CYREN Inbound Anti-Spam Daemon (ctasd) Integration Manual

Contacts

Any technical questions you or your developers have about using the ctwsd 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 support@cyren.com.

Trademark and Copyright Statement

Table of Contents

1 Introduction

Every massive email-borne outbreak, spam or malware, can be identified by one or more recurrent message patterns. Even when spammers intentionally compose messages within the attack differently in an attempt to evade lexical analysis-based filters, patterns still exist to enable these messages to be identified as spam. CYREN RPD™ technology detects these patterns as it analyzes data gathered 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 filtering to your mail delivery system in order to provide real-time classification for inbound messages, already in the first minutes after a new outbreak is launched.

With ctasd, CYREN enables OEM Partners and Service Providers to integrate both anti-spam service and antivirus service to protect their end users from viruses and spam attacks. The Virus Outbreak Detection service (VOD), also enables you to protect against new viruses, as they are released to the Internet, even before antivirus vendors have released signatures. While this Zero-Hour Protection service can identify new virus attacks, it was not designed to maintain a repository of virus definitions. CYREN's Command Antivirus Service is responsible for identifying recognized viruses in incoming messages or viruses that have infected files within specified directories within your network.

ctasd is a flexible solution that can be tailored to the needs and preferences of CYREN's OEM partners and Service Providers. It can be configured to offer some or all of the following services:

- Anti-Spam service – classifies standard rfc822-compliant email messages for the likelihood of being Spam messages. Options for classifications are: NonSpam, Confirmed, Bulk, Suspect, Unknown, and ValidBulk.

- Zero-hour Virus Protection service – also known as Virus Outbreak Detection (VOD), this service identifies the likelihood that a filtered message contains email-borne malware. Options for threat levels are: NonVirus, Virus, High Risk, Medium Risk and Unknown.
- CYREN's Antivirus Protection service – delivers superior, efficient detection with a small footprint, appropriate for integration into a wide variety of products or services. It blocks malware of all types, including worms, Trojans and spyware. The Antivirus Protection service can be used to scan objects such as files or email messages.

When integrated as a total email security service, CYREN provides comprehensive spam and virus protection, identifying both existing and known malware threats as well as those just emerging but as yet unidentified viruses.

For those customers who implement a dual-engine anti-spam strategy, CYREN's LocalView engine introduces a SpamAssassin-like scoring and threshold system that provides additional detection functionality. The LocalView combines each customer's Black and White lists and scoring preferences, CYREN-defined rules, and Custom Rules defined and managed by the CYREN customer, to provide maximum accuracy and detection.

1.1 Working with ctasd

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

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

ctasd includes an HTTP server listening to HTTP requests on a predefined port. When it receives input about messages requiring filtering, it passes the data to an embedded Detection Engine (ctengine™). The message patterns are analyzed and using this analysis, a spam and/or virus outbreak detection classification is determined. ctasd is then responsible for responding to the HTTP client with the specific classifications of the messages.

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-querying device.

The querying device is responsible for passing information about the messages to ctasd. It will then receive classifications from ctasd and use these classifications to apply policies or actions to the messages (for example delete the message because it was found to be spam, quarantine the bulk message until more information is available, or forward the message to the intended recipient).

The querying device connects to a predefined TCP port on the ctasd daemon and passes information about the messages that require filtering 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).

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 and filter email-borne spam and malware outbreaks 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 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.

1.2 ctasd Solution Components

The overall ctasd solution involves the following components:

- CYREN Datacenter
- Querying devices or Mail Servers
- ctasd daemon
- ctasd protocol

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 or Service Provider Mail Servers 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.

ctasd

The CYREN Advanced Security daemon (ctasd) performs various functions to provide the most advanced anti-spam, antivirus protection available. It is responsible for receiving and processing incoming queries from query devices to determining the spam and/or virus classifications of incoming messages. Responses are generated back to the querying device with the information required to protect end-users from existing and emerging threats.

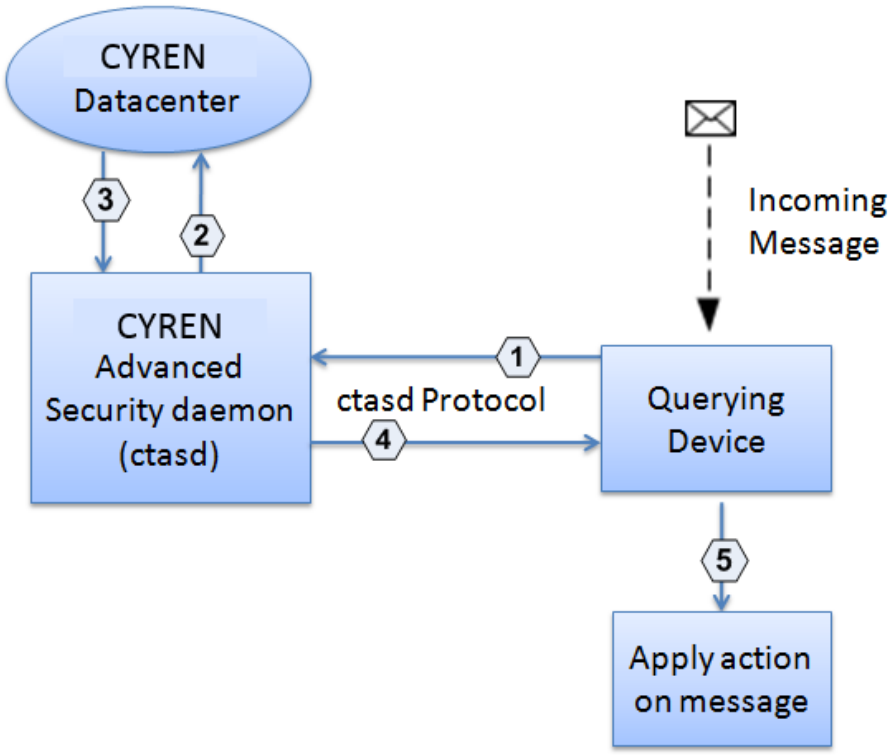
ctasd Protocol

In order to enable communication between a querying device and ctasd, CYREN has developed a simple protocol for its OEM partners using ctasd. This enables 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.

Note: *ctasd also supports a SpamAssassin-compatible protocol, referred to as Spamd. The Spamd protocol should be used when a Mail Server has built-in integration to SpamAssassin. In these cases, it may be easier to integrate with ctasd using the spamd protocol. The Exim Mail Server has been approved to run with the ctasd spamd protocol. See SpamAssassin Network Protocol.*

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 incoming 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.
2. ctasd prepares and forwards a query to the 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.
5. The querying device, 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.).

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.

1.4 Synchronous vs. Asynchronous Communication

ctasd communicates with the Resolver servers located in the CYREN Datacenter to pass queries for classification results and to receive responses. In Synchronous communication mode, a thread is allocated per classification request sent to a Resolver. The querying device initiates a query and then waits for the response. In this mode, higher memory resources are required.

In Asynchronous communication mode, a single thread can handle multiple classify queries to the Resolver and requires less resources To ensure backwards compatibility, ctasd continues to work, by default, in Synchronous 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.*

1.5 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 daemon (spamd) that communicates with client email system (e.g. Exim). 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 integration benefits:

- 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.

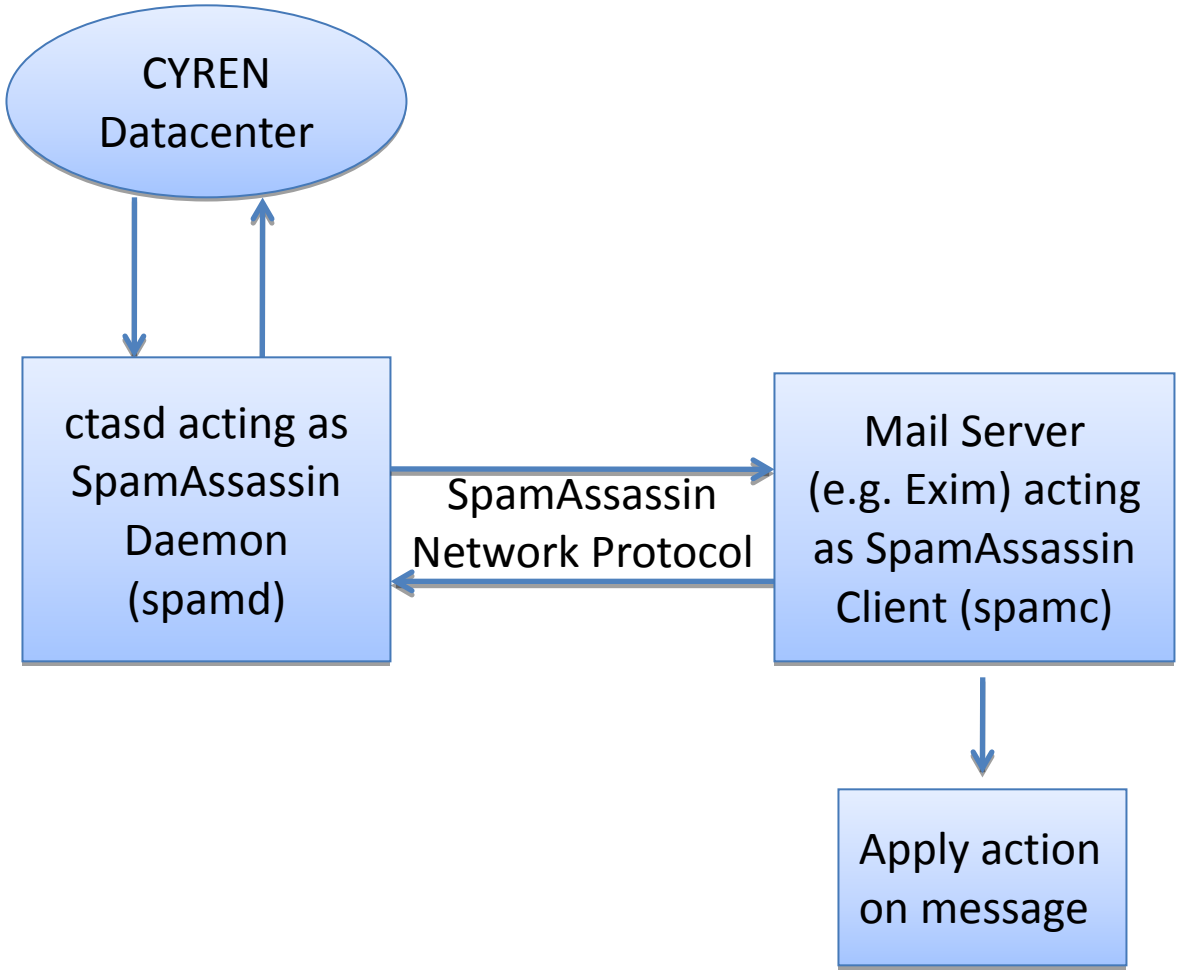
- Cross platform support works with virtually any SpamAssassin environment.
- In-the-cloud architecture that scales to the needs of the most demanding environment.

1.5.1.1 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.

1.6 SpamAssassin-Compatible Network Protocol Workflow

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



- An incoming 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.

1.7 Spam Classifications

The following table describes the possible Spam classifications that ctasd can return in response to a `ClassifyMessage_File/Inline` request from the querying devices. Additionally, this table outlines some guidelines for how to handle messages of each type. (X-CTCH-Spam)

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.
ValidBulk	Messages that are determined by the Datacenter to be valid bulk (e.g. solicited bulk messages such as newsletters).

1.7.1 Valid Bulk Classification

ctasd includes a Mail Sort classification that enables better inbox management by distinguishing personal emails from valid, general mailings such as newsletters Bulk messages are those that are sent out in large quantities and are often unsolicited and considered spam. By contrast, some bulk messages are valid and

wanted by recipients (for example, newsletters). If you enable the ValidBulkEnabled in the ctasd.conf file, ctasd can differentiate between these two types of bulk messages.

Note: *If you enable this option, you should verify that pre-existing policy definitions are still valid.*

By default, the Valid Bulk classification is not enabled. To enable this classification, you must uncomment the ValidBulkEnabled parameter in the ctasd.conf and change the value to 1 (enabled). The default value is 0 (disabled).

1.8 Virus Outbreak Detection Threat Level Classifications (X-CTCH-VOD)

Because CYREN's Virus Outbreak Detection services (VOD™) are designed to detect new virus outbreaks. When ctasd finds enough evidence to suggest the likelihood that an emerging virus is present, it is often recommended that you hold the message until the next relevant Command or other antivirus update instead of immediately deleting it (to avoid cases of false positives) or forwarding to the targeted recipients (to avoid cases of false negatives).

Holding the message until the next immediate antivirus update might not always be the best tactic to use, if the antivirus vendor has not had an opportunity to release the appropriate signature. Therefore, it is recommended that you determine the average response time for detecting new virus outbreaks for the particular antivirus software in use. You can then calculate how long to hold the message before again passing it to the antivirus software.

The following table describes the possible VOD classifications ctasd can return for a query:

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

1.9 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 Definition 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 Antivirus Protection classifications ctasd can return for a query:

Classification	Explanation
vseScanResultNone	Nothing was found.
vseScanResultFound	An object of concern was detected. In this case, the following information will also be returned: • Detection Type (see Command Antivirus Detection Types) • Detection Accuracy (see Detection Accuracy Values) • Virus Name

1.10 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.

Header	Explanation
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.
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.

Header	Explanation
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 is 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 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

Header	Explanation
	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.

1.11 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.

1.12 ctasd Deployment Options

ctasd can integrate with a wide variety of applications and devices to enable spam, antivirus, and/or Zero-Hour Virus Protection detection services. Each deployment option is adaptable to the individual requirements and infrastructure of the CYREN OEM partner and its customers. 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 OEM partner's 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.

1.13 ctasd Resource Requirements

Following is a list of recommended hardware requirements:

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

1.14 ctasd Package Contents

The ctasd package contains the following:

- ctasd daemon and associated binary
- ctasd documentation
- Sample files for quick evaluation and testing
- SNMP script for counters
- OS binaries for compatibility with multiple platforms

1.15 Internal Directory Structure

Depending on which platform your application uses, you may need to download and use different packages of ctasd. Following is a list of the internal directories for a standard installation:

```
./ctasd-<version>-<platform>-<compiler>/
```

`ReleaseNotes.txt`

`/bin`	Binary files, the configuration file, Command SDK engine + SDK files, and definition files
`/bin/oslib`	Operating system libraries
`/bin/snmp`	SNMP files, including sample scripts for evaluating and testing
`/bin/snmp/mibs`	SNMP MIB files
`/corpus`	Files used for testing ctasd integration
`/docs`	ctasd documentation
`/samples`	Sample scripts for evaluation and testing

1.15.1 RPM Structure

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

`/usr/lib/ctasd`	Contains all the binaries in the same structure as in the bin directory (above) for the ctasd package
`/usr/lib/ctasd/snmp`	Contains all SNMP counters scripts
`/etc/ctasd/`	Contains the 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.

While optional in the sense that ctasd will function properly without being configured, the `IP_ignore_list` parameter is important to set. The IP ignore list contains a list of all IP addresses of mail servers within the organization. It is important for ctasd to be able to identify these as trusted sources in order to more easily distinguish from external suspected IP addresses. The list should only contain internal mail servers and it is a good practice to modify it as changes occur in the messaging environment (e.g., adding/removing mail servers, modifying addresses to existing servers, etc.). Where an entry from the IP Ignore list is found in either the Whitelist or the Blacklist, the Whitelist/Blacklist entry is ignored.

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 = xxxxxxxxxxxxxxxxxxx
Server_address = xxxxxxxxxxxxxxxxxxx

#   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
#   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 paramter 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 LocaView 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 stopped. 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 use the Mail Sort feature to return classifications of Valid Bulk. While Bulk is considered unsolicited messages that are received in large quantities within a given period of time, and therefore suspected of spam. Valid Bulk is defined as solicited or wanted bulk messages such as newsletters. The determination is made by the Datacenter. Default value: 0 (disabled).

If this parameter is not enabled, no differentiation will be made between Unknown and Valid Bulk; all Valid Bulk messages will receive the Unknown classification.

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 (enabled). Default value: 0.

For improved performance, it is recommended that you work in Asynchronous mode. 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.

IP_ignore_list = <IP address:mask>, <IP address:mask>…

The IP Ignore list contains a list of IP addresses of all local mail servers that should automatically be considered non-spammers and therefore, should not be reported to the Datacenter. When checking the servers from which the suspected message originated, ctasd ignores all references to local or remote mail servers predefined in the IP ignore list section of the `ctasd.conf` file. Where an entry from the IP Ignore list is found in either the Whitelist or the Blacklist, the Whitelist/Blacklist entry is ignored. The IP Ignore List parameter supports IP addresses in only IPv4 IP addresses, either as a single IP address or as an IP mask definition.

Example: 1.2.3.0:255.255.255.0

SpamdServerEnabled

The SpamdServerEnabled 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. Default value: 0 (disabled).

Note: *If Outbound Spam is enabled, a unique license key for this service should be used. For more information, see the ctasd Outbound Spam Protection Integration Manual.*

2.1.2 [Connectivity]

License_key_code = <license key code>

Enter the license key code for ctasd. If an incorrect number is entered, 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 OEM/Service Provider partner.
- **OEM token:** A unique identifier (up to 35 alphanumeric characters) provided by the OEM/Service Provider partner.

The OEM/Service Provider partner's 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 OEM 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 OEM 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.

Note: *If you connect to the Internet through a proxy server, you will need to set and enable the Proxy server options. For more information, see Optional Configuration Parameters.*

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.

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, than ctasd will monitor traffic on all available IP addresses.

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, than ctasd will monitor traffic on all available IP addresses.

2.1.6 [AV]

AVDefPath

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/ sub-directory, where the host application has Read/Write permissions.

To enable the Antivirus 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 to download the latest 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

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, than 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 an Outbound Spam Protection license key from CYREN. For more information about the CYREN Outbound Spam Protection Service, see the *ctasd Outbound Protection Service Integration Manual*.

2.1.9 [LocalView]

Note: *If LocalView services are provisioned in your license key, the following parameters should be enabled and modified, if needed.*

CustomRulesFilePath

Defines the directory where the Custom Rule files should be stored. For more information on LocalView Custom Rules, see the *LocalView Rules Management* document. It is recommended that you create a separate folder in which to store the custom rule files. If LocalView is enabled, the CustomRulesFilePath is a mandatory parameter that requires configuring.

Note: *The System-Wide LocalView Custom Rules must be defined in a single file with the name SWCustomRules.txt.*

LocalView_BulkThreshold

Defines the LocalView threshold for classifying messages as bulk.

Default value: 5

LocalView_ConfirmedThreshold

Defines the LocalView threshold for classifying messages as confirmed spam.

Default value: 10

ShortCircuitEnabled

Enables or disables the LocalView short-circuit functionality that determines whether a scan will be completed, or stopped after the threshold is reached.

Default value: 0

ShortCircuitThreshold

If enabled, the short-circuit threshold defines when the scanning process will be stopped. It is recommended that the short-circuit value be larger than the spam threshold value.

Default value: 100

CTIPRepRBL_Tags

LocalView leverages CYREN's IP reputation data to add/remove message scores based on the Sender IP of the message. The SenderIP may receive one of the following IP reputation classifications rule tags:

- IP-Black
- IP-Dark-Grey
- IP-Grey
- IP-White

- IP-Very-White

The default score for each one of the IP classifications is:

- IP-Black=5
- IP-Dark-Grey=4
- IP-Grey=3.5
- IP-White= -0.3
- IP-Very-White= -0.6

A customer can change the default IP classification scores as detailed above, by updating the score values in the parameter CTIPRepRBL_Tags.

WBLHeaderListFrom

Defines the headers used to match the white_from and black_from lists.

Following headers checked: From=Envelope-Sender, Resent-Sender, X-Envelope-From, From, list-unsubscribe, Sender, Mail-From

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

2.2 Optional Configuration Parameters

The following parameters are optional. They do not appear in the configuration file unless manually added by the IT administrator.

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 = <integer>

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.

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)

-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.

-i

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 OEM partner 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 inbound messages from the messaging network and for each message it generates and posts a request to ctasd to classify the message.

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.

Note: For the ctasd protocol for SpamAssassin-compatible networks, see

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.*

4.2.1.1 ClassifyMessage_File Headers

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 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 represents the actual SMTP machine that was used by the spammer, this is unfortunately not possible because 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. ***Note:*** *All CYREN products support IPv6-format for handling IP addresses. ctasd supports SenderIP blocking of IPv6 addresses at the specific IP level and at defined IPV6 subnets. ctasd can receive and process*

	queries containing IP addresses in IPv6 format (as well as IPv4).
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-FileName	The `path` to the message file requires filtering. Verify that ctasd has access permission to the files.

4.2.1.2 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-FileName: /var/spool/incoming/00000E44E1.eml

4.2.1.3 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.

Header	Explanation
X-CTCH-Spam	Returns the final classification of the message, taking into consideration all known factors (RPD results, LocalView CYREN Rules, LocalView System-wide Custom Rules, LocalView Local Custom Rules, etc.). Results will be in the form of X-CTCH-Spam: <Classification>. Classification options are: `Confirmed`, `Bulk`, `Suspect`, `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-Score	Returns the final score of the message, taking into consideration all known factors (CYREN score values, Customer score values, etc.). Results will be in the form: X-CTCH-Score: 4.4
X-CTCH-ScoreCust	Returns the final score of the message by factoring in only LocalView Custom Rules. Results will be in the form of X-CTCH-ScoreCust: 1.4
X-CTCH-Rules	Details the list of caught rules. Results will be in the form of X-CTCT-Rules: SCG_Rule

4.2.2 ClassifyMessage_File Response

4.2.2.1 Sample ClassifyMessage_File Response

HTTP

HTTP/1.0 200 OK
Date: Sat, 12 May 2006 22:25:21 GMT
Server: 165.34.21.87
Content-Length: 122
Connection: close

Content-Type: text/plain

Response Envelope

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

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.

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

HTTP

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

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

X-CTCH-PVer:0000001
X-CTCH-SenderIP: 150.215.71.29
X-CTCH-MailFrom: uxg4pbb6y@yahoo.com

POST data msg

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

--B6B_273_5877FA._0FCA._7

msg . body Content-Type: text/html;

.

.

.

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

4.4 Response to ClassifyMessage_Inline 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	Returns the final classification of the message, taking into consideration all known factors (RPD results, LocalView CYREN Rules, LocalView System-wide Custom Rules, LocalView Local Custom Rules, etc.). Results will be in the form of X-CTCH-Spam: <Classification>. Classification options are: `Confirmed`, `Bulk`, `Suspect`, `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.

Header	Explanation
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.

4.5 ClassifyMessage Response for Antivirus Protection

Note: *The following response is only applicable when the Command Antivirus Protection service is enabled.*

The ClassifyMessage request is the same, regardless of which service has been enabled. The response, however, varies. When ctasd's email and/or VOD solution is combining with its Command Antivirus Protection service, the following additional parameters will appear in the ClassifyMessage response.

4.5.1 Response Envelope per Detected Threat

The result of the scan varies, depending on what was found. The possible options are detailed below.

4.5.1.1 When No Threat was Found

Header	Explanation
X-CTCH-AV-ThreatsCount =0	Indicates no threats were found.

4.5.1.2 When a Threat was Detected

Header	Explanation
X-CTCH-AV-ScanResult:Infected	X-CTCH-AV-DetectionType X-CTCH-AV-DetectionAccuracy X-CTCH-AV-DetectionName X-CTCH-AV-ThreatsCount=1 where 1 represents the number of threats found.

X-CTCH-AV-DetectionType — Details the detection type found during the scanning of the message. See *Detection Type Values* for a list of possible results.

X-CTCH-AV-DetectionAccuracy	The detection accuracy determined by ctasd. See *Detection Accuracy Level Values* for a list of possible results.
X-CTCH-AV-DetectionName	The name of the virus that was detected in the scanned message.

4.5.1.3 If an Error Occurs

Header	Explanation
X-CTCH-AV-ScanResult	X-CTCH-AVScanResult:Error X-CTCH-AVThreatsCount:0 or 1…

4.5.2 ClassifyObject

A new request for ctasd's unified antivirus classification is made using ClassifyObject, which enables the Command AV engine to scan files and return virus classifications. The ClassifyObject request has two modes:

- ClassifyObject_inline
- ClassifyObject_File

4.5.2.1 ClassifyObject Request Format

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-FileName	Specifies the file path to scan for viruses.

4.5.2.2 ClassifyObject Response Format

Header	Explanation
X-CTCH-AV-ThreatsCount	A counter for viruses found in the request

4.5.2.3 Response Envelope per Threat

Header	Explanation
X-CTCH-AV-ScanResult	The result of the scan. Possible options include: • *vseScanResultNone* Nothing was found. • *vseScanResultFound* An object of concern was detected. • *vseScanResultError* An error occurred.
X-CTCH-AV-DetectionType	Details the detection type found during the scanning of the object. See *Detection Type Values* for a list of possible results.
X-CTCH-AV-DetectionAccuracy	The detection accuracy determined by ctasd. See *Detection Accuracy Level Values* for a list of possible results.
X-CTCH-AV-DetectionName	The name of the virus that was detected in the scanned object.

4.5.2.4 Response Envelope per Scanned Object

Header	Explanation
X-CTCH-FileName	The name of the scanned file.

4.6 ClassifyMessage Response when LocalView is Enabled

Header	Explanation
X-CTCH-SpamCust	Returns the spam classification when only LocalView Custom rules are factored into the classification results.
X-CTCH-Score	Returns the final score of the message, taking into consideration all known factors (CYREN score values, Customer score values, etc.). Results will be in the form: X-CTCH-Score: 4.4

Header	Explanation
X-CTCH-ScoreCust	Returns the final score of the message by factoring in only LocalView Custom Rules. Results will be in the form of X-CTCH-ScoreCust: 1.4
X-CTCH-Rules	Details the list of caught rules. Results will be in the form of X-CTCT-Rules: SCG_Rule

4.6.1.1 Example of Response with LocalView Services

```
Real example with all headers:
X-CTCH-PVer: 0000001
X-CTCH-Spam: Bulk
X-CTCH-VOD: Medium
X-CTCH-Flags: 12
X-CTCH-RefID:
str=00D6.C0A843D6.4E5355CE.0001,ss=1,re=4.013,recr=4.013,recu=0.000,reip=0.000
,vtr=str,vl=1,cl=2,cld=2,fgs=12
X-CTCH-SpamCust: Suspect
X-CTCH-Score: 0.000
X-CTCH-ScoreCust: 0.000
X-CTCH-Rules: C_4166,C_4205,C_61
```

4.7 Method: ValidateCustomRules

ValidateCustomRules is used to validate and report errors on LocalView Custom Rules files. The ValidateCustomRules parameter does not load the custom rules, rather it checks to confirm that the syntax of the defined rule is valid and can be applied by the LocalView engine. It does not check the logic of the content of the files.

During the validation process, an error message will be generated if any rules are found to be invalid, and a list of which rule expressions could not be validated is included in the error message. The ValidateCustomRules parameter is used to check all files within a specified directory. This can include both System-wide custom rules, Local custom rules.

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.

Header	Explanation
X-CTCH-Path	Specifies the path where the local files are stored. ctasd locates the file containing the local rules and validates them. If there are no errors, no response is returned.

4.7.1 Response to Error in ValidateCustomRules Process

Header	Explanation
X-CTCH-Error	Indicates ctasd was unable to validate some (or all) of the Custom rules contained in the files in the specified path. In this case, a list of which rule expressions could not be validated will be included in the error message.

4.8 Method: GetRulesList

GetRulesList is used to display the rules that are loaded to ctasd. The set includes: CYREN CT Rules, System-wide and Local Custom Rules, Blacklists and Whitelists. When used, this parameter will result in the Rule Tag name being displayed for each rule.

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.

4.8.1 Response to GetRulesList

Header	Explanation
X-CTCH-RulesList	A list of all the Rules tag names that have been loaded to ctasd.

4.9 Method: GetCTRuleDefinition

GetCTRuleDefinition enables the user to input the CYREN rule tag and get a definition of the specified rule.

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-RuleTag	Specify a rule expression for a CYREN rule

4.9.1Response to GetCTRuleDefinition

Header	Explanation
X-CTCH-Rule	The definition of the specified CYREN rule.

4.10 Method: GetCTRulesDefinitionList

GetCTRulesDefinitionList is used to display the CYREN rules that are loaded to ctasd. When used, this parameter will result in the rule definition of each rule.

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.

4.10.1Response to GetCTRulesDefinitionList

Header	Explanation
X-CTCH-Rules	A list of the CT rule definition that are loaded in ctasd.

4.11 Method: ReportFP Request

ReportFP is used to report a case of false positive back to ctasd. False positives are non-spam or non-malware messages which were incorrectly classified as spam or malware. 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 `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 and 2= svcVOD service. Based on the service, the message is automatically routed to different analyzing procedures at CYREN.

4.11.1.1 Request Body

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

4.11.2 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 `protocol version`. The protocol of this version is 0000001.

4.12 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 and 2= `svcVOD` service. Based on the service, the message is automatically routed to different analyzing procedures at CYREN.

4.12.1.1 Request Body

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

4.12.2 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.

4.13 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.

4.13.1 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 and 3 = both.

4.14 Method: GetStatus Request

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

4.14.1 Response to GetStatus Request

ctasd will respond with the protocol version and the status is returned in the http status header as either ‘ http ok’ or some http error specifying 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:

- 4xx - HTTP bad request: invalid request format
- 5xx - HTTP internal server error:

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 Error Body

The body will include free text describing the error.

4.15.2 Sample HTTP Error

```
HTTP/1.0 400 HTTP bad request: invalid request format
```

4.15.2.1 HTTP headers

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

4.15.2.2 Response Envelope

```
X-CTCH-PVer: 0000001
X-CTCH-ERROR: 501
```

4.15.2.3 Error body

```
Your license key is invalid or blocked in the CYREN Datacenter.
Contact CYREN technical support.
```

5 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.

5.1.1 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 sysexits.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	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:
	`SPAMD/1.1 0 EX_OK\r\n`
	`Spam: True ; 15 / 5\r\n`
	`\r\n`
	`This mail is probably spam.`
	`[...]`
	Note: The Report Command is used by the Exim Mail Server.

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	The recipient's count of the message. This header is optional. *Note: The X-CTCH-RcptCount header is applicable only to an Outbound solution.*

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
X-CTCH-SenderID: bary008@ctxx.com

POST data Received: FROM [218.5.5.228] By c9diamond03.diamond.amadis.com;
Sun, 22 Jun 2003 05:28:32 -0800

msg. headers

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 Inbound Spam Detection Exim Response Sample

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

```
Return-path: <r@test.com>
Envelope-to: bob@xyx.com
Delivery-date: Sun, 08 Aug 2010 19:04:18 +0300
Received: from [172.20.20.6] (helo=test.com)
        by xyz.com with smtp (Exim 4.69)
        (envelope-from <r@test.com>)
        id 1Oi8Lu-0001Hy-DC
        for bob@xyz.com; Sun, 08 Aug 2010 19:04:18 +0300
X-Spam-Status: Yes, score=10.0
X-Spam-Score: 100
X-Spam-Bar: ++++++++++
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,fgs=32
X-Spam-Flag: YES
Subject: ***SPAM*** confirm spam classification test
```

5.7 Exim Implementation of 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,fgs=32
X-Spam-Flag: YES
```

5.7.1 Inbound Example

```
Envelope-to: name@email.com
Delivery-date: Wed, 03 Aug 2011 06:19:47 +0000
from: jhfgjhfj@hvgjhgj.com
to: jhyygjhgj@jghjgh.njhvg
Subject: jhgjhg req31_7
Date: Wed, 03 Aug 2011 06:19:13 +0000
X-Spam-Score: 100.0
(++++++++++++++++++++++++++++++++++++++++++++++++++)
X-Spam-Report: X-CTCH-Spam: Confirmed
        X-CTCH-VOD: Unknown
        X-CTCH-Flags: 0
        X-CTCH-RefID:
str=00FB.C0A843FB.4E38E6A8.0001,ss=1,re=0.000,recr=0.000,recu=0.000,re
ip=0.000,pt=F_7511688,cl=4,cld=1,fgs=0
        X-CTCH-SenderID: jhfgjhfj@hvgjhgj.com
        X-CTCH-SenderID-Flags: 0
Subject: *SPAM* jhgjhg req31_7
```

The following describes the spam classification related headers:

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 [general] 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 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	The total number of seconds elapsed since the daemon was started.
totalCenterRequests	The total number of queries forwarded to the Datacenter.
totalCommErrors	The total number of communication errors that occurred while trying to query the Datacenter.
totalCenterRequestTime	The total amount of time the client waited for a response from the Datacenter.
asapTotalClassifyMessageRequests	The total number of ClassifyMessage queries.
asapTotalClassifyMessageCenterRequests	The total number of ClassifyMessage queries sent to the Datacenter.
asapTotalClassifyMessageErrors	The total number of general errors in ClassifyMessage queries.
asapTotalClassifyMessageCurrRequests	The total number of ClassifyMessage queries currently being processed.
asapTotalGetStatusRequests	Total number of Get Status queries.

Counter Name	Description
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 Name	Description
asapTotalSuspected	Total number of messages classified as suspected.
asapTotalBulk	Total number of messages classified as bulk.
asapTotalConfirmed	Total number of messages classified as confirmed.
asapTotalNonSpam	Total number of messages classified as non-spam.
asapTotalUnknown	Total number of messages which were classified as Unknown.
asapTotalValidBulk	Total number of valid bulk messages. This classification counter will appear only if the Valid Bulk feature is enabled.

6.3 Virus Outbreak Detection (VOD) Classification Counters

Counter Name	Description
asapTotalVodMedium	Total number of messages classified as VOD Medium.
asapTotalVodHigh	Total number of messages classified as VOD High.
asapTotalVodVirus	Total number of messages classified as VOD Virus.
asapTotalVodNonVirus	Total number of messages classified as VOD Non-Virus.
asapTotalVodUnknown	Total number of messages where the VOD classification is not known.

6.4 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.

Counter Name	Description
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.

6.5 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.

6.6 Command Antivirus SNMP Counters

Counter Name	Description
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.

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 CYREN 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 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.

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 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. For Anti-Spam or VOD testing: execute either the sample code http_client.pl or socket_client.pl with reference to a directory into which you have stored test messages.
4. For Command Antivirus testing: execute the sample code http_client_av.pl reference to a directory into which you have stored test messages.
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.

8.3.1 For Spam Classification

You can use the following files to test the integration. You can use the following files to test the ctasd spam classification.

File name	Expected Result	Classification Source
msg_cs.eml	Confirmed	Datacenter
msg_bs.em	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	Unknown	Datacenter

Note: *If the option for ValidBulk is enabled, the expected results for msg_sh.eml file will be ValidBulk, rather than unknown.*

8.3.2 For Virus Outbreak Detection

You can use the following files to test the ctasd 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 eicar.zip	Confirmed virus	http://www.eicar.org/anti_virus_test_file.htm

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

8.3.3 For Command Antivirus

You can use the following files to test the ctasd Command Antivirus 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
```

8.4 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. To use the sample client, you should execute the client and ctasd from the same host.

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 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]

-s, --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 -s 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.

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
---------- 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
---------- 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
```

9 Index