Commtouch
Advanced Security Daemon

Integration Manual

(ctasd)

Trademark and Copyright Statement

For more information, visit our website: http://www.commtouch.com/

Table of Contents

Introduction **3**
Working with ctasd 3
ctasd Solution Components 5
System Architecture and Data Flow 6
ctasd Deployment Options 7
ctasd Minimum Requirements 8
ctasd Package 8
Internal Directory Structure 8

ctasd Configuration **10**
Sample Configuration File 10
Optional Configuration Parameters 15

Running ctasd **16**
Command Line Options 16
Stopping ctasd 17

ctasd Protocol **18**
Request and Response Conventions 18
Method: ClassifyMessage_File 18
Response to ClassifyMessage_File Request 20
Method: ClassifyMessage_Inline 22
Response to ClassifyMessage_Inline Request 24
Method: ReportFP Request 25
Response to ReportFP Request 25
Method: ReportFN Request 26
Response to ReportFN Request 26
Method: GetServices Request 27
Response to GetServices Request 27
Method: GetStatus Request 28
Response to GetStatus Request 28
Error Handling 28
Spam Classifications (X-CTCH-Spam) 30
Virus Threat Level Classifications (X-CTCH-VOD) 31

ctasd Reports and Logging **32**

Deploying ctasd over WAN **35**
Implementing a Failover Mechanism 35

ctasd Testing and Verification **37**
Connectivity Test 37
Acceptance Test 38
Corpus of Messages for Evaluation 38
Running the Sample Client 40

Introduction

Commtouch 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, already in the first minutes after a new outbreak is launched.

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 is 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™) and then it is 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.

The client application is responsible for passing information about the messages to ctasd. It will then receive classifications that can be translated afterwards to applied actions on the messages (i.e., delete, quarantine, forward).

The client 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 Commtouch-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 hundreds bytes to the Commtouch Datacenter. The Datacenter is responsible for warehousing a vast, constantly-expanding Spam and virus pattern repository. The Datacenter responds with classification to message patterns found in the original query. Full round-trip time for query/response is less than 300 ms, excluding Internet latency.

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

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

The Commtouch 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.

ctasd Solution Components

The overall ctasd solution involves the following components:

- Commtouch Datacenter
- ctasd daemon
- ctasd protocol
- Querying devices

Commtouch Datacenter

The Commtouch Datacenter monitors global email traffic in real-time (24*7*365) from various sources on an ongoing basis and maintains a vast database of message patterns and classifications that are determined based on numerous dynamically changing parameters.

ctasd

The Commtouch Advanced Security daemon (ctasd) performs various functions, from receiving and processing incoming queries from query devices to determining the Spam and/or virus outbreak detection (VOD) classifications of incoming messages and quickly responding to the querying devices.

ctasd Protocol

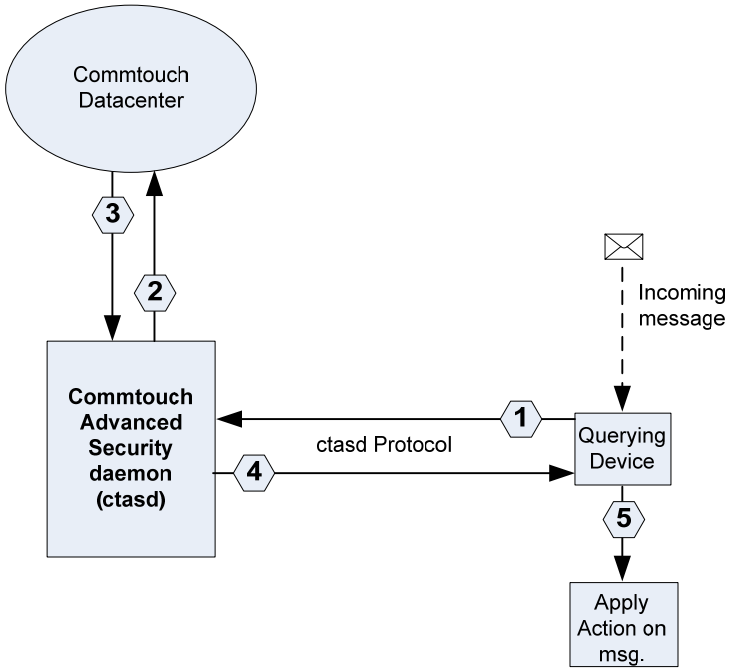
In order to enable communication between a querying device and ctasd, Commtouch has developed a simple protocol for its OEM partners using ctasd. This enables OEM partners to provide advanced anti-spam and virus outbreak detection services to their users. Communication between the ctasd and the querying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

Querying Device

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

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 Commtouch ctasd protocol to generate a query to ctasd requesting spam and VOD classifications.
2. ctasd prepares and forwards a query to a Commtouch Datacenter to retrieve the most up-to-date classification.
3. The Commtouch 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 patterns classifications received in responses from the Commtouch 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 Commtouch Datacenter. If no match is found, ctasd will then prepare and send a query to a

Commtouch 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.

ctasd Deployment Options

ctasd can integrate with a wide variety of applications and devices to enable spam and/or Zero-Hour Virus Protection detection services. Each deployment option is adaptable to the individual requirements and infrastructure of the Commtouch 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 Commtouch Datacenter. Alternatively, ctasd can be deployed over WAN and remotely from the querying devices. To learn more about this deployment option, review Deploying ctasd over WAN.

ctasd Minimum Requirements

The ctasd package includes all the necessary system dependencies and can therefore be run on any Windows, Solaris, Linux or FreeBSD platform. There is a special ctasd version for each of these platforms. 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

ctasd Package

The ctasd package contains the following:

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

Internal Directory Structure

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

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

ReleaseNotes.txt	
/bin	Binary files and the configuration file
/docs	ctasd documentation
/corpus	Files used for testing ctasd integration
/samples	Sample scripts for evaluation and testing
/osbin-kernel24	Linux binaries for Kernel 2.4
/osbin-kernel26	Linux binaries for kernel 2.6
/osbin	Operating system libraries

Although you can unpack and set up ctasd's directory structure according to your preferences, following is a list of the directory structure in which ctasd expects to find various files and components.

/usr/lib/ctasd	Stores all the binaries in the same structure as in bin directory of the ctasd package
/usr/lib/ctasd/snmp	Stores all SNMP counters scripts
/etc/ctasd/	Stores configuration file (ctasd.conf)
/etc/init.d/	Contains the ctasd_initd. To assist you in identifying the folder, you may want to rename it to ctasd.

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.).

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.

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
#                 Commtouch ctasd in order to have your
#                 changes take effect.
#
#-------------------------------------------------------------------
```

```
[General]
#    IP Ignore List for IP addresses of all local mail servers.

IP_ignore_list =
10.0.0.0:255.255.255.0,192.168.0.0:255.255.0.0,127.0.0.0:255.255.255.0,172.16.0
.0:255.240.0.0
PersistentCacheEnabled=1
UseAuthMode=0

# 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 = 80
#ProxyServerAddress = myproxy
#ProxyAuth = NoAuth
#ProxyUserName = user@proxy
#ProxyPassword = 1234
#ProxyAccess = 1

#  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 connect by the client
#     and the relevant connectivity and performance parameters.
#

[HttpServer]
Port=8088
ListenBackLog=100
InitialThreads=1
MaxThreads=50
Concurrency=50
# If BindingAddress is empty (or commented), BindingAddress is set to
INADDR_ANY.
```

```
#BindingAddress=<IP Address>

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

[General]

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.

PersistentCacheEnabled=1

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 = 0

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)

[Connectivity]

License_key_code = <license key code>

Enter the license key code for ctasd. If an incorrect number is entered, Commtouch Datacenter will be unable to authenticate the organization and therefore decline detection services. This is a mandatory field that must be configured.

Note: Multiple instances of ctasd in the same organization can use the same license key code.

Contact your Commtouch sales representative to obtain a valid key code.

Server_address = <DNS string>

The DNS string is the server address at Commtouch Datacenter. Contact your Commtouch 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

[Security]

User = <user name>

User name associated with ctasd.

Group = <group name>

Group associated with ctasd.

[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

[Stats]

StatusPort

The Port option defines the server port for the statistics information collected from the SNMP counters.

Default value: /var/run/ctasd/ctasd.stats

BindingAddress=<IP Address>

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

Default value: commented setting (monitor all IP addresses)

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

Optional Configuration Parameters

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

[Connectivity]

ProxyPort = <port number>

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

Default value: 8088

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.

Default value: Basic

ProxyUserName = <user name>

The name of an authorized user.

ProxyPassword = <password>

The password of the authorized user.

ProxyAccess = 1

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.

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).

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

-v

Prints version information and exits.

-? or –h

Displays the help screen.

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.

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.

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 Commtouch-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. Commtouch uses and requires standard rfc822 headers structure.

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.

Header	Explanation
X-CTCH-PVer	The Commtouch `protocol version`. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The Commtouch `license key`. This header is used in conjunction with the UseAuthMode=1 option in ctasd.conf and only when deploying ctasd over WAN.
X-CTCH-SenderIP	The `IP address` of the SMTP that sent the message. This header is not always available to be included in the request and therefore is not mandatory.
X-CTCH-MailFrom	The `mailfrom` of the message envelope. This header is also not always available to be included in the request and therefore is not mandatory.
X-CTCH-FileName	The `path` to the message file requires filtering. Verify that ctasd has access permission to the files.

X-CTCH-SenderIP

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

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

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

While you may want to assume that the address at the bottom of the chain in the **'Received:'** message header representing the actual SMTP machine that was used by the spammer, this is unfortunately not possible 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 Commtouch because it may expose SMTP machines that spammers use or abuse frequently (i.e. 'zombies'). This address is compared at the Commtouch Datacenter against dynamically-generated lists of both 'bad' and 'good' IP addresses and is used for a host of applications.

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

X-CTCH-MailFrom

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

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: Commtouch 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
```

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 Commtouch protocol version. The protocol of this version is 0000001.
X-CTCH-Spam	This is the Spam classification of the message for which a query was sent by ctasd to the Datacenter. Options are: Confirmed, Bulk, Suspected, Unknown and Non-Spam. 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 Non-Virus. 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 Commtouch. It is very important that you keep the RefID with the filtered message (e.g., as a X-header).

Sample ClassifyMessage_File Response

HTTP headers

```
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
```

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).

Request Envelope

Header	Explanation
X-CTCH-PVer	The Commtouch `protocol version`. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The Commtouch `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.

Request Body

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

Sample HTTP Request with Streaming

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

HTTP headers

```
POST /ctasd/ClassifyMessage_Inline HTTP/1.0
Accept-Language: en-us
Accept: */*
Content-Length: 3867
Host: 150.215.71.23
User-Agent: Commtouch HTTP Client
```

POST data request envelope

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

POST data msg. headers

```
Received: FROM [218.5.5.228] By c9diamond03.diamond.amadis.com;
Sun, 22 Jun 2003 05:28:32 -0800
Received: from 46sx.amgyw.net [150.215.71.17] by 216.163.188.55
with SMTP; Sun, 22 Jun 2003 17:21:18 +0100
Message-ID: <4b$$76w-$2d1e-lf6h4-1-0cxs7@tvu.809p8ve.1b>
From: "John Smith" <uxg4pbb6y@yahoo.com>
To: bob@company.com
Subject:confirm spam classification test
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...

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 Commtouch `protocol version`. The protocol of this version is 0000001.
X-CTCH-Spam	This is the Spam `classification` of the message for which a query was sent by ctasd to the Datacenter. Options are: Confirmed, Bulk, Suspected, Unknown and Non-Spam. 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 Non-Virus. 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 Commtouch.

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 Commtouch for analysis.

It is critical that the RefID always be included in the message sent in the request. Without the RefID, Commtouch is unable to analyze the report.

Header	Explanation
X-CTCH-PVer	The Commtouch `protocol version`. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The Commtouch `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 Commtouch.

Request Body

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

Response to ReportFP Request

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

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

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 Commtouch 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 Commtouch for analysis.

Header	Explanation
X-CTCH-PVer	The Commtouch protocol version. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The Commtouch `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 Commtouch.

Request Body

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

Response to ReportFN Request

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

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

Method: GetServices Request

The querying device can query ctasd to determine which Commtouch 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 Commtouch `protocol version`. This header is mandatory in each request. The protocol of this version is 0000001.
X-CTCH-Key	The Commtouch `license key`. This header is used in conjunction with the UseAuthMode=1 option in ctasd.conf and only when deploying ctasd over WAN.

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 Commtouch `protocol version`. The protocol of this version is 0000001.
X-CTCH-Services	When queried, Commtouch 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.

Method: GetStatus Request

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

Header	Explanation
X-CTCH-PVer	The Commtouch `protocol version`. This header is mandatory in each request. The protocol of this version is 0000001.

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

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 Commtouch `protocol version`. The protocol of this version is 0000001.
X-CTCH-Error	`Error number`.

Error Body

The body will include free text describing the error.

Sample HTTP Error

HTTP headers	HTTP/1.0 400 HTTP bad request: invalid request format Date: Sat, 12 May 2006 22:25:21 GMT Server: hostname Content-Length: 122 Connection: close Content-Type: text/plain
Response Envelope	X-CTCH-PVer: 0000001 X-CTCH-ERROR: 501
Error body	Your license key is invalid or blocked in the Commtouch Datacenter. Contact Commtouch technical support.

Spam Classifications (X-CTCH-Spam)

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

Classification	Explanation	Optional Action(s)
Confirmed-Spam	Spam messages from known spam sources (e.g. zombies).	▪ Delete on arrival with or without generating a bounce-back message to the sender. ▪ Direct to site-level quarantine for the administrator to manage.
Bulk	Spam messages from sources that are not confirmed spammers.	▪ Direct to site-level quarantine for an administrator to manage. ▪ Direct to user-level quarantine for the end-user to manage. ▪ Tag with warning and forward to intended recipients.
Suspected-Spam	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.	▪ Forward to intended recipient.
Unknown	Messages for which ctasd does not have any incriminating information, and are therefore assumed to represent legitimate correspondence.	▪ Forward to intended recipient.
Non-Spam	Messages that are confirmed, without doubt, as coming from trusted sources. This classification is very rarely used.	▪ Forward to intended recipient.

Virus Threat Level Classifications (X-CTCH-VOD)

Because Commtouch's Virus Outbreak Detection services (VOD™) are designed to detect new virus outbreaks, it is highly recommended that you send to ctasd messages after they already were scanned by your current anti-virus application.

When ctasd finds enough evidence to suggest the likelihood that a virus is present, it is often recommended that you hold the message until the next relevant anti-virus 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 anti-virus update might not always be the best tactic to use, if the anti-virus 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 anti-virus software in use. You can then calculate how long to hold the message before again passing it to the anti-virus software.

Classification	Explanation	Optional Action(s)
Virus	The message contains characteristics of confirmed malware	▪ Reject or delete the message. ▪ Peel off the malware from the message if you have the means to do this.
High	High likelihood of the message presenting a malware threat.	▪ Delete the message. ▪ Direct the message to your anti-virus quarantine (if applicable) for manual release by the administrator. ▪ Hold the message in a special queue for the next relevant anti-virus update.
Medium	Probable threat of malware in the message has been detected.	▪ Hold the message in a special queue for the next 2 or 3 relevant anti-virus updates. ▪ Forward to intended recipients.
Unknown	Threat for malware could not be determined at this time.	▪ Treat this as an email without a malware.
Non-Virus	Confirmed that message does not contain a malware.	▪ Treat this as an email without a malware.

ctasd Reports and Logging

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 Commtouch in the package.

Following is a list of SNMP counters related to communication over HTTP:

Counter	Explanation
pid	Shows the Process ID number.
uptime	The length of time of the current session.
totalEmails	The total number of `ClassifyMessage` queries successfully received and classified by ctasd.
totalRequestTime	The total amount of time required to process the requests sent to date.
totalSpamConfirm	The total number of `Confirmed-Spam` classifications returned by ctasd.
totalSpamBulk	The total number of `Bulk` classifications returned by ctasd.
totalSpamSuspected	The total number of `Suspected-Spam` classifications returned by ctasd.
totalSpamUncategorized	The total number of `Uncategorized` spam classifications returned by ctasd.
totalNonSpam	The total number of `Non-Spam` classifications returned by ctasd.

Counter	Explanation
totalVodVirus	The total number of confirmed `Virus` classifications returned by ctasd.
totalVodHigh	The total number of `High` malware risk classifications returned by ctasd.
totalVodMedium	The total number of `Medium` malware risk classifications returned by ctasd.
totalVodUncategorized	The total number of `Uncategorized` malware The total classifications returned by ctasd.
totalVodNonVirus	The total number of `Non-virus` classifications returned by ctasd.
totalEmailsUncategorized	The total number of `Uncategorized` messages processed by ctasd containing both spam and malware services.
httpCurrReq	The number of HTTP requests that ctasd is currently processing.
httpQueueSize	The number of HTTP connections waiting to be processed.
totalWaitTime	Total wait time of all requests in the queue.
httpTotalClassifyReq	The total number of `ClassifyMessage` requests which were sent to ctasd.
httpTotalClassifyErr	The total number of errors occurred as a result of `ClassifyMessage` requests which were sent to ctasd.
httpTotalClassifyTime	The total amount of processing time on `ClassifyMessage` requests which were sent to ctasd.
httpTotalReportFPReq	The total number of `ReportFP` requests which were sent to ctasd.
httpTotalReportFPErr	The total number of errors occurred as a result of `ReportFP` requests which were sent to ctasd.

Counter	Explanation
httpTotalReportFPTime	The total amount of processing time on ReportFP requests which were sent to ctasd.
httpTotalReportFNReq	The total number of ReportFN requests which were sent to ctasd.
httpTotalReportFNErr	The total number of errors occurred as a result of ReportFN requests which were sent to ctasd.
httpTotalReportFNTime	The total amount of processing time on ReportFN requests which were sent to ctasd.
httpTotalGetServicesReq	The total number of GetServices requests which were sent to ctasd.
httpTotalGetServicesErr	The total number of errors occurred as a result of GetServices requests which were sent to ctasd.
httpTotalGetServicesTime	The total amount of processing time on GetServices requests which were sent to ctasd.
httpTotalGetStatusReq	The total number of GetStatus requests which were sent to ctasd.
httpTotalGetStatusErr	The total number of errors occurred as a result of GetStatus requests which were sent to ctasd.
httpTotalGetStatusTime	The total amount of processing time on GetStatus requests which were sent to ctasd.

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.

Implementing a Failover Mechanism

ctasd is designed with an internal failover mechanism allowing it to connect to any Commtouch 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 Commtouch 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.

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 Commtouch 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 Commtouch 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.

Connectivity Test

ctasd performs an automatic connectivity test with the Commtouch 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 Commtouch daemon is still unable to connect to the Datacenter, contact your Commtouch technical support representative or email to oem.support@commtouch.com.

Acceptance Test

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

1. Modify ctasd.conf configuration file.
2. Launch the ctasd daemon.
3. Execute either the sample code **http_client.pl** or **socket_client.pl** with reference to a directory into which you have stored test messages.
4. Evaluate the messages to confirm that they were classified correctly.

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 Commtouch 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.

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-Spam	Datacenter
msg_bs.em	Bulk	Datacenter
msg_us.eml	Unknown	Datacenter
msg_ns.eml	Non-Spam	Datacenter
msg_cs_cache.eml	Confirmed-Spam	Local Spam cache
msg_cr.eml	Confirmed-Spam	Local Proactive Patterns Engine
msg_sh.eml	Legitimate mass-mailing newsletter	Datacenter

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	High risk	http://www.eicar.org/anti_virus_test_file.htm

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

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 Commtouch 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 a 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::UsarAgent 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
```

About Commtouch

Commtouch Software Ltd. (NASDAQ:CTCH) is dedicated to protecting and preserving the integrity of the world's most important communications tool -- email. Commtouch has 14 years of experience developing messaging software, and is a global developer and provider of proprietary anti-spam and Zero-Hour virus protection solutions. Using core technologies including RPD Recurrent Pattern Detection, the Commtouch Datacenter analyzes billions of email messages per month to identify new spam and malware outbreaks within minutes of their introduction into the Internet. Integrated by over 30 OEM partners, Commtouch technology protects thousands of organizations, with over 35 million users in 100 countries. Commtouch is headquartered in Netanya, Israel and has a subsidiary in Mountain View, CA. For more information, see: www.commtouch.com.

Contact Information

For questions related to the product, features and implementation questions, contact your technical account manager at Commtouch, or to oemsupport@commtouch.com.