, 8	Notification of crossing <i>SuspectedThreshold1</i> , <i>SuspectedThreshold2</i> , or <i>SuspectedThreshold3</i> A sample message is sent to CYREN SpamLab for further inspection.
4-6	16, 32, 64	Notification of crossing <i>SpamThreshold1</i> , <i>SpamThreshold2</i> , or <i>SpamThreshold3</i> .
7-9	128, 256, 512	Notification of crossing <i>TotalThreshold1</i> , <i>TotalThreshold2</i> , or <i>TotalThreshold3</i> .
10-12	1024,2048,4096	Notification of crossing <i>BulkThreshold1</i> , <i>BulkThreshold2</i> , or <i>BulkThreshold3</i>
13-15	8192,16384,32768	Notification of crossing <i>ConfirmedThreshold1</i> , <i>ConfirmedThreshold2</i> , or <i>ConfirmedThreshold3</i>
16-18	65536,131072,262144	Notification of crossing <i>RecipientsThreshold1</i> , <i>RecipientsThreshold2</i> , or <i>RecipientsThreshold3</i>
19-21	524288,1048576,2097152	Notification of crossing <i>VirusThreshold1</i> , <i>VirusThreshold2</i> , or <i>VirusThreshold3</i>

Notes: *In order to avoid overloading the system with repeated alerts, an alert suppression mechanism is offered. This alert suppression logic is defined by configuring the *SenderIDReportingInterval* parameter in *ctasd.conf*.*

Thresholds for the Total Message counter and a Spam counter or a Suspected counter can be crossed simultaneously.

4.2.2.1 Sample `ClassifyMessage_File` Response

	HTTP/1.0 200 OK
HTTP headers >>	Date: Sat, 12 May 2006 22:25:21 GMT
	Server: 165.34.21.87
	Content-Length: 122
	Connection: close
	Content-Type: text/plain
Response	X-CTCH-PVer: 0000001
	X-CTCH-Spam: Confirmed
	X-CTCH-VOD: High
	X-CTCH-Flags: 0
	X-CTCH-REFID: str=0001.0A090204.43D8B3EB.0038,ss=4,vl=2,fgs=0
	X-CTCH-SenderID= bary008@bxyxy.com
	X-CTCH-SenderID-Flags=16
	X-CTCH-SenderID-TotalMessages=23
	X-CTCH-SenderID-TotalSpam=7
	X-CTCH-SenderID-TotalSuspected=0
	X-CTCH-SenderID-Bulk=4
	X-CTCH-SenderID-Confirmed=3
	X-CTCH-SenderID-Recipients=53
	X-CTCH-SenderID-Virus=2

4.3 Method: `ClassifyMessage_Inline`

The message headers and body are included in the request only if `ClassifyMessage_Inline` request method is used instead of `ClassifyMessage_File`. These are standard rfc822-compliant headers and body of the messages.

The request ends at the end of the message-body data block without a CRLF. In this case, the querying device opens the file and streams the message data to ctasd over HTTP. In this format, the message body can be included (optional).

4.3.1.1 Request Envelope

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The CYREN license key. This header is used in conjunction with the UseAuthMode=1 option in ctasd.conf and only when deploying ctasd over WAN.
X-CTCH-SenderIP	The IP address of the SMTP that sent the message. This header is not always available to be included in the request and therefore is not mandatory.
X-CTCH-MailFrom	The mailfrom of the message envelope. This header is also not always available to be included in the request and therefore is not mandatory.
X-CTCH-SenderID	This is the SenderID of the outbound message. This additional header is required only if the outbound spam service was configured in ctasd.conf to explicitly receive the SenderID.
X-CTCH-RcptCount	The recipients count of the message. This header is optional.

4.3.1.2 Request Body

The `ClassifyMessage_Inline` request includes the message headers and message body within the query.

4.3.2 Sample HTTP Request with Streaming

URL: `http://host:port/ctasd/ClassifyMessage_Inline`, method: POST

HTTP headers >>

POST /ctasd/ClassifyMessage_Inline HTTP/1.0

Accept-Language: en-us

Accept: */*

Content-Length: 3867

Host: 150.215.71.23

User-Agent: CYREN HTTP Client

POST data request
envelope >>

X-CTCH-PVer:0000001

X-CTCH-SenderIP: 150.215.71.29

X-CTCH-MailFrom: uxg4pbb6y@yahoo.com

X-CTCH-SenderID: bary008@bxyxy.com

X-CTCH-Rcptcount: 10

POST data
msg. headers >>

```
Received: FROM [218.5.5.228] By
c9diamond03.diamond.amadis.com;
Sun, 22 Jun 2003 05:28:32 -0800
Received: from 46sx.amgyw.net [150.215.71.17] by
216.163.188.55
with SMTP; Sun, 22 Jun 2003 17:21:18 +0100
Message-ID: <4b$$76w-$2d1e-lf6h4-1-0cxs7@tvu.809p8ve.1b>
From: "John Smith" <uxg4pbb6y@yahoo.com>
To: bob@company.com
Subject:confirm spam classification test
Date: Sun, 22 Jun 03 17:21:18 GMT
X-Mailer: The Bat! (v1.52f) Business
MIME-Version: 1.0
Content-Type: multipart/alternative;
    boundary="B6B_273_5877FA._0FCA._7"
X-Priority: 3
X-MSMail-Priority: Normal
Return-Path: uxg4pbb6y@yahoo.com
X-CTCH-ID: _B32353B3-8A3E-47EA-889F-
EF1FCOD19C62_
X-OriginalArrivalTime: 22 Jun 2003 12:31:51.0891 (UTC)
FILETIME=[42008A30:01C338BA]
```

This is a multi-part message in MIME format.

```
--B6B_273_5877FA._0FCA._7
Content-Type: text/html;
```

..

...all the message data is included here...

4.4 Response to ClassifyMessage_Inline Request

The HTTP response headers are standard HTTP 1.0. The response to ClassifyMessage_File consists of the following data blocks:

- HTTP response header
- Response envelope

- Errors (if any occur)

Header	Explanation
X-CTCH-Pver	The CYREN protocol version. The protocol of this version is 0000001.
X-CTCH-Spam	This is the Spam classification of the message for which a query was sent by ctasd to the Datacenter. Options are: Confirmed, Bulk, Suspected, Unknown and NonSpam. For more explanation, see Spam Classifications .
X-CTCH-VOD	This is the VOD classification of the message for which a query was sent by ctasd to the Datacenter. Options are: Virus, High, Medium, Unknown and NonVirus. For more explanation, see Virus Threat Level Classifications .
X-CTCH-Flags	This is the value of the classification flags of the message for which a query was sent by ctasd to the Datacenter. This is a bitwise value.
X-CTCH-RefID	This is a value representing the transaction between ctasd and the Datacenter on behalf of the message and is used for various technical support diagnostics by CYREN.
X-CTCH-SenderID	This is the SenderID that was included in the ClassifyMessage request message or alternately extracted from the outbound message using the defined <i>SenderIDHeaderName</i> in ctasd.conf.
X-CTCH-SenderID-Flags	The SenderID-Flags provide notifications regarding SenderID counter threshold crossing. The SenderID-Flags is a bitwise flag with a predefined value set.
X-CTCH-SenderID-TotalMessages	This is the value of the Total Messages counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalSpam	This is the value of the Spam counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.

Header	Explanation
X-CTCH-SenderID-TotalSuspected	This is the counter value of the Suspected counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalBulk	This is the counter value of the Bulk counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalConfirmed	This is the counter value of the Confirmed counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalRecipients	This is the counter value of the Recipients counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.
X-CTCH-SenderID-TotalVirus	This is the counter value of the Virus counter for this SenderID. This message header will appear only if the <i>ReportCounters</i> parameter in ctasd.conf is enabled (value=1) and if this counter was enabled in the CountersMask ctasd.conf parameter definition.

4.5 Method: UpdateSenderIDList Request

The ctasd protocol has been extended for the Outbound Spam Protection service to manage senderID lists. Two lists are offered:

- **A white list:** listing senders who have been cleared for any activity in the system including sending out suspected and spam message. No notifications will be triggered for white-listed senders.
- **A blue list:** listing senders who have been cleared only for sending out suspected messages but not spam messages. Notifications will be triggered for blue-listed senders sending out spam messages.

These are the headers that are required to perform an Outbound Spam updateSenderIDList Request:

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The CYREN license key. This header is used in conjunction with the UseAuthMode=1 option in ctasd.conf and only when deploying ctasd over WAN.
X-CTCH-SenderID	The SenderID that should be added or removed from list.
X-CTCH-ListName	The name of the list. Current version supports either a “white” or “blue” list name.
X-CTCH-ListCommand	The command to be performed on the list. The supported commands are “add” or “remove.”

4.6 UpdateSenderIDList Response

ctasd will respond with the protocol version, and return the http status header as either ‘http ok’ or a specific http error detailing the list command error.

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. The protocol of this version is 0000001.

4.7 Method: ReportFP Request

False positives are non-spam or non-malware messages which were incorrectly classified as spam or malware. The ReportFP method can be used for reporting a case of false positive (FP) back to the CYREN Datacenter through ctasd by supplying the Message's RefID and optionally the message itself as an additional information.

The ReportFP functionality enables you to include the entire message or portions of it and forward it to CYREN for analysis. For large emails, you should send only the RefID. There is no need to send the entire email. It is critical that the RefID always be included in the message sent in the request. Without the RefID, CYREN is unable to analyze the report.

Header	Explanation
X-CTCH-PVer	The CYREN <code>protocol</code> version. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The CYREN <code>license</code> key. This header is used in conjunction with the <code>UseAuthMode=1</code> option in <code>ctasd.conf</code> and only when deploying ctasd over WAN.
X-CTCH-Service	Indicates the <code>service</code> for which the message was incorrectly classified, whereas 1= <code>svcAntiSpam</code> service, 2= <code>svcVOD</code> service and 8 = <code>svcOutboundSpam</code> . Based on the service, the message is automatically routed to different analyzing procedures at CYREN.

4.7.1.1 Request Body

The message headers and message body should be included as described above for the `ClassifyMessage_File/Inline` requests.

4.8 Response to ReportFP Request

As an acknowledgement of receipt, ctasd returns a response containing the current protocol version of the CYREN protocol.

Header	Explanation
X-CTCH-PVer	The CYREN <code>protocol</code> version. The protocol of this version is 0000001.

4.9 Method: ReportFN Request

Since cases of false negatives are spam and malware messages that were misclassified as non-spam, using the ReportFN request will enable CYREN to determine why the message was misclassified and, more importantly, prevent similar messages from reaching your end-users in the future. When reporting cases of false negative, the entire message is needed by CYREN for analysis.

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The CYREN license key. This header is used in conjunction with the UseAuthMode=1 option in ctasd.conf and only when deploying ctasd over WAN.
X-CTCH-Service	Indicates the service for which the message was incorrectly classified, whereas 1= svcAntiSpam service, 2= svcVOD service, 4=svcWebSec service and 8 = svcOutboundSpam service. Based on the service, the message is automatically routed to different analyzing procedures at CYREN.

4.9.1.1 Request Body

Include the message headers and message body as described above about `ClassifyMessage_File/Inline` requests.

4.10 Response to ReportFN Request

As an acknowledgement of receipt, ctasd returns a response containing the current protocol version of the CYREN protocol.

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. The protocol of this version is 0000001.

4.11 Method: GetServices Request

The querying device can query ctasd to determine which CYREN services are provisioned to the customer using ctasd, based on the license key associated with the ctasd. The query includes the following:

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The CYREN license key. This header is used in conjunction with the UseAuthMode=1 option in ctasd.conf and only when deploying ctasd over WAN.

4.12 Response to GetServices Request

ctasd will respond with the protocol version as well as the services that are currently provisioned to the license key used by ctasd. The response to the request will include the following:

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. The protocol of this version is 0000001.
X-CTCH-Services	When queried, CYREN will respond with the appropriate services currently provisioned for the license key used by ctasd, whereas 1= Anti-spam and 2 = Zero-Hour virus protection, 4=URL Filtering and 8=Outbound Spam. An Outbound service always have the Anti-Spam enabled, and will give a value of 9 (8+1).

4.13 Method: GetStatus Request

The querying device can query ctasd to validate connectivity with the ctasd unit and the CYREN Datacenter.

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The CYREN license key. This header is used in conjunction with the UseAuthMode=1 option in ctasd.conf and only when

	deploying ctasd over WAN.
--	---------------------------

4.14 Response to GetStatus Request

ctasd will respond with the protocol version, and return the http status header as either 'http ok' or a specific http error detailing the connectivity problem.

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. The protocol of this version is 0000001.

4.15 Error Handling

When an error occurs, either during the sending of a `ClassifyMessageFile/Inline` request or when ctasd is sending a response back to the querying device, an error message is generated and sent to the querying device over HTTP.

The following HTTP error messages exist:

- 400 Bad Request
 - Such an error indicates that the client application that is using ctasd has provided a faulty request format. You should check your code at the integration phase and make sure no such errors are returned.
- 500 Internal Server Error
 - Such an error may indicate a network error or similar problem that prevents ctasd from accessing the Datacenter.

A typical error will contain the following envelope headers:

Header	Explanation
X-CTCH-PVer	The CYREN protocol version. The protocol of this version is 0000001.
X-CTCH-Error	Text string that explains the error that has occurred.

4.15.1 Sample HTTP Error

HTTP/1.0 400 Bad Request

Or

HTTP/1.0 500 Internal Server Error

4.15.1.1 HTTP headers

```
Date: Sat, 12 May 2006 22:25:21 GMT
Server: hostname
Content-Length: 122
Connection: close
Content-Type: text/plain
```

4.15.1.2 Response Envelope

```
X-CTCH-PVer: 0000001
X-CTCH-ERROR: 500 Network Error
```

4.16 SenderID

Header	Explanation
X-CTCH-SenderID	This is the SenderID of the outbound message. This additional header is required only if the outbound spam service was configured in ctasd.conf to explicitly receive the SenderID.

SenderIDHeaderName = <string>

The SenderIDHeaderName parameter defines the name of the message header from which to extract the SenderID. Example message headers are: From, Reply-To headers.

There is no default for this parameter. If the SenderID is not sent explicitly in each message, then this parameter value must be defined.

4.17 SenderID Definition and Threshold Crossing Notifications

Each Service Provider or OEM partner can configure the appropriate header to define the SenderID. This implementation decision is important as it will define how senders are tracked and managed in the system. The SenderID may be extracted from a message header or explicitly defined in the ClassifyMessage envelope.

More advanced SenderID logics, for example setting the SenderID to be the SMTP Auth account, and if not found, than setting the SenderID to be the message Sender IP, are supported. However, such configurations require contacting the CYREN Support team. A SenderID counter **threshold** is a limit set by the CYREN customer on each SenderID, as defined in the configuration file. If thresholds are set, a flag is raised when the threshold is crossed. There are several SenderID thresholds that can be defined. If within a given period of time, more than one threshold is crossed, only a flag for the highest value will be raised.

It can be assumed within that period of time that all other thresholds below the flagged one have also been crossed. Each counter can have up to three threshold options set (referred to as SpamCounter1, SpamCounter2, and SpamCounter3). Thus if all thresholds have been crossed, the SpamCounter3 flag is the only one that will be raised. The X-CTCH-SenderID-Flags is a bitwise flag with a predefined value set. The value received by the X-CTCH-SenderID-Flags indicates which counter threshold value has been crossed. By default, all of the flags are disabled. They must be enabled in the ctasd.conf before they can be used.

The following table describes the value received for each threshold crossing: Additionally, this table outlines some guidelines for how to handle the different event notifications.

Event Type	Optional Actions
Suspected Counter Threshold crossed	The number of suspected messages sent out by the sender in a pre-defined time window.
Spam Counter Threshold crossed	The number of confirmed spam and bulk messages sent out by the sender in a pre-defined time window.
Bulk Counter Threshold crossed	The number of bulk messages sent out by the sender in a pre-defined time window. The Bulk and Confirmed counters should be used instead of the Spam Counter if a Service Provider or OEM partner implements different policies for confirmed and bulk classified messages.
Confirmed Counter Threshold crossed	The number of confirmed messages sent out by the sender in a pre-defined time window. The Bulk and Confirmed counters should be used instead of the Spam Counter if a Service Provider or OEM partner implements different policies for confirmed and bulk classified messages.
Virus Counter Threshold crossed	Total number of Virus and High VOD classified messages sent out by the sender in a pre-defined time window.
Total Counter Threshold crossed	Total number of messages sent out by the sender in a pre-defined time window.
Recipients Counter Threshold crossed	Total number of recipients listed in the messages sent out by sender in a pre-defined time window.

5 The SpamAssassin Network Protocol

ctasd for SpamAssassin Network protocol The protocol for communication between spamc/spamd is similar to its protocol for HTTP. The request/response process is similar to the following:

```
spamc --> REPORT SPAMC/1.2\r\n
spamc --> Content-length: <size>\r\n
```

(optional) spamc --> User: <username>\r\n

```
spamc --> [optional \r\n-delimited headers...]
spamc --> \r\n [blank line]
spamc --> --message sent here--
```

```
spamd --> SPAMD/1.1 0 EX_OK\r\n
spamd --> Content-length: <size>\r\n
spamd --> [optional \r\n-delimited headers...]
spamd --> \r\n [blank line]
spamd --> --processed message sent here--
```

After each side is done writing, it shuts down its side of the connection.

5.1 Request and Response Conventions

Both the requests and the responses contain information with the following conventions:

The first line from spamc is the command for spamd to execute, followed by the protocol version.

The request may include a set of optional additional headers.

The first line of the response from spamd is the protocol version followed by a response code from `sysexit.h` followed by a response message string which describes the error, if there was one.

If the response is 0, the processed message will be sent.

If the response code is not 0, then the processed message will not be sent, and the socket will be closed after the first line is sent.

The response will include additional defined headers. Clients which do not support these headers will ignore them, and continue looking for headers which they do support, or the "\r\n\r\n" end-of-header's marker.

5.2 spamc Commands

The SpamAssassin Network Protocol supports different Query Commands. ctasd supports the REPORT command of the SpamAssassin Network Protocol:

Command	Description
REPORT	<p>The REPORT command returns a response code and message followed by a header called "Spam:" with a value of "True" or "False", then a semi-colon, then the score for this message, " / " and then the threshold. It is then followed immediately by the a text report generated by spamd:</p> <pre>SPAMD/1.1 0 EX_OK\r\n Spam: True ; 15 / 5\r\n \r\n This mail is probably spam. [...]</pre> <hr/> <p><i>Note: The Report Command is used by the Exim Mail Server.</i></p> <hr/>

5.3 SpamAssassin Network Protocol Headers Description

The following optional headers are defined as of protocol 1.4:

Header	Description
Content-length	Length of a request or response body.
Spam	Used in responses to the several spamc Commands. See "spamc Commands" section.
User	Username of the user on whose behalf this scan is being

	performed. The meaning of this is up to the server; format is that of a traditional UNIX username ([-A-Za-z0-9_]+).
--	---

5.4 Method: spamc command

The message headers and body are included in the request. These are in standard rfc822-compliant format.

5.4.1.1 Request Envelope

These optional headers may be added as part of the request envelope:

Header	Explanation
X-CTCH-SenderIP	The IP address of the SMTP that sent the message. This header is not always available to be included in the request and therefore is not mandatory.
X-CTCH-MailFrom	The mailfrom of the message envelope. This header is also not always available to be included in the request and therefore is not mandatory.
X-CTCH-SenderID	This is the SenderID of the outbound message. This additional header is required only if the Outbound Spam Protection Service was configured in ctasd.conf to explicitly receive the SenderID.
X-CTCH-RcptCount	<p>The recipient's count of the message. This header is optional.</p> <hr/> <p><i>Note: The X-CTCH-RcptCount header is applicable only to an Outbound solution.</i></p> <hr/>

5.4.1.2 Request Body

The spamc request includes the message headers and message body within the query.

5.4.1.3 Sample spamc Request

```
spamc headers >> Report SPAMC/1.2
User:User12
Content-Length: 3867
X-CTCH-SenderIP: 150.215.71.29
X-CTCH-MailFrom: ux@yahoo.com
```

POST data
 msg. headers

X-CTCH-SenderID: bary008@ctxx.com
 Received: FROM [218.5.5.228] By
 c9diamond03.diamond.amadis.com;
 Sun, 22 Jun 2003 05:28:32 -0800
 Received: from 46sx.amgyw.net [150.215.71.17] by
 216.163.188.55
 with SMTP; Sun, 22 Jun 2003 17:21:18 +0100
 Message-ID: <4b\$\$76w-\$2d1e-lf6h4-1-
 0cxs7@tvu.809p8ve.1b>
 From: "John Smith" <uxg4pbb6y@yahoo.com>
 To: bob@company.com
 Subject:confirm spam classification test
 mate: Sun, 22 Jun 03 17:21:18 GMT
 X-Mailer: The Bat! (v1.52f) Business
 MIME-Version: 1.0
 Content-Type: multipart/alternative;
 boundary="B6B_273_5877FA._0FCA._7"
 X-Priority: 3
 X-MSMail-Priority: Normal
 Return-Path: uxg4pbb6y@yahoo.com
 X-CTCH-ID: _B32353B3-8A3E-47EA-889F-
 EF1FC0D19C62_
 X-OriginalArrivalTime: 22 Jun 2003 12:31:51.0891 (UTC)
 FILETIME=[42008A30:01C338BA]

This is a multi-part message in MIME format.

POST data
 msg. body

--B6B_273_5877FA._0FCA._7
 Content-Type: text/html;

...all the message data is included here...

5.5 Response to spamc Command

The response to the Mail Server consists of the following data blocks:

- spamd response headers
- Response envelope
- Errors (if any)

5.5.1.1 Response Envelope

The Spam Assassin Network Protocol supports optional response headers. These optional headers are returned by ctasd.

5.6 Outbound Spam Detection Exim Response Sample

Following is a sample response generated by an Exim MTA implementing CYREN's Inbound Spam Detection service.

```
Envelope-to: bob@xyx.com
Delivery-date: Tue, 26 Jul 2011 12:35:20 +0000
from: mahedmshadg2mjgb@msjzdbh.sdf
to: mhasdmas@szd, j.asd
subject: msgjasd
Date: Tue, 26 Jul 2011 12:35:20 +0000
X-Spam-Score: 100.0
(+++++)
X-Spam-Report: X-CTCH-Spam: Confirmed
                X-CTCH-VOD: Unknown
                X-CTCH-Flags: 0
                X-CTCH-RefID:
wb,re=10.000,recr=0.000,recu=10.000,reip=0.000,pt=R_186404,cl
=4,cld=1,fgs=0
                X-CTCH-SenderID: mahedmshadg2mjgb@msjzdbh.sdf
                X-CTCH-SenderID-Flags: 8336
                X-CTCH-SenderID-TotalMessages: 1
                X-CTCH-SenderID-TotalSpam: 1
                X-CTCH-SenderID-TotalSuspected: 0
                X-CTCH-SenderID-TotalConfirmed: 1
                X-CTCH-SenderID-TotalBulk: 0
                X-CTCH-SenderID-TotalVirus: 0
```

```

X-CTCH-SenderID-TotalRecipients: 0
Subject: *SPAM* amsgjasd
tttttttt
sakjhsad
NewRule5

```

5.7 Exim for the SpamAssassin Network Protocol

Exim is a leading open source Mail Server. Exim supports the SpamAssassin Network Protocol. In order to seamlessly integrate ctasd with the Exim product, you need to configure ctasd to work with the SpamAssassin Network Protocol, and define the spamd parameters in ctasd.conf.

The following describes how Exim displays a ctasd SpamAssassin Network protocol response message. Only relevant response headers are listed here:

```

X-Spam-Status: Yes, score=10.0
X-Spam-Score: 100
X-Spam-Report: X-CTCH-Spam: Confirmed
X-CTCH-VOD: Unknown
X-CTCH-Flags: 32
X-CTCH-RefID:
str=0001.0A0B0208.4E06DAD5.010D,ss=4,re=0.000,pt=F_4810491,fg
s=32
X-Spam-Flag: YES

```

The following table describes the spam classification related headers:

Header	Explanation
X-Spam-Status	Yes/No indicates if the message is spam or not. Score is the spam threshold (as defined in ctasd.conf)
X-Spam-Score	The score of the message
X-Spam-Report	Includes all the optional spamd response headers, as defined by ctasd.
X-Spam-Flag	YES/NO flag if the message is spam or not.

6 SNMP Counters

To use the SNMP counters you can configure a special port in configuration file as described in the [stats] section. When you telnet to this port, the daemon will dump the data of the counters and will close the connection. You can write your own SNMP sub-agent that will access the data in the predefined port and will expose them in SNMP format. A sample of such a sub-agent is supplied by CYREN in the package.

The ctasd SNMP counters are detailed in the CTCH-CTASD-MIB.txt file.

6.1 General SNMP Counters

Counter Name	Description
Pid	The daemon's process ID.
Uptime	Total number of seconds elapsed since the daemon was started.
totalCenterRequests	Total number of queries forwarded to the Datacenter.
totalCommErrors	Total number of communication errors that occurred while trying to query the Datacenter.
totalCenterRequestTime	Total amount of time the client waited for a response from the Datacenter.
asapTotalClassifyMessageRequests	Total number of ClassifyMessage queries.
asapTotalClassifyMessageCenterRequests	Total number of ClassifyMessage queries sent to the Datacenter.
asapTotalClassifyMessageErrors	Total number of general errors in ClassifyMessage queries.

Counter Name	Description
asapTotalClassifyMessageCurrRequests	Total number of ClassifyMessage queries currently being processed.
asapTotalGetStatusRequests	Total number of Get Status queries.
asapTotalGetStatusErrors	Total number of errors that occurred while handling Get Status queries.
asapCacheSize	Number of records in the local cache.
asapCacheTotalHits	Total number of times a response to a query was found in the local cache.
asapCacheTotalMisses	Total number of times a query response was not found in the local cache.

6.2 Spam Classification Counters

Counter	Explanation
asaptotalSpamConfirm	Total number of Confirmed classifications returned by ctasd.
asapTotalSpamBulk	Total number of Bulk classifications returned by ctasd.
asaptotalSpamSuspected	Total number of Suspected classifications returned by ctasd.
asaptotalSpamUnknown	Total number of unknown spam classifications returned by ctasd.
asapTotalSpam	Total number of NonSpam classifications returned by ctasd.

6.3 HTTP Protocol Counters

Note: When the SpamAssassin-compatible network protocol is used, the HTTP protocol counters are not relevant.

Counter Name	Description
asapHttpCurrRequests	Number of HTTP queries that ctasd is currently processing.
asapHttpQueueSize	Number of HTTP connections waiting to be processed.
asapHttpTotalWaitTime	Total wait time for all HTTP queries in the queue.
asapHttpTotalClassifyMessageRequests	Total number of ClassifyMessage queries.
asapHttpTotalClassifyMessageErrors	Total number of errors that occurred as a result of ClassifyMessage queries which were sent to ctasd.
asapHttpTotalReportFPRequests	Total number of ReportFP queries sent to ctasd.
asapHttpTotalReportFPErrors	Total number of errors that occurred as a result of ReportFP queries which were sent to ctasd.
asapHttpTotalReportFNRequests	Total number of ReportFN queries sent to ctasd.
asapHttpTotalReportFNErrors	Total number of errors that occurred as a result of ReportFN queries which were sent to ctasd.
asapHttpTotalGetStatusRequests	Total number of GetStatus queries sent to ctasd.
asapHttpTotalGetStatusErrors	Total number of errors that occurred as a result of HTTP GetStatus queries sent to ctasd.

6.4 Outbound Spam SNMP Counters

Counter	Explanation
outBoundTotalWhiteListed	Total number of whitelisted SenderIDs.
outBoundTotalBlueListed	Total number of bluelisted SenderIDs.
outBoundSenderIdCacheSize	Number of SenderID records in the local cache.
outBoundPatternsCacheSize	Number of message pattern records in the local cache.

6.5 Outbound SenderID Threshold SNMP Counters

Counter	Explanation
outBoundTotalMessagesThresholdCrossed	Total number of outbound messages which crossed the threshold.
outBoundTotalBulkThresholdCrossed	Total number of outbound bulk messages which crossed the threshold.
outBoundTotalSuspectedThresholdCrossed	Total number of outbound suspected messages which crossed the threshold.
outBoundTotalSpamThresholdCrossed	Total number of outbound spam messages which crossed the threshold.
outBoundTotalVirusThresholdCrossed	Total number of outbound virus messages which crossed the threshold.
outBoundTotalConfirmedThresholdCrossed	Total number of outbound confirmed messages which crossed the threshold.
outBoundTotalRecipientsThresholdCrossed	Total number of outbound recipients messages which crossed the threshold.

6.6 Spamd Protocol Counters

Note: When the standard HTTP-like ctasd protocol is used, the spamd protocol counters are not relevant.

Counter Name	Description
asapSpamdCurrRequests	Number of spamd queries that ctasd is currently processing.
asapSpamdQueueSize	Number of spamd connections waiting to be processed.
asapSpamdTotalWaitTime	Total wait time for all spamd queries in the queue.
asapSpamdTotalClassifyMessageRequests	Total number of ClassifyMessage queries.
asapSpamdTotalClassifyMessageErrors	Total number of errors that occurred as a result of ClassifyMessage queries which were sent to ctasd.

6.7 Virus Outbreak Detection SNMP Counters

Counter	Explanation
asaptotalVodVirus	Total number of confirmed <code>Virus</code> classifications returned by ctasd.
asaptotalVodHigh	Total number of <code>High</code> malware risk classifications returned by ctasd.
asaptotalVodMedium	Total number of <code>Medium</code> malware risk classifications returned by ctasd.
asaptotalVodUnknown	Total number of <code>Unknown</code> malware classifications returned by ctasd.
asaptotalVodNonVirus	Total number of <code>NonVirus</code> classifications returned by

Counter	Explanation
	ctasd.

6.8 Command Antivirus SNMP Counters

Counter	Explanation
avTotalClassifyObjectRequests	Total number of ClassifyObject queries.
avTotalClassifyObjectErrors	Total number of general errors in ClassifyObject queries.
avTotalItemsScannedInObjects	Total number of items that were scanned in ClassifyObject.
avTotalObjectsInfected	Total number of classify Objects that at least one thread was found.
avTotalObjectsThreats	Total number of threats found in scanned ClassifyObjects.
avHttpTotalClassifyObjectRequests	Total number of ClassifyObject queries which were sent to ctasd over HTTP protocol.
avHttpTotalClassifyObjectErrors	Total number of errors that occurred as a result of ClassifyObject queries which were sent to ctasd over HTTP protocol.

7 Deploying ctasd Over WAN

While ctasd is typically deployed within the customer's premises and LAN for best performance, it is possible to deploy ctasd over WAN and remotely to the querying devices. This deployment scenario comes with some limitations as described below but in some cases it is justified and an essential solution for specific cases.

When deploying ctasd over WAN, you must be aware of the following limitations:

- Authentication of the querying devices is required to avoid misuse or abuse by unauthorized sources and devices (use the X-CTCH-Key header).
- There will be no local cache next to the querying device.
- It is recommended that you provision a failover mechanism to the remote ctasd units as described later in this section.
- `ClassifyMessage_Inline` becomes the practical method in this scenario and not `ClassifyMessage_File`.

To deploy ctasd over WAN follow these steps:

- Set the option `UseAuthMode=1` in `ctasd.conf` to instruct ctasd to serve only requests that come with a valid X-CTCH-Key value.
- Make sure that each request to ctasd (except in the case of `GetStatus` request) contains the header `X-CTCH-Key`. Use the same license key that is used in `ctasd.conf`.

7.1 Implementing a Failover Mechanism

ctasd is designed with an internal failover mechanism allowing it to connect to any CYREN Datacenter, worldwide. This is done automatically and requires no special settings by the ctasd operators.

When deploying ctasd over WAN, the connectivity between the querying devices and ctasd units may be interrupted from time to time and the responsibility for enabling connectivity continuity to the ctasd unit is with the Service Provider or OEM partner deploying ctasd.

If you plan a failover mechanism to ctasd units that are deployed over WAN follow these guidelines:

- Check the IP addresses of all ctasd units periodically, (i.e., `gethostbyname`) to find out if one or more addresses were changed (a good mechanism will check every 30-50 seconds).

- Try to maintain connectivity with the first responding ctasd unit rather than switching back and forth frequently between ctasd units.
- If the last-used ctasd unit is not responding, you may implement a retry mechanism for several times (i.e., 3 times) and only then attempt to connect to the next ctasd unit.
- Switch back to the first ctasd unit if connectivity with it was resumed after a failure and if it is available for several consecutive connection attempts.

8 ctasd Testing and Verification

The best way to properly test ctasd's effectiveness is to test it against a known corpus of messages.

When evaluating spam and malware detection using ctasd, you should be aware that ctasd was designed to protect all users from massive spam and virus outbreaks that are in progress in real-time. This means that you must test ctasd with:

- **Current messages rather than old messages.** Inactive message patterns, for example those resulting from outbreaks that no longer populate the Internet, may have already been removed from the CYREN Classification Database. For a successful test ensure you evaluate with messages that are not older than 5 days.
- **Standard SMTP-compliant messages.** You should only use messages that comply with the SMTP standard (rfc822) and include representations of massive outbreaks.
- **Threat messages.** It is important that the messages used for testing contain recognized spam, phishing and/or malware patterns. Unlike most solutions, ctasd does not rely on a lexical analysis of the content of an email message; therefore, it is not enough to put some "bad" words in an email. The messages must be part of known outbreaks that currently populate the Internet.

When evaluating local spam patterns detection, please select and include a local message to the verification test. You are then required to send the message at a minimum rate of 4 messages/ 5 minutes for this pattern to be detected as a local pattern.

Verify that you have specified all the necessary parameters required in the configuration file (by default, ctasd.conf). Note that you may also perform the test via Proxy Server. For more details, contact your CYREN representative.

It is highly recommended that you perform an evaluation by first running a series of previously-tested messages as specified below and then compare the results to verify that the expected classifications by ctasd were properly assigned to each test message.

8.1 Connectivity Test

ctasd performs an automatic connectivity test with the CYREN Datacenter at startup and, if it is unable to connect, generates and displays an error message. If necessary, restart the daemon manually to see if an error message is generated at startup. ctasd is in contact with the Datacenter and therefore "knows" if communication is unavailable. Should this occur, ctasd will not send queries until communication is restored and will try connecting to a different Datacenter. During

the time that communication with the Datacenter is unavailable, detection and filtering continues uninterrupted, provided that the local cache option is enabled and contains information.

If the administrator is unable to achieve connectivity after launching the daemon, the following checks should be performed:

- Use `'telnet resolver1.ctmail.com 80'` from the same machine running ctasd to validate the connection. If you are unable to connect using Telnet, it is possible that the license key code entry in the configuration file was entered incorrectly. Check the license key code in the configuration file and correct if necessary.
- Confirm that connection with the Datacenter is not via a server proxy or, if it is through a server proxy, that the appropriate parameters have been added to the configuration file.
- Confirm that there is no network problem and that connectivity with the Internet is possible.

If the CYREN daemon is still unable to connect to the Datacenter, contact your CYREN technical support representative or email to support@cyren.com.

8.2 Acceptance Test

To test and evaluate ctasd after deployment, follow these steps:

1. Modify ctasd.conf configuration file.
2. Launch the ctasd daemon.
3. Execute either the sample code **http_client.pl** or **socket_client.pl** with reference to a directory into which you have stored test messages to perform Anti-Spam and VOD testing.
4. Execute the sample code **http_client_av.pl** reference to a directory into which you have stored test messages to perform Command Anti-Virus testing.
5. Evaluate the messages to confirm that they were classified correctly.

8.3 Corpus of Messages for Evaluation

To initialize the evaluation procedure, you may want to use the following corpus of files that are included in the package to confirm that ctasd is properly deployed and that communication with the CYREN Datacenter yields expected results for predefined testing patterns. The files are included in the `./corpus/` sub-directory.

You may also include additional test messages in this directory by following these general guidelines:

- Spam messages should not be older than 10 days.
- Messages with viruses should be part of a current outbreak.
- Local pattern messages (not included in the ctasd package)

8.3.1 For Spam classification

You can use the following files to test ctasd integration for spam classification. These files are located in the zipped file: EvalCorpus.zip.

File name	Expected Result	Classification Source
msg_cs.eml	Confirmed	Datacenter
msg_bs.eml	Bulk	Datacenter
msg_us.eml	Unknown	Datacenter
msg_ns.eml	NonSpam	Datacenter
msg_cs_cache.eml	Confirmed	Local Spam cache
msg_cr.eml	Confirmed	Local Proactive Patterns Engine
msg_sh.eml	Legitimate mass-mailing newsletter	Datacenter

8.3.2 For Virus Outbreak Detection

You can use the following files to test the ctasd integration for VOD-based classification.

File name	Expected Result	Comments
ctchvodv.ex_	Confirmed virus	Before using this file, you should change the name from ctchvodv.ex_ to ctchvodv.exe.
ctchvodh.ex_	High risk	Before using this file, you should change the name from ctchvodh.ex_ to ctchvodh.exe.
ctchvodm.ex_	Medium risk	Before using this file, you should change the name from ctchvodm.ex_ to chctvodm.exe
Eicar files: eicar.com and	Confirmed virus	http://www.eicar.org/anti_virus_test_file.htm

eicar.zip		
-----------	--	--

Make sure to embed these files within rfc822-compliant messages when conducting a VOD test.

8.3.3 For Command Anti-Virus

You can use the following files to test the ctasd Command Anti-Virus classification data.

File name	Expected Result	Comments
Eicar files: eicar.com and eicar.zip	Infected	http://www.eicar.org/anti_virus_test_file.htm

Example of Command Antivirus Eicar Response

```
X-CTCH-AV-ThreatsCount: 1
X-CTCH-AV-ScanResult: Infected
X-CTCH-AV-DetectionType: Virus
X-CTCH-AV-DetectionAccuracy: Exact
X-CTCH-AV-DetectionName: EICAR_Test_File
```

8.3.4 For SenderID Notifications

When evaluating the Outbound Spam Protection service's ability to track and issue alerts regarding spammer activity, you must send suspected spam and spam messages from the same SenderID at a rate higher than the one defined by the threshold settings in the ctasd.conf file.

8.4 Running the Sample Client

The sample scripts are Perl-based. Before running the sample clients make sure that the Perl environment is installed on your testing machine. Perl source and binary package can be downloaded freely from any number of locations on the Internet and for some operating systems it is already included in the base installation by default.

ctasd package includes two optional sample client codes, named **http_client.pl** and **socket_client.pl**.

- **http_client.pl** uses a standard http library and is preferred for use in order to emulate exactly the communication between a querying device and ctasd daemon.

- **socket_client.pl** may be used when the tester is unable to use a standard http library for testing. Instead, this sample code opens a socket to the ctasd daemon.

Other than the above difference, both sample clients are used and operate the same way and produce the same results.

Using the sample client you may connect to ctasd and evaluate the entire process quickly. The client demonstrates the use of the protocol and is not a complete application within itself.

In order to finalize the deployment process on a fully productive environment you should create your own client. Make sure to pass the hostname with the file reference if the client is deployed on a different host than the one used for ctasd.

The client is responsible for delivering the following services:

- Connecting to ctasd on a predefined TCP port as prescribed in ctasd.conf.
- Passing message data to ctasd by reference to files.
- Receiving classifications per-message from ctasd (Spam or VOD).
- Receiving SenderID threshold crossing notifications from ctasd.
- Receiving the CYREN transaction reference record (RefID), per-message from ctasd. It is highly recommended that you keep this record with the message for technical support purposes (i.e., attach it to the message as an X-header such as X-CTCH-RefID:<RefID record>).

Usage: http_client.pl [OPTIONS] DIR-NAME

OPTIONS

--stream

Send stream through HTTP session

--host <hostname>

Ctasd host name or IP address

-p, --port <port number>

Port number, for example [8088]

-m, --mailfrom <mailfrom>

MailFrom address, for example [sender@domain.com]

--senderip <senderip>

SenderIP address, for example [111.222.3.4]

-, --help

Show the help screen

Notes:

- **http_client.pl** will scan the DIR-NAME (recursively) and for each file within it will generate a separate request to ctasd for filtering. The messages must be rfc822-compliant.
- If you do not specify a port, than the default TCP port is assumed.
- The -m and -senderip are optional parameters.
- Verify that ctasd has Access permissions to load the files for filtering.
- When you create your own client you may choose how to input messages to ctasd. Options are: reference to a file (as sampled by **http_client.pl**) or stream the message data as prescribed later in this document.
- Output is sent to **stdout**.
- To execute the client successfully, verify that you have the following modules installed:
- LWP::UserAgent and HttpRequest::Common available from the LWP (The World-Wide Web library for Perl) library at <http://search.cpan.org/dist/libwww-perl/lib/LWP.pm>
- Getopt::Long and File::Find, usually available by default on any Perl distribution.

8.5 Sample Response from ctasd

The following is an excerpt of a typical ctasd response to http_client.pl sent to stdout:

```
----- File: /home/ctasd/corpus/c9mailgw11/00000E44DF
200 OK
X-CTCH-PVer: 0000001
X-CTCH-Spam: Bulk
X-CTCH-VOD: Unknown
X-CTCH-Flags: 0
X-CTCH-RefID: str=0001.0A090209.43DF250A.000E,ss=3,sh,fgs=0
X-CTCH-SenderID: as defined in your test
X-CTCH-SenderID-Flags: value depending if a SenderID spam threshold
has been crossed
----- File: /home/ ctasd/corpus/c9mailgw11/00000E44E0
200 OK
X-CTCH-PVer: 0000001
X-CTCH-Spam: Bulk
X-CTCH-VOD: Unknown
X-CTCH-Flags: 0
```

```
X-CTCH-RefID: str=0001.0A090206.43DF2506.0031,ss=3,sh,fgs=0
X-CTCH-SenderID: as defined in your test
X-CTCH-SenderID-Flags: value depending if the SenderID spam
threshold has been crossed
----- File: /home/ ctasd/corpus/c9mailgw11/00000E44E1
200 OK
X-CTCH-PVer: 0000001
X-CTCH-Spam: Bulk
X-CTCH-VOD: Unknown
X-CTCH-Flags: 0
X-CTCH-RefID: str=0001.0A090207.43DF25A8.0044,ss=3,sh,fgs=0
X-CTCH-SenderID: as defined in your test
X-CTCH-SenderID-Flags: value depending if a SenderID spam threshold
has been crossed
```

9 Index

Acceptance Test	66	Virus Outbreak Detection.....	62
Accuracy Level Values	10	CountersMask	28
Antivirus	8	Crossing Notifications	50
Asynchronous mode	11	ctasd.conf	16
AsyncResolverRequests	22	Deployment Options	14
AVDePath	26	Detection Accuracy	8, 10
AVScanMode.....	27	Detection Type Values	8
AVWaitForUpdatedDefFiles	27	Error Handling	49
BindingAddress	25, 26	Exim	11
Blue list	6, 45	Exim Response Sample.....	56
BulkCounter	28	Failover Mechanism	64
BulkScore	25	GetServices Request.....	48
BulkThreshold	30	GetStatus Request.....	48
Cache_records	23	HighScore	25
CacheMaxEntries	29	HTTP Protocol Counters	60
Classifications		HTTP Request	
Spam	6	with Streaming	42
Virus	7	http_client.pl	68
ClassifyMessage_File	34	HttpServer	24
ClassifyMessage_Inline	41	InitialThreads.....	24, 26
Command Antivirus	8	Internal Directory Structure	15
Command Line Options.....	32	IP address	
Concurrency	25, 26	mask	29
Configuration	16	License_key_code	16, 23
ConfirmedCounter	28	ListenBacklog.....	24, 26
ConfirmedScore	25	MaxThreads.....	24, 26
ConfirmedThreshold	30	MediumScore	25
Connectivity	23	Minimum Requirements	14
Connectivity Test	65	NonSpamScore	25
Corpus for Evaluation	66	Outbound configuration parameters	27
Counters		Outbound Sender Threshold Counters	61
HTTP Protocol	60	Outbound Spam Classification Counters.....	61
Outbound Spam	61	OutboundEnabled	22
Spam Classification	59, 61	Package contents	14

PersistentCacheEnabled	22	SpamThreshold	26
Protocol.....	34	Stats.....	27
Proxy configuration parameters	31	StatusPort.....	27
Querying Device.....	4	Stopping ctasd.....	33
ReceiveTimeout	26	SuspectedCounter	28
RecipientsCounter.....	28	SuspectedScore	25
RecipientThreshold	31	SuspectedThreshold.....	30
ReportFN Request.....	47	Synchronous mode.....	11
ReportFP Request	46	System Architecture.....	5
Request Conventions	34	TempFolder	26
Requirements.....	14	Testing.....	65
Response Conventions.....	34	Testing ctasd	65
Running ctasd.....	32	Threshold Configuration	30
Sample Client	68	Threshold Crossing Notifications	50
Sample Configuration File.....	16	TotalMessagesCounter.....	28
Sample HTTP Error	49	TotalThreshold	30
Score-based engine.....	25	UpdateSenderIdList Request	45
Security	23	UpdateSenderIdList Response	45
SenderId	50	UseAuthMode	22
SenderIdBlueListFile	29	Virus Classifications.....	7
SenderIdHeaderFormat.....	29	Virus Outbreak Detection.....	7
SenderIdHeaderName	29	Virus Outbreak Detection Counters	62
SenderIdIgnoreList	29	VirusCounter	28
SenderIdReportingInterval	29	VirusScore	25
SenderIdWindows	28	VirusThreshold	30
SenderIdWindowSize.....	28	vseDetectionAccuracyExact	11
Server_address	23	vseDetectionAccuracyHeuristic.....	11
SNMP Counters.....	58	vseDetectionAccuracyLow	11
socket_client.pl.....	68	vseDetectionAccuracyNewOrModified.....	11
Spam Classification Counters.....	59	vseDetectionAccuracyNone	11
Spam Classifications.....	6	vseScanResultFound.....	8
SpamAssassin.....	11, 25	vseScanResultNone	8
Headers description.....	53	White list	6, 45
Network Protocol.....	11, 52	X-CTCH-FileName	35
Workflow	13	X-CTCH-Flags	37
spamc command.....	54	X-CTCH-ListCommand	45
response.....	55	X-CTCH-MailFrom.....	35
spamc Commands.....	53	X-CTCH-RcptCount	35
SpamCounter	28	X-CTCH-RefID.....	37
Spamd	25	X-CTCH-SenderID.....	35, 37
SpamServerEnabled	22	X-CTCH-SenderID-Flags	38

X-CTCH-SenderID-TotalBulk	38	X-CTCH-SenderIP	35
X-CTCH-SenderID-TotalConfirmed	38	X-CTCH-Service	46
X-CTCH-SenderID-TotalMessages	38	X-CTCH-VOD	37
X-CTCH-SenderID-TotalRecipients	38	X-Spam-Flag.....	57
X-CTCH-SenderID-TotalSpam	38	X-Spam-Report	57
X-CTCH-SenderID-TotalSuspected	38	X-Spam-Score	57
X-CTCH-SenderID-TotalVirus.....	38	X-Spam-Status	57