aiospamc Documentation

Release 0.3.0

Michael Caley

Jul 04, 2017

Contents:

aiospamc is an asyncio-based library to interact with SpamAssassin's SPAMD service.

CHAPTER 1

Contents

1.1 User Guide

1.1.1 Requirements

- Python 3.5 or later is required to use the new async/await syntax provided by the asyncio library.
- SpamAssassin running as a service.

1.1.2 Install

With PIP

```
pip install aiospamc
```

With GIT

```
git clone https://github.com/mjcaley/aiospamc.git
python3 aiospamc/setup.py install
```

1.1.3 How to use aiospamc

Instantiating the *aiospamc.client.Client* class will be the primary way to interact with aiospamc.

Parameters are available to specify how to connect to the SpamAssassin SPAMD service including host, port, and whether SSL is enabled. They default to `localhost`, `783`, and SSL being disabled. Additional optional parameters are the username that requests will be sent as (no user by default) and whether to compress the request body (disabled by default).

A coroutine method is available for each type of request that can be sent to SpamAssassin.

An example using the *aiospamc.client.Client.check()* method:

```
import asyncio
import aiospamc

example_message = ('From: John Doe <jdoe@machine.example>'
                   'To: Mary Smith <mary@example.net>'
                   'Subject: Saying Hello'
                   'Date: Fri, 21 Nov 1997 09:55:06 -0600'
                   'Message-ID: <1234@local.machine.example>'
                   ''
                   'This is a message just to say hello.'
                   'So, "Hello".')

loop = asyncio.get_event_loop()
client = aiospamc.Client()
response = loop.run_until_complete(client.check(example_message))
print(response)
```

Other requests can be seen in the *aiospamc.client.Client* class.

1.1.4 Making your own requests

If a request that isn't built into aiospamc is needed a new request can be created and sent.

A new request can be made by instantiating the *aiospamc.requests.Request* class. The *aiospamc.requests.Request.verb* defines the method/verb of the request.

Standard headers or the *aiospamc.headers.XHeader* extension header is available in the *aiospamc.headers* module. Headers are managed on the request object with the methods:

- *aiospamc.requests.Request.add_header()*
- *aiospamc.requests.Request.get_header()*
- *aiospamc.requests.Request.delete_header()*

Once a request is composed, it can be sent through the *aiospamc.client.Client.send()* method as-is. The method will automatically add the *aiospamc.headers.User* and *aiospamc.headers.Compress* headers if required.

For example:

import asyncio

import aiospamc from aiospamc.requests import Request from aiospamc.headers import XHeader

example_message = ('From: John Doe <jdoe@machine.example>' 'To: Mary Smith <mary@example.net>' 'Subject: Saying Hello' 'Date: Fri, 21 Nov 1997 09:55:06 -0600' 'Message-ID: <1234@local.machine.example>' '' 'This is a message just to say hello.' 'So, "Hello".')

request = Request(verb='FAKE') fake_header1 = XHeader('fake_header', 'Fake values') request.add_header(fake_header1) request.body = example_message

loop = asyncio.get_event_loop() client = aiospamc.Client() response = loop.run_until_complete(client.send(request)) print(response)

1.1.5 Interpreting results

Responses are encapsulated in the *aiospamc.responses.Response* class. It includes the status code, headers and body.

1.2 aiospamc API Reference

1.2.1 aiospamc package

Submodules

aiospamc.client module

Contains the Client class that is used to interact with SPAMD.

class aiospamc.client.**Client**(*socket_path='/var/run/spamassassin/spamd.sock'*, *host=None*, *port=783*, *user=None*, *compress=False*, *ssl=False*, *loop=None*)

Bases: object

Client object for interacting with SPAMD.

connection
aiospamc.connections.ConnectionManager – Manager instance to open connections.

user
str – Name of the user that SPAMD will run the checks under.

compress
bool – If true, the request body will be compressed.

sleep_len
float – Length of time to wait to retry a connection in seconds.

retry_attempts
float – Number of times to retry a connection.

loop
asyncio.AbstractEventLoop – The asyncio event loop.

logger
logging.Logger – Logging instance, logs to 'aiospamc.client'

__init__(*socket_path='/var/run/spamassassin/spamd.sock'*, *host=None*, *port=783*, *user=None*, *compress=False*, *ssl=False*, *loop=None*)

Client constructor.

Parameters

- **socket_path** (str, optional) – The path to the Unix socket for the SPAMD service.
- **host** (str, optional) – Hostname or IP address of the SPAMD service, defaults to localhost.
- **port** (int, optional) – Port number for the SPAMD service, defaults to 783.
- **user** (str, optional) – Name of the user that SPAMD will run the checks under.
- **compress** (bool, optional) – If true, the request body will be compressed.
- **ssl** (bool, optional) – If true, will enable SSL/TLS for the connection.

- **loop** (asyncio.AbstractEventLoop) – The asyncio event loop.

Raises ValueError – Raised if the constructor can't tell if it's using a TCP or a Unix domain socket connection.

coroutine check(*message*)

Request the SPAMD service to check a message with a CHECK request.

The response will contain a 'Spam' header if the message is marked as spam as well as the score and threshold.

SPAMD will perform a scan on the included message. SPAMD expects an RFC 822 or RFC 2822 formatted email.

Parameters **message** (str) – A string containing the contents of the message to be scanned.

SPAMD will perform a scan on the included message. SPAMD expects an RFC 822 or RFC 2822 formatted email.

Returns The response will contain a 'Spam' header if the message is marked as spam as well as the score and threshold.

Return type *aiospamc.responses.Response*

Raises

- *aiospamc.exceptions.BadResponse* – If the response from SPAMD is ill-formed this exception will be raised.
- *aiospamc.exceptions.AIOSpamcConnectionFailed* – Raised if an error occurred when trying to connect.
- *aiospamc.exceptions.UsageException* – Error in command line usage.
- *aiospamc.exceptions.DataErrorException* – Error with data format.
- *aiospamc.exceptions.NoInputException* – Cannot open input.
- *aiospamc.exceptions.NoUserException* – Addressee unknown.
- *aiospamc.exceptions.NoHostException* – Hostname unknown.
- *aiospamc.exceptions.UnavailableException* – Service unavailable.
- *aiospamc.exceptions.InternalSoftwareException* – Internal software error.
- *aiospamc.exceptions.OSErrorException* – System error.
- *aiospamc.exceptions.OSFileException* – Operating system file missing.
- *aiospamc.exceptions.CantCreateException* – Cannot create output file.
- *aiospamc.exceptions.IOErrorException* – Input/output error.
- *aiospamc.exceptions.TemporaryFailureException* – Temporary failure, may reattempt.
- *aiospamc.exceptions.ProtocolException* – Error in the protocol.
- *aiospamc.exceptions.NoPermissionException* – Permission denied.
- *aiospamc.exceptions.ConfigException* – Error in configuration.
- *aiospamc.exceptions.TimeoutException* – Timeout during connection.

coroutine headers(*message*)

Request the SPAMD service to check a message with a HEADERS request.

Parameters **message** (str) – A string containing the contents of the message to be scanned.

SPAMD will perform a scan on the included message. SPAMD expects an RFC 822 or RFC 2822 formatted email.

Returns

The response will contain a 'Spam' header if the message is marked as spam as well as the score and threshold.

The body will contain the modified headers of the message.

Return type *aiospamc.responses.Response*

Raises

- *aiospamc.exceptions.BadResponse* – If the response from SPAMD is ill-formed this exception will be raised.
- *aiospamc.exceptions.AIOSpamcConnectionFailed* – Raised if an error occurred when trying to connect.
- *aiospamc.exceptions.UsageException* – Error in command line usage.
- *aiospamc.exceptions.DataErrorException* – Error with data format.
- *aiospamc.exceptions.NoInputException* – Cannot open input.
- *aiospamc.exceptions.NoUserException* – Addressee unknown.
- *aiospamc.exceptions.NoHostException* – Hostname unknown.
- *aiospamc.exceptions.UnavailableException* – Service unavailable.
- *aiospamc.exceptions.InternalSoftwareException* – Internal software error.
- *aiospamc.exceptions.OSErrorException* – System error.
- *aiospamc.exceptions.OSFileException* – Operating system file missing.
- *aiospamc.exceptions.CantCreateException* – Cannot create output file.
- *aiospamc.exceptions.IOErrorException* – Input/output error.
- *aiospamc.exceptions.TemporaryFailureException* – Temporary failure, may reattempt.
- *aiospamc.exceptions.ProtocolException* – Error in the protocol.
- *aiospamc.exceptions.NoPermissionException* – Permission denied.
- *aiospamc.exceptions.ConfigException* – Error in configuration.
- *aiospamc.exceptions.TimeoutException* – Timeout during connection.

coroutine ping()

Sends a ping request to the SPAMD service and will receive a response if the serivce is alive.

Returns Response message will contain 'PONG' if successful.

Return type *aiospamc.responses.Response*

Raises

- *aiospamc.exceptions.BadResponse* – If the response from SPAMD is ill-formed this exception will be raised.

- *aiospamc.exceptions.AIOSpamcConnectionFailed* – Raised if an error occurred when trying to connect.
- *aiospamc.exceptions.UsageException* – Error in command line usage.
- *aiospamc.exceptions.DataErrorException* – Error with data format.
- *aiospamc.exceptions.NoInputException* – Cannot open input.
- *aiospamc.exceptions.NoUserException* – Addressee unknown.
- *aiospamc.exceptions.NoHostException* – Hostname unknown.
- *aiospamc.exceptions.UnavailableException* – Service unavailable.
- *aiospamc.exceptions.InternalSoftwareException* – Internal software error.
- *aiospamc.exceptions.OSErrorException* – System error.
- *aiospamc.exceptions.OSFileException* – Operating system file missing.
- *aiospamc.exceptions.CantCreateException* – Cannot create output file.
- *aiospamc.exceptions.IOErrorException* – Input/output error.
- *aiospamc.exceptions.TemporaryFailureException* – Temporary failure, may reattempt.
- *aiospamc.exceptions.ProtocolException* – Error in the protocol.
- *aiospamc.exceptions.NoPermissionException* – Permission denied.
- *aiospamc.exceptions.ConfigException* – Error in configuration.
- *aiospamc.exceptions.TimeoutException* – Timeout during connection.

coroutine process(*message*)

Request the SPAMD service to check a message with a PROCESS request.

Parameters **message** (str) – A string containing the contents of the message to be scanned.

SPAMD will perform a scan on the included message. SPAMD expects an RFC 822 or RFC 2822 formatted email.

Returns

The response will contain a 'Spam' header if the message is marked as spam as well as the score and threshold.

The body will contain a modified version of the message.

Return type *aiospamc.responses.Response*

Raises

- *aiospamc.exceptions.BadResponse* – If the response from SPAMD is ill-formed this exception will be raised.
- *aiospamc.exceptions.AIOSpamcConnectionFailed* – Raised if an error occurred when trying to connect.
- *aiospamc.exceptions.UsageException* – Error in command line usage.
- *aiospamc.exceptions.DataErrorException* – Error with data format.
- *aiospamc.exceptions.NoInputException* – Cannot open input.
- *aiospamc.exceptions.NoUserException* – Addressee unknown.

- *aiospamc.exceptions.NoHostException* – Hostname unknown.
- *aiospamc.exceptions.UnavailableException* – Service unavailable.
- *aiospamc.exceptions.InternalSoftwareException* – Internal software error.
- *aiospamc.exceptions.OSErrorException* – System error.
- *aiospamc.exceptions.OSFileException* – Operating system file missing.
- *aiospamc.exceptions.CantCreateException* – Cannot create output file.
- *aiospamc.exceptions.IOErrorException* – Input/output error.
- *aiospamc.exceptions.TemporaryFailureException* – Temporary failure, may reattempt.
- *aiospamc.exceptions.ProtocolException* – Error in the protocol.
- *aiospamc.exceptions.NoPermissionException* – Permission denied.
- *aiospamc.exceptions.ConfigException* – Error in configuration.
- *aiospamc.exceptions.TimeoutException* – Timeout during connection.

coroutine report(*message*)

Request the SPAMD service to check a message with a REPORT request.

Parameters **message** (str) – A string containing the contents of the message to be scanned.

SPAMD will perform a scan on the included message. SPAMD expects an RFC 822 or RFC 2822 formatted email.

Returns

The response will contain a 'Spam' header if the message is marked as spam as well as the score and threshold.

The body will contain a report composed by the SPAMD service.

Return type *aiospamc.responses.Response*

Raises

- *aiospamc.exceptions.BadResponse* – If the response from SPAMD is ill-formed this exception will be raised.
- *aiospamc.exceptions.AIOSpamcConnectionFailed* – Raised if an error occurred when trying to connect.
- *aiospamc.exceptions.UsageException* – Error in command line usage.
- *aiospamc.exceptions.DataErrorException* – Error with data format.
- *aiospamc.exceptions.NoInputException* – Cannot open input.
- *aiospamc.exceptions.NoUserException* – Addressee unknown.
- *aiospamc.exceptions.NoHostException* – Hostname unknown.
- *aiospamc.exceptions.UnavailableException* – Service unavailable.
- *aiospamc.exceptions.InternalSoftwareException* – Internal software error.
- *aiospamc.exceptions.OSErrorException* – System error.
- *aiospamc.exceptions.OSFileException* – Operating system file missing.

- *aiospamc.exceptions.CantCreateException* – Cannot create output file.
- *aiospamc.exceptions.IOErrorException* – Input/output error.
- *aiospamc.exceptions.TemporaryFailureException* – Temporary failure, may reattempt.
- *aiospamc.exceptions.ProtocolException* – Error in the protocol.
- *aiospamc.exceptions.NoPermissionException* – Permission denied.
- *aiospamc.exceptions.ConfigException* – Error in configuration.
- *aiospamc.exceptions.TimeoutException* – Timeout during connection.

coroutine report_if_spam(*message*)

Request the SPAMD service to check a message with a REPORT_IFSPAM request.

Parameters **message** (str) – A string containing the contents of the message to be scanned.

SPAMD will perform a scan on the included message. SPAMD expects an RFC 822 or RFC 2822 formatted email.

Returns

The response will contain a 'Spam' header if the message is marked as spam as well as the score and threshold.

The body will contain a report composed by the SPAMD service only if message is marked as being spam.

Return type *aiospamc.responses.Response*

Raises

- *aiospamc.exceptions.BadResponse* – If the response from SPAMD is ill-formed this exception will be raised.
- *aiospamc.exceptions.AIOSpamcConnectionFailed* – Raised if an error occurred when trying to connect.
- *aiospamc.exceptions.UsageException* – Error in command line usage.
- *aiospamc.exceptions.DataErrorException* – Error with data format.
- *aiospamc.exceptions.NoInputException* – Cannot open input.
- *aiospamc.exceptions.NoUserException* – Addressee unknown.
- *aiospamc.exceptions.NoHostException* – Hostname unknown.
- *aiospamc.exceptions.UnavailableException* – Service unavailable.
- *aiospamc.exceptions.InternalSoftwareException* – Internal software error.
- *aiospamc.exceptions.OSErrorException* – System error.
- *aiospamc.exceptions.OSFileException* – Operating system file missing.
- *aiospamc.exceptions.CantCreateException* – Cannot create output file.
- *aiospamc.exceptions.IOErrorException* – Input/output error.
- *aiospamc.exceptions.TemporaryFailureException* – Temporary failure, may reattempt.
- *aiospamc.exceptions.ProtocolException* – Error in the protocol.

- *aiospamc.exceptions.NoPermissionException* – Permission denied.
- *aiospamc.exceptions.ConfigException* – Error in configuration.
- *aiospamc.exceptions.TimeoutException* – Timeout during connection.

coroutine send(*request*)

Sends a request to the SPAMD service.

If the SPAMD service gives a temporary failure response, then

Parameters **request** (*aiospamc.requests.Request*) – Request object to send.

Returns

Return type *aiospamc.responses.Response*

Raises

- *aiospamc.exceptions.BadResponse* – If the response from SPAMD is ill-formed this exception will be raised.
- *aiospamc.exceptions.AIOSpamcConnectionFailed* – Raised if an error occurred when trying to connect.
- *aiospamc.exceptions.UsageException* – Error in command line usage.
- *aiospamc.exceptions.DataErrorException* – Error with data format.
- *aiospamc.exceptions.NoInputException* – Cannot open input.
- *aiospamc.exceptions.NoUserException* – Addressee unknown.
- *aiospamc.exceptions.NoHostException* – Hostname unknown.
- *aiospamc.exceptions.UnavailableException* – Service unavailable.
- *aiospamc.exceptions.InternalSoftwareException* – Internal software error.
- *aiospamc.exceptions.OSErrorException* – System error.
- *aiospamc.exceptions.OSFileException* – Operating system file missing.
- *aiospamc.exceptions.CantCreateException* – Cannot create output file.
- *aiospamc.exceptions.IOErrorException* – Input/output error.
- *aiospamc.exceptions.TemporaryFailureException* – Temporary failure, may reattempt.
- *aiospamc.exceptions.ProtocolException* – Error in the protocol.
- *aiospamc.exceptions.NoPermissionException* – Permission denied.
- *aiospamc.exceptions.ConfigException* – Error in configuration.
- *aiospamc.exceptions.TimeoutException* – Timeout during connection.

coroutine symbols(*message*)

Request the SPAMD service to check a message with a SYMBOLS request.

The response will contain a 'Spam' header if the message is marked as spam as well as the score and threshold.

Parameters **message** (`str`) – A string containing the contents of the message to be scanned.

SPAMD will perform a scan on the included message. SPAMD expects an RFC 822 or RFC 2822 formatted email.

Returns

Will contain a 'Spam' header if the message is marked as spam as well as the score and threshold.

The body will contain a comma separated list of all the rule names.

Return type *aiospamc.responses.Response*

Raises

- *aiospamc.exceptions.BadResponse* – If the response from SPAMD is ill-formed this exception will be raised.
- *aiospamc.exceptions.AIOSpamcConnectionFailed* – Raised if an error occurred when trying to connect.
- *aiospamc.exceptions.UsageException* – Error in command line usage.
- *aiospamc.exceptions.DataErrorException* – Error with data format.
- *aiospamc.exceptions.NoInputException* – Cannot open input.
- *aiospamc.exceptions.NoUserException* – Addressee unknown.
- *aiospamc.exceptions.NoHostException* – Hostname unknown.
- *aiospamc.exceptions.UnavailableException* – Service unavailable.
- *aiospamc.exceptions.InternalSoftwareException* – Internal software error.
- *aiospamc.exceptions.OSErrorException* – System error.
- *aiospamc.exceptions.OSFileException* – Operating system file missing.
- *aiospamc.exceptions.CantCreateException* – Cannot create output file.
- *aiospamc.exceptions.IOErrorException* – Input/output error.
- *aiospamc.exceptions.TemporaryFailureException* – Temporary failure, may reattempt.
- *aiospamc.exceptions.ProtocolException* – Error in the protocol.
- *aiospamc.exceptions.NoPermissionException* – Permission denied.
- *aiospamc.exceptions.ConfigException* – Error in configuration.
- *aiospamc.exceptions.TimeoutException* – Timeout during connection.

coroutine tell(*message_class*, *message*, *remove_action=None*, *set_action=None*)

Instruct the SPAMD service to to mark the message

Parameters

- **message_class** (*aiospamc.options.MessageClassOption*) – An enumeration to classify the message as 'spam' or 'ham.'
- **message** (`str`) – A string containing the contents of the message to be scanned.

  SPAMD will perform a scan on the included message. SPAMD expects an RFC 822 or RFC 2822 formatted email.
- **remove_action** (*aiospamc.options.ActionOption*) – Remove message class for message in database.

- **set_action** (*aiospamc.options.ActionOption*) – Set message class for message in database.

Returns

Will contain a 'Spam' header if the message is marked as spam as well as the score and threshold.

The body will contain a report composed by the SPAMD service only if message is marked as being spam.

Return type *aiospamc.responses.Response*

Raises

- *aiospamc.exceptions.BadResponse* – If the response from SPAMD is ill-formed this exception will be raised.
- *aiospamc.exceptions.AIOSpamcConnectionFailed* – Raised if an error occurred when trying to connect.
- *aiospamc.exceptions.UsageException* – Error in command line usage.
- *aiospamc.exceptions.DataErrorException* – Error with data format.
- *aiospamc.exceptions.NoInputException* – Cannot open input.
- *aiospamc.exceptions.NoUserException* – Addressee unknown.
- *aiospamc.exceptions.NoHostException* – Hostname unknown.
- *aiospamc.exceptions.UnavailableException* – Service unavailable.
- *aiospamc.exceptions.InternalSoftwareException* – Internal software error.
- *aiospamc.exceptions.OSErrorException* – System error.
- *aiospamc.exceptions.OSFileException* – Operating system file missing.
- *aiospamc.exceptions.CantCreateException* – Cannot create output file.
- *aiospamc.exceptions.IOErrorException* – Input/output error.
- *aiospamc.exceptions.TemporaryFailureException* – Temporary failure, may reattempt.
- *aiospamc.exceptions.ProtocolException* – Error in the protocol.
- *aiospamc.exceptions.NoPermissionException* – Permission denied.
- *aiospamc.exceptions.ConfigException* – Error in configuration.
- *aiospamc.exceptions.TimeoutException* – Timeout during connection.

aiospamc.common module

Common classes for the project.

class aiospamc.common.**RequestResponseBase**(*body=None*, *headers=None*)

Bases: object

Base class for requests and responses.

__init__(*body=None*, *headers=None*)

Parameters

- **body** (str, optional) – String representation of the body. An instance of the aiospamc.headers.ContentLength will be automatically added.
- **headers** (tuple of *aiospamc.headers.Header*, optional) – Collection of headers to be added. If it contains an instance of aiospamc.headers.Compress then the body is automatically compressed.

add_header(*header*)
Adds a header to the request. A header with the same name will be overwritten.

Parameters **header** (*aiospamc.headers.Header*) – A header object to be added.

body
Contains the contents of the body.

The getter will return a bytes object.

The setter expects a string. If the *aiospamc.headers.Compress* header is present then the value of body will be compressed.

The deleter will automatically remove the *aiospamc.headers.ContentLength* header.

delete_header(*header_name*)
Deletes the header from the request.

Parameters **header_name** (str) – String name of the header.

Raises KeyError

get_header(*header_name*)
Gets the header matching the name.

Parameters **header_name** (str) – String name of the header.

Returns A Header object or subclass of it.

Return type *aiospamc.headers.Header*

Raises KeyError

aiospamc.connections package

Submodules

aiospamc.connections.tcp_connection module

TCP socket connection and manager.

class aiospamc.connections.tcp_connection.**TcpConnection**(*host*, *port*, *ssl*, *loop=None*)

Bases: *aiospamc.connections.Connection*

Manages a TCP connection.

host
str – Hostname or IP address of server.

port
str – Port number

ssl
bool – Whether to use SSL/TLS.

loop
asyncio.AbstratEventLoop – The asyncio event loop.

__init__(*host*, *port*, *ssl*, *loop=None*)
Constructor for TcpConnection.

host
str – Hostname or IP address of server.

port
str – Port number

ssl
`bool` or optional – SSL/TLS enabled.

connection_string
String representation of the connection.

Returns Hostname and port.

Return type str

coroutine open()
Opens a connection.

Returns

- *asyncio.StreamReader*
- *asyncio.StreamWriter*

Raises *aiospamc.exceptions.AIOSpamcConnectionFailed*

class aiospamc.connections.tcp_connection.**TcpConnectionManager**(*host*, *port*, *ssl=False*, *loop=None*)

Bases: *aiospamc.connections.ConnectionManager*

Creates new connections based on host and port provided.

host
str – Hostname or IP address of server.

port
str – Port number.

ssl
bool – Whether to use SSL/TLS.

__init__(*host*, *port*, *ssl=False*, *loop=None*)
Constructor for TcpConnectionManager.

Parameters

- **host** (*str*) – Hostname or IP address of server.
- **port** (*str*) – Port number
- **ssl** (`bool` or optional) – SSL/TLS enabled.
- **loop** (*asyncio.AbstractEventLoop*) – The asyncio event loop.

new_connection()
Creates a new TCP connection.

Raises *aiospamc.exceptions.AIOSpamcConnectionFailed*

aiospamc.connections.unix_connection module

Unix domain socket connection and manager.

class aiospamc.connections.unix_connection.**UnixConnection**(*path*, *loop=None*)
Bases: *aiospamc.connections.Connection*

Manages a Unix domain socket connection.

path
str – Path of the socket.

loop
asyncio.AbstractEventLoop – The asyncio event loop.

__init__(*path*, *loop=None*)
Constructor for UnixConnection.

Parameters

- **path** (*str*) – Path of the socket.
- **loop** (*asyncio.AbstractEventLoop*) – The asyncio event loop.

connection_string
String representation of the connection.

Returns Path to the Unix domain socket.

Return type str

coroutine open()
Opens a connection.

Returns

- *asyncio.StreamReader*
- *asyncio.StreamWriter*

Raises *aiospamc.exceptions.AIOSpamcConnectionFailed*

class aiospamc.connections.unix_connection.**UnixConnectionManager**(*path*, *loop=None*)
Bases: *aiospamc.connections.ConnectionManager*

Creates new connections based on Unix domain socket path provided.

path
str – Path of the socket.

__init__(*path*, *loop=None*)
Constructor for UnixConnectionManager.

Parameters

- **path** (*str*) – Path of the socket.
- **loop** (*asyncio.AbstractEventLoop*) – The asyncio event loop.

new_connection()
Creates a new Unix domain socket connection.

Raises AIOSpamcConnectionFailed

Module contents

Connection and ConnectionManager base classes.

class aiospamc.connections.**Connection**(*loop=None*)
Bases: object

Base class for connection objects.

connected
bool – Status on if the connection is established.

loop
asyncio.AbstratEventLoop – The asyncio event loop.

logger
logging.Logger – Logging instance. Logs to 'aiospamc.connections'

__init__(*loop=None*)
Connection constructor.

Parameters **loop** (*asyncio.AbstractEventLoop*) – The asyncio event loop.

close()
Closes the connection.

connection_string
String representation of the connection.

Returns String of the connection address.

Return type str

coroutine open()
Connect to a service.

Returns

- *asyncio.StreamReader* – Instance of stream reader.
- *asyncio.StreamWriter* – Instance of stream writer.

Raises *aiospamc.exceptions.AIOSpamcConnectionFailed* – If connection failed for some reason.

coroutine receive()
Receives data from the connection.

Returns Data received.

Return type bytes

coroutine send(*data*)
Sends data through the connection.

Parameters **data** (*bytes*) – Data to send.

class aiospamc.connections.**ConnectionManager**(*loop=None*)
Bases: object

Stores connection parameters and creates connections.

loop
asyncio.AbstratEventLoop – The asyncio event loop.

new_connection()
Creates a connection object.

Returns Instance of a Connection object.

Return type *Connection*

aiospamc.exceptions module

Collection of exceptions.

exception aiospamc.exceptions.**AIOSpamcConnectionException**
Bases: Exception

Base class for exceptions from the connection.

exception aiospamc.exceptions.**AIOSpamcConnectionFailed**
Bases: *aiospamc.exceptions.AIOSpamcConnectionException*

Connection failed.

exception aiospamc.exceptions.**BadRequest**
Bases: *aiospamc.exceptions.ClientException*

Request is not in the expected format.

exception aiospamc.exceptions.**BadResponse**
Bases: *aiospamc.exceptions.ClientException*

Response is not in the expected format.

exception aiospamc.exceptions.**CantCreateException**
Bases: *aiospamc.exceptions.ResponseException*

Can't create (user) output file.

code = 73

exception aiospamc.exceptions.**ClientException**
Bases: Exception

Base class for exceptions raised from the client.

exception aiospamc.exceptions.**ConfigException**
Bases: *aiospamc.exceptions.ResponseException*

Configuration error.

code = 78

exception aiospamc.exceptions.**DataErrorException**
Bases: *aiospamc.exceptions.ResponseException*

Data format error.

code = 65

exception aiospamc.exceptions.**IOErrorException**
Bases: *aiospamc.exceptions.ResponseException*

Input/output error.

code = 74

exception aiospamc.exceptions.**InternalSoftwareException**
Bases: *aiospamc.exceptions.ResponseException*

Internal software error.

code = 70

exception aiospamc.exceptions.**NoHostException**
Bases: *aiospamc.exceptions.ResponseException*

Hostname unknown.

code = 68

exception aiospamc.exceptions.**NoInputException**
Bases: *aiospamc.exceptions.ResponseException*

Cannot open input.

code = 66

exception aiospamc.exceptions.**NoPermissionException**
Bases: *aiospamc.exceptions.ResponseException*

Permission denied.

code = 77

exception aiospamc.exceptions.**NoUserException**
Bases: *aiospamc.exceptions.ResponseException*

Addressee unknown.

code = 67

exception aiospamc.exceptions.**OSErrorException**
Bases: *aiospamc.exceptions.ResponseException*

System error (e.g. can't fork the process).

code = 71

exception aiospamc.exceptions.**OSFileException**
Bases: *aiospamc.exceptions.ResponseException*

Critical operating system file missing.

code = 72

exception aiospamc.exceptions.**ProtocolException**
Bases: *aiospamc.exceptions.ResponseException*

Remote error in protocol.

code = 76

exception aiospamc.exceptions.**ResponseException**
Bases: Exception

Base class for exceptions raised from a response.

exception aiospamc.exceptions.**TemporaryFailureException**
Bases: *aiospamc.exceptions.ResponseException*

Temporary failure, user is invited to try again.

code = 75

exception aiospamc.exceptions.**TimeoutException**
Bases: *aiospamc.exceptions.ResponseException*

Read timeout.

code = 79

exception aiospamc.exceptions.**UnavailableException**
Bases: *aiospamc.exceptions.ResponseException*

Service unavailable.

code = 69

exception aiospamc.exceptions.**UsageException**
Bases: *aiospamc.exceptions.ResponseException*

Command line usage error.

code = 64

aiospamc.headers module

Collection of request and response headers.

class aiospamc.headers.**Compress**
Bases: *aiospamc.headers.Header*

Compress header. Specifies what encryption scheme to use. So far only 'zlib' is supported.

zlib
bool – True if the zlib compression algorithm is used.

field_name()

class aiospamc.headers.**ContentLength**(*length=0*)
Bases: *aiospamc.headers.Header*

ContentLength header. Indicates the length of the body in bytes.

length
int – Length of the body.

__init__(*length=0*)
ContentLength constructor.

Parameters **length** (int, optional) – Length of the body.

field_name()

class aiospamc.headers.**DidRemove**(*action=None*)
Bases: aiospamc.headers._SetRemoveBase

DidRemove header. Used by SPAMD to indicate if a message was removed from either a local or remote database in response to a TELL request.

action
aiospamc.options.ActionOption – Actions to be done on local or remote.

__init__(*action=None*)
_SetRemoveBase constructor.

Parameters **action** (*aiospamc.options.ActionOption*, optional) – Actions to be done on local or remote.

field_name()

class aiospamc.headers.**DidSet**(*action=None*)
Bases: aiospamc.headers._SetRemoveBase

DidRemove header. Used by SPAMD to indicate if a message was added to either a local or remote database in response to a TELL request.

action
aiospamc.options.ActionOption – Actions to be done on local or remote.

__init__(*action=None*)
_SetRemoveBase constructor.

Parameters **action** (*aiospamc.options.ActionOption*, optional) – Actions to be done on local or remote.

field_name()

class aiospamc.headers.**Header**
Bases: object

Header base class.

field_name()
Returns the the field name for the header.

Returns

Return type str

class aiospamc.headers.**MessageClass**(*value=None*)
Bases: *aiospamc.headers.Header*

MessageClass header. Used to specify whether a message is 'spam' or 'ham.'

value
aiospamc.options.MessageClassOption – Specifies the classification of the message.

__init__(*value=None*)
MessageClass constructor.

Parameters **value** (*aiospamc.options.MessageClassOption*, optional) – Specifies the classification of the message.

field_name()

class aiospamc.headers.**Remove**(*action=None*)
Bases: aiospamc.headers._SetRemoveBase

Remove header. Used in a TELL request to ask the SPAMD service remove a message from a local or remote database. The SPAMD service must have the –allow-tells switch in order for this to do anything.

action
aiospamc.options.ActionOption – Actions to be done on local or remote.

__init__(*action=None*)
_SetRemoveBase constructor.

Parameters **action** (*aiospamc.options.ActionOption*, optional) – Actions to be done on local or remote.

field_name()

class aiospamc.headers.**Set**(*action=None*)
Bases: aiospamc.headers._SetRemoveBase

Set header. Used in a TELL request to ask the SPAMD service add a message from a local or remote database. The SPAMD service must have the –allow-tells switch in order for this to do anything.

action
aiospamc.options.ActionOption – Actions to be done on local or remote.

__init__(*action=None*)
_SetRemoveBase constructor.

Parameters **action** (*aiospamc.options.ActionOption*, optional) – Actions to be done on local or remote.

field_name()

class aiospamc.headers.**Spam**(*value=False*, *score=0.0*, *threshold=0.0*)
Bases: *aiospamc.headers.Header*

Spam header. Used by the SPAMD service to report on if the submitted message was spam and the score/threshold that it used.

value
bool – True if the message is spam, False if not.

score
float – Score of the message after being scanned.

threshold
float – Threshold of which the message would have been marked as spam.

__init__(*value=False*, *score=0.0*, *threshold=0.0*)
Spam header constructor.

Parameters

- **value** (bool, optional) – True if the message is spam, False if not.
- **score** (float, optional) – Score of the message after being scanned.
- **threshold** (float, optional) – Threshold of which the message would have been marked as spam.

field_name()

class aiospamc.headers.**User**(*name=None*)
Bases: *aiospamc.headers.Header*

User header. Used to specify which user the SPAMD service should use when loading configuration files.

name
str – Name of the user account.

__init__(*name=None*)
User constructor.

Parameters **name** (str, optional) – Name of the user account.

field_name()

class aiospamc.headers.**XHeader**(*name*, *value*)
Bases: *aiospamc.headers.Header*

Extension header. Used to specify a header that's not supported natively by the SPAMD service.

name
str – Name of the header.

value
str – Contents of the value.

__init__(*name*, *value*)
XHeader constructor.

Parameters

- **name** (str) – Name of the header.
- **value** (str) – Contents of the value.

field_name()

aiospamc.options module

Data structures used for function parameters.

class aiospamc.options.**ActionOption**(*local*, *remote*)
Bases: tuple

count(*value*) → integer – return number of occurrences of value

index(*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

local
Alias for field number 0

remote
Alias for field number 1

class aiospamc.options.**MessageClassOption**
Bases: enum.IntEnum

Option to be used for the MessageClass header.

ham = 2

spam = 1

aiospamc.parser module

Parser combinators for the SPAMC/SPAMD protocol.

class aiospamc.parser.**Body**
Bases: *aiospamc.parser.Parser*

Matches the remaining stream as the body of the request/response.

class aiospamc.parser.**Boolean**
Bases: *aiospamc.parser.Parser*

Matches and returns a bool.

class aiospamc.parser.**CompressHeader**
Bases: *aiospamc.parser.Parser*

Matches a *aiospamc.headers.Compress* header and returns an instance if successful.

class aiospamc.parser.**ContentLengthHeader**
Bases: *aiospamc.parser.Parser*

Matches a *aiospamc.headers.ContentLength* header and returns an instance if successful.

class aiospamc.parser.**CustomHeader**
Bases: *aiospamc.parser.Parser*

Matches a custom header and returns an instance of *aiospamc.headers.XHeader* if successful.

class aiospamc.parser.**DidRemoveHeader**
Bases: *aiospamc.parser.Parser*

Matches a *aiospamc.headers.DidRemove* header and returns an instance if successful.

class aiospamc.parser.**DidSetHeader**
Bases: *aiospamc.parser.Parser*

Matches a *aiospamc.headers.DidSet* header and returns an instance if successful.

class aiospamc.parser.**Digits**
Bases: *aiospamc.parser.Parser*

Matches a sequence of digits and returns a string.

class aiospamc.parser.**Discard**(*parser*)
Bases: *aiospamc.parser.Parser*

Shortcut for *Replace* which replaces with a `None` value.

Parameters **parser** (*aiospamc.parser.Parser*) –

class aiospamc.parser.**Failure**(*error*, *remaining*)
Bases: `object`

Contains error message and location the where the parser failed.

error
`str` – Error message.

remaining
aiospamc.parser.Stream – Remaining stream to parse.

__init__(*error*, *remaining*)
Failure object constructor.

Parameters

- **error** (`str`) – Error message.
- **remaining** (*aiospamc.parser.Stream*) – Remaining stream to parse.

class aiospamc.parser.**FalseValue**
Bases: *aiospamc.parser.Parser*

Matches the string “false” or “False” and returns the boolean value.

class aiospamc.parser.**Float**
Bases: *aiospamc.parser.Parser*

Matches a floating point number and returns a float.

class aiospamc.parser.**Header**(*name*, *value*)
Bases: *aiospamc.parser.Parser*

Matches a header format, name and value separated by a colon and terminated by a newline.

class aiospamc.parser.**Headers**
Bases: *aiospamc.parser.Parser*

Matches headers and returns an instance if successful.

class aiospamc.parser.**Integer**
Bases: *aiospamc.parser.Parser*

Matches a sequence of digits and returns an integer.

class aiospamc.parser.**Map**(*parser*, *func*)
Bases: *aiospamc.parser.Parser*

Applies the referenced function to the value returned from the parser.

parser
aiospamc.parser.Parser

func
callable

class aiospamc.parser.**Match**(*match*)
Bases: *aiospamc.parser.Parser*

Matches a regular expression. Returns the regular expression result as the value if successful.

Parameters **match** (bytes) – Regular expression to match.

class aiospamc.parser.**MessageClassHeader**
Bases: *aiospamc.parser.Parser*

Matches a *aiospamc.headers.MessageClass* header and returns an instance if successful.

class aiospamc.parser.**MessageClassOption**
Bases: *aiospamc.parser.Parser*

Matches 'spam' or 'ham' and returns an instance of *aiospamc.options.MessageClassOption*.

class aiospamc.parser.**Newline**
Bases: *aiospamc.parser.Parser*

Matches newline sequence and returns regular expression result.

class aiospamc.parser.**Number**
Bases: *aiospamc.parser.Parser*

Matches either an integer or a float.

class aiospamc.parser.**OneOrMore**(*parser*)
Bases: *aiospamc.parser.Parser*

Matches one or more repetitions of the parser.

Parameters **parser** (*aiospamc.parser.Parser*) –

class aiospamc.parser.**Optional**(*parser*)
Bases: *aiospamc.parser.Parser*

If the parser is found then the value is returned, if not then a None value is returned.

Parameters **parser** (*aiospamc.parser.Parser*) –

class aiospamc.parser.**Or**(*left*, *right*)
Bases: *aiospamc.parser.Parser*

Matches the left or right parser.

left
aiospamc.parser.Parser

right
aiospamc.parser.Parser

class aiospamc.parser.**Parser**
Bases: object

Parser base class. Overloads operators for ease of writing parsers.

The following lists the operator and the effect it has:

Operator	Effect
-A	Matches A, but discards the result
~A	Optionally matches A
A | B	Logical OR
A >> B	Sequence of A then B
A ^ B	Logical XOR

class aiospamc.parser.**RemoveHeader**
Bases: *aiospamc.parser.Parser*

Matches a *aiospamc.headers.Remove* header and returns an instance if successful.

class aiospamc.parser.**Replace**(*parser*, *replace*)
Bases: *aiospamc.parser.Parser*

If parser returns a successful result then it's replaced with the value given.

Parameters

- **parser** (*aiospamc.parser.Parser*) –
- **replace** – Value to replace a successful result with.

class aiospamc.parser.**Request**
Bases: *aiospamc.parser.Parser*

Matches a SPAMC request and returns an instance of *aiospamc.requests.Request*.

class aiospamc.parser.**RequestMethod**
Bases: *aiospamc.parser.Parser*

Matches a request method.

class aiospamc.parser.**Response**
Bases: *aiospamc.parser.Parser*

Matches a SPAMD response and returns an instance of *aiospamc.responses.Response*.

class aiospamc.parser.**SAParser**
Bases: *aiospamc.parser.Parser*

Start rule for the parser. Returns an instance of either *aiospamc.requests.Request* or *aiospamc.responses.Response*.

class aiospamc.parser.**Sequence**(*left*, *right*)
Bases: *aiospamc.parser.Parser*

Matches the left and then the right parser in sequence.

left
aiospamc.parser.Parser

right
aiospamc.parser.Parser

class aiospamc.parser.**SetHeader**
Bases: *aiospamc.parser.Parser*

Matches a *aiospamc.headers.Set* header and returns an instance if successful.

class aiospamc.parser.**SetRemoveValue**
Bases: *aiospamc.parser.Parser*

Matches a value for the *aiospamc.headers.DidRemove*, *aiospamc.headers.DidSet*, *aiospamc.headers.Remove*, or *aiospamc.headers.Set*.

class aiospamc.parser.**SpamHeader**
Bases: *aiospamc.parser.Parser*

Matches a *aiospamc.headers.Spam* header and returns an instance if successful.

class aiospamc.parser.**Str**(*match*)
Bases: *aiospamc.parser.Parser*

Matches a regular expression and casts it to a string.

Parameters **match** (bytes) – Regular expression to match.

class aiospamc.parser.**Stream**(*stream*, *index*)
Bases: tuple

count(*value*) → integer – return number of occurrences of value

index
Alias for field number 1

stream
Alias for field number 0

class aiospamc.parser.**Success**(*value*, *remaining*)
Bases: object

Contains successful result and location for a parser.

value
Result of a successful parse.

remaining
aiospamc.parser.Stream – Remaining stream to parse.

__init__(*value*, *remaining*)
Success object constructor.

Parameters

- **value** – Result of a successful parse.
- **remaining** (*aiospamc.parser.Stream*) – Remaining stream to parse.

class aiospamc.parser.**TrueValue**
Bases: *aiospamc.parser.Parser*

Matches the string “true” or “True” and returns the boolean value.

class aiospamc.parser.**UserHeader**
Bases: *aiospamc.parser.Parser*

Matches a *aiospamc.headers.User* header and returns an instance if successful.

class aiospamc.parser.**Version**
Bases: *aiospamc.parser.Parser*

Matches a version code and returns a string.

class aiospamc.parser.**Whitespace**
Bases: *aiospamc.parser.Parser*

Matches sequence of whitespace and returns regular expression result.

class aiospamc.parser.**Xor**(*left*, *right*)
Bases: *aiospamc.parser.Parser*

Matches only either the left or right parser.

left
aiospamc.parser.Parser

right
aiospamc.parser.Parser

class aiospamc.parser.**ZeroOrMore**(*parser*)
Bases: *aiospamc.parser.Parser*

Matches zero or more repetitions of the parser.

Parameters **parser** (*aiospamc.parser.Parser*) –

aiospamc.parser.**concat**(*container*)
Concatenates items within a collection into a string.

aiospamc.parser.**flatten**(*container*)
Flattens nested lists into one list.

aiospamc.parser.**remove_empty**(*container*)
Removes None and empty strings from the container.

aiospamc.requests module

Contains all requests that can be made to the SPAMD service.

class aiospamc.requests.**Request**(*verb*, *version='1.5'*, *headers=None*, *body=None*)
Bases: *aiospamc.common.RequestResponseBase*

SPAMC request object.

verb
str – Method name of the request.

version
str – Protocol version.

body
str or bytes – String representation of the body. An instance of the *aiospamc.headers.ContentLength* will be automatically added.

__init__(*verb*, *version='1.5'*, *headers=None*, *body=None*)
Request constructor.

Parameters

- **verb** (str) – Method name of the request.
- **version** (str) – Version of the protocol.

- **body** (str or bytes, optional) – String representation of the body. An instance of the *aiospamc.headers.ContentLength* will be automatically added.
- **headers** (tuple of *aiospamc.headers.Header*, optional) – Collection of headers to be added. If it contains an instance of *aiospamc.headers.Compress* then the body is automatically compressed.

add_header(*header*)
Adds a header to the request. A header with the same name will be overwritten.

Parameters **header** (*aiospamc.headers.Header*) – A header object to be added.

body
Contains the contents of the body.

The getter will return a bytes object.

The setter expects a string. If the *aiospamc.headers.Compress* header is present then the value of body will be compressed.

The deleter will automatically remove the *aiospamc.headers.ContentLength* header.

delete_header(*header_name*)
Deletes the header from the request.

Parameters **header_name** (str) – String name of the header.

Raises KeyError

get_header(*header_name*)
Gets the header matching the name.

Parameters **header_name** (str) – String name of the header.

Returns A Header object or subclass of it.

Return type *aiospamc.headers.Header*

Raises KeyError

aiospamc.responses module

Contains classes used for responses.

class aiospamc.responses.**Response**(*version*, *status_code*, *message*, *headers=None*, *body=None*)
Bases: *aiospamc.common.RequestResponseBase*

Class to encapsulate response.

protocol_version
str – Protocol version given by the response.

status_code
aiospamc.responess.Status – Status code give by the response.

message
str – Message accompanying the status code.

body
str or bytes – Contents of the response body.

__init__(*version*, *status_code*, *message*, *headers=None*, *body=None*)
Response constructor.

Parameters

- **version** (str) – Version reported by the SPAMD service response.
- **status_code** (*aiospamc.responses.Status*) – Success or error code.
- **message** (str) – Message associated with status code.
- **body** (str or bytes, optional) – String representation of the body. An instance of the *aiospamc.headers.ContentLength* will be automatically added.
- **headers** (tuple of *aiospamc.headers.Header*, optional) – Collection of headers to be added. If it contains an instance of *aiospamc.headers.Compress* then the body is automatically compressed.

add_header(*header*)
Adds a header to the request. A header with the same name will be overwritten.

Parameters **header** (*aiospamc.headers.Header*) – A header object to be added.

body
Contains the contents of the body.

The getter will return a bytes object.

The setter expects a string. If the *aiospamc.headers.Compress* header is present then the value of body will be compressed.

The deleter will automatically remove the *aiospamc.headers.ContentLength* header.

delete_header(*header_name*)
Deletes the header from the request.

Parameters **header_name** (str) – String name of the header.

Raises KeyError

get_header(*header_name*)
Gets the header matching the name.

Parameters **header_name** (str) – String name of the header.

Returns A Header object or subclass of it.

Return type *aiospamc.headers.Header*

Raises KeyError

class aiospamc.responses.**Status**
Bases: enum.IntEnum

Enumeration of status codes that the SPAMD will accompany with a response.

Reference: https://svn.apache.org/repos/asf/spamassassin/trunk/spamd/spamd.raw Look for the %resphash variable.

EX_CANTCREAT = 73

EX_CONFIG = 78

EX_DATAERR = 65

EX_IOERR = 74

EX_NOHOST = 68

EX_NOINPUT = 66

EX_NOPERM = 77

EX_NOUSER = 67

EX_OK = 0

EX_OSERR = 71

EX_OSFILE = 72

EX_PROTOCOL = 76

EX_SOFTWARE = 70

EX_TEMPFAIL = 75

EX_TIMEOUT = 79

EX_UNAVAILABLE = 69

EX_USAGE = 64

Module contents

aiospamc package.

An asyncio-based library to communicate with SpamAssassin's SPAMD service.

1.3 SPAMC/SPAMD Protocol As Implemented by SpamAssassin

1.3.1 Requests and Responses

The structure of a request is similar to an HTTP request.[1] The method/verb, protocol name and version are listed followed by headers separated by newline characters (carriage return and linefeed or `\r\n`). Following the headers is a blank line with a newline (`\r\n`). If there is a message body it will be added after all headers.

The current requests are *CHECK*, *HEADERS*, *PING*, *PROCESS*, *REPORT*, *REPORT_IFSPAM*, *SKIP*, *SYMBOLS*, and *TELL*:

```
METHOD SPAMC/1.5\r\n
HEADER_NAME1 : HEADER_VALUE1\r\n
HEADER_NAME2 : HEADER_VALUE2\r\n
\r\n
REQUEST_BODY
```

The structure of responses are also similar to HTTP responses. The protocol name, version, status code, and message are listed on one line. Any headers are also listed and all are separated by newline characters. Following the headers is a newline. If there is a message body it's included after all headers:

```
SPAMD/1.5 STATUS_CODE MESSAGE\r\n
HEADER_NAME1 : HEADER_VALUE1\r\n
HEADER_NAME2 : HEADER_VALUE2\r\n
\r\n
RESPONSE_BODY
```

The following are descriptions of the requests that can be sent and examples of the responses that you can expect to receive.

[1] https://svn.apache.org/viewvc/spamassassin/branches/3.4/spamd/PROTOCOL?revision=1676616&view=co

CHECK

Instruct SpamAssassin to process the included message.

Request

Required Headers

- *Content-length*

Optional Headers

- *Compress*
- *User*

Required body

An email based on the **RFC 5322** standard.

Response

Will include a Spam header with a "True" or "False" value, followed by the score and threshold. Example:

```
SPAMD/1.1 0 EX_OK
Spam: True ; 1000.0 / 5.0
```

HEADERS

Process the included message and return only the modified headers.

Request

Required Headers

- *Content-length*

Optional Headers

- *Compress*
- *User*

Required Body

An email based on the **RFC 5322** standard.

Response

Will return the modified headers of the message in the body. The *Spam* header is also included.

```
SPAMD/1.1 0 EX_OK
Spam: True ; 1000.0 / 5.0
Content-length: 654

Received: from localhost by debian
    with SpamAssassin (version 3.4.0);
    Tue, 10 Jan 2017 11:09:26 -0500
From: Sender <sender@example.net>
To: Recipient <recipient@example.net>
Subject: Test spam mail (GTUBE)
Date: Wed, 23 Jul 2003 23:30:00 +0200
Message-Id: <GTUBE1.1010101@example.net>
X-Spam-Checker-Version: SpamAssassin 3.4.0 (2014-02-07) on debian
X-Spam-Flag: YES
X-Spam-Level: **********************************************
X-Spam-Status: Yes, score=1000.0 required=5.0 tests=GTUBE,NO_RECEIVED,
    NO_RELAYS autolearn=no autolearn_force=no version=3.4.0
MIME-Version: 1.0Content-Type: multipart/mixed; boundary="----------=_58750736.
↪8D9F70BC"
```

PING

Send a request to test if the server is alive.

Request

Required Headers

None.

Optional Headers

None.

Response

Example:

```
SPAMD/1.5 0 PONG
```

PROCESS

Instruct SpamAssassin to process the message and return the modified message.

Request

Required Headers

- *Content-length*

Optional Headers

- *Compress*
- *User*

Required Body

An email based on the **RFC 5322** standard.

Response

Will return a modified message in the body. The *Spam* header is also included. Example:

```
SPAMD/1.1 0 EX_OK
Spam: True ; 1000.0 / 5.0
Content-length: 2948

Received: from localhost by debian
    with SpamAssassin (version 3.4.0);
    Tue, 10 Jan 2017 10:57:02 -0500
From: Sender <sender@example.net>
To: Recipient <recipient@example.net>
Subject: Test spam mail (GTUBE)
Date: Wed, 23 Jul 2003 23:30:00 +0200
Message-Id: <GTUBE1.1010101@example.net>
X-Spam-Checker-Version: SpamAssassin 3.4.0 (2014-02-07) on debian
X-Spam-Flag: YES
X-Spam-Level: **************************************************
X-Spam-Status: Yes, score=1000.0 required=5.0 tests=GTUBE,NO_RECEIVED,
    NO_RELAYS autolearn=no autolearn_force=no version=3.4.0
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----------=_5875044E.D4EFFFD7"

This is a multi-part message in MIME format.

------------=_5875044E.D4EFFFD7
Content-Type: text/plain; charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Spam detection software, running on the system "debian",
has identified this incoming email as possible spam.  The original
message has been attached to this so you can view it or label
similar future email.  If you have any questions, see
@@CONTACT_ADDRESS@@ for details.

Content preview:  This is the GTUBE, the Generic Test for Unsolicited Bulk Email
```

```
If your spam filter supports it, the GTUBE provides a test by which you can
verify that the filter is installed correctly and is detecting incoming spam.
You can send yourself a test mail containing the following string of characters
(in upper case and with no white spaces and line breaks): [...]

Content analysis details:   (1000.0 points, 5.0 required)

pts rule name              description
---- ---------------------- --------------------------------------------------
1000 GTUBE                   BODY: Generic Test for Unsolicited Bulk Email
-0.0 NO_RELAYS               Informational: message was not relayed via SMTP
-0.0 NO_RECEIVED             Informational: message has no Received headers



------------=_5875044E.D4EFFFD7
Content-Type: message/rfc822; x-spam-type=original
Content-Description: original message before SpamAssassin
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Subject: Test spam mail (GTUBE)
Message-ID: <GTUBE1.1010101@example.net>
Date: Wed, 23 Jul 2003 23:30:00 +0200
From: Sender <sender@example.net>
To: Recipient <recipient@example.net>
Precedence: junk
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

This is the GTUBE, the
    Generic
    Test for
    Unsolicited
    Bulk
    Email

If your spam filter supports it, the GTUBE provides a test by which you
can verify that the filter is installed correctly and is detecting incoming
spam. You can send yourself a test mail containing the following string of
characters (in upper case and with no white spaces and line breaks):

XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X

You should send this test mail from an account outside of your network.


------------=_5875044E.D4EFFFD7--
```

REPORT

Send a request to process a message and return a report.

Request

Required Headers

- *Content-length*

Optional Headers

- *Compress*
- *User*

Required body

An email based on the **RFC 5322** standard.

Response

Response returns the *Spam* header and the body containing a report of the message scanned.

Example:

```
SPAMD/1.1 0 EX_OK
Content-length: 1071
Spam: True ; 1000.0 / 5.0

Spam detection software, running on the system "debian",
has identified this incoming email as possible spam.  The original
message has been attached to this so you can view it or label
similar future email.  If you have any questions, see
@@CONTACT_ADDRESS@@ for details.

Content preview:  This is the GTUBE, the Generic Test for Unsolicited Bulk Email
   If your spam filter supports it, the GTUBE provides a test by which you can
   verify that the filter is installed correctly and is detecting incoming spam.
   You can send yourself a test mail containing the following string of characters
   (in upper case and with no white spaces and line breaks): [...]

Content analysis details:   (1000.0 points, 5.0 required)

 pts rule name              description
---- ---------------------- --------------------------------------------------
1000 GTUBE                  BODY: Generic Test for Unsolicited Bulk Email
-0.0 NO_RELAYS              Informational: message was not relayed via SMTP
-0.0 NO_RECEIVED            Informational: message has no Received headers
```

REPORT_IFSPAM

Matches the *REPORT* request, with the exception a report will not be generated if the message is not spam.

SKIP

Sent when a connection is made in error. The SPAMD service will immediately close the connection.

Request

Required Headers

None.

Optional Headers

None.

SYMBOLS

Instruct SpamAssassin to process the message and return the rules that were matched.

Request

Required Headers

- *Content-length*

Optional Headers

- *Compress*
- *User*

Required body

An email based on the **RFC 5322** standard.

Response

Response includes the *Spam* header. The body contains the SpamAssassin rules that were matched. Example:

```
SPAMD/1.1 0 EX_OK
Content-length: 27
Spam: True ; 1000.0 / 5.0

GTUBE,NO_RECEIVED,NO_RELAYS
```

TELL

Send a request to classify a message and add or remove it from a database. The message type is defined by the *Message-class*. The *Remove* and *Set* headers are used to choose the location ("local" or "remote") to add or remove it. SpamAssassin will return an error if a request tries to apply a conflicting change (e.g. both setting and removing to the same location).

Note: The SpamAssassin daemon must have the `--allow-tell` option enabled to support this feature.

Request

Required Headers

- *Content-length*
- *Message-class*
- *Remove* or *Set*
- *User*

Optional Headers

- *Compress*

Required Body

An email based on the **RFC 5322** standard.

Response

If successful, the response will include the *DidRemove* or *DidSet* headers depending on the request.

Response from a request that sent a *Remove*:

```
SPAMD/1.1 0 EX_OK
DidRemove: local
Content-length: 2
```

Response from a request that sent a *Set*:

```
SPAMD/1.1 0 EX_OK
DidSet: local
Content-length: 2
```

1.3.2 Headers

Headers are structured very simply. They have a name and value which are separated by a colon (:). All headers are followed by a newline. The current headers include *Compress*, *Content-length*, *DidRemove*, *DidSet*, *Message-class*, *Remove*, *Set*, *Spam*, and *User*.

For example:

```
Content-length: 42\r\n
```

The following is a list of headers defined by SpamAssassin, although anything is allowable as a header. If an unrecognized header is included in the request or response it should be ignored.

Compress

Specifies that the body is compressed and what compression algorithm is used. Contains a string of the compression algorithm. Currently only `zlib` is supported.

Content-length

The length of the body in bytes. Contains an integer representing the body length.

DidRemove

Included in a response to a *TELL* request. Identifies which databases a message was removed from. Contains a string containing either `local`, `remote` or both seprated by a comma.

DidSet

Included in a response to a *TELL* request. Identifies which databases a message was set in. Contains a string containing either `local`, `remote` or both seprated by a comma.

Message-class

Classifies the message contained in the body. Contains a string containing either `local`, `remote` or both seprated by a comma.

Remove

Included in a *TELL* request to remove the message from the specified database. Contains a string containing either `local`, `remote` or both seprated by a comma.

Set

Included in a *TELL* request to remove the message from the specified database. Contains a string containing either `local`, `remote` or both seprated by a comma.

Spam

Identify whether the message submitted was spam or not including the score and threshold. Contains a string containing a boolean if the message is spam (either `True`, `False`, `Yes`, or `No`), followed by a `;`, a floating point number representing the score, followed by a `/`, and finally a floating point number representing the threshold of which to consider it spam.

For example:

```
Spam: True ; 1000.0 / 5.0
```

User

Specify which user the request will run under. SpamAssassin will use the configuration files for the user included in the header. Contains a string containing the name of the user.

1.3.3 Status Codes

A status code is an integer detailing whether the request was successful or if an error occurred.

The following status codes are defined in the SpamAssassin source repository[2].

EX_OK

Code: 0

Definition: No problems were found.

EX_USAGE

Code: 64

Definition: Command line usage error.

EX_DATAERR

Code: 65

Definition: Data format error.

EX_NOINPUT

Code: 66

Definition: Cannot open input.

EX_NOUSER

Code: 67

Definition: Addressee unknown.

EX_NOHOST

Code: 68

Definition: Hostname unknown.

[2] https://svn.apache.org/viewvc/spamassassin/branches/3.4/spamd/spamd.raw?revision=1749346&view=co

EX_UNAVAILABLE

Code: 69

Definition: Service unavailable.

EX_SOFTWARE

Code: 70

Definition: Internal software error.

EX_OSERR

Code: 71

Definition: System error (e.g. can't fork the process).

EX_OSFILE

Code: 72

Definition: Critical operating system file missing.

EX_CANTCREAT

Code: 73

Definition: Can't create (user) output file.

EX_IOERR

Code: 74

Definition: Input/output error.

EX_TEMPFAIL

Code: 75

Definition: Temporary failure, user is invited to retry.

EX_PROTOCOL

Code: 76

Definition: Remote error in protocol.

EX_NOPERM

Code: 77

Definition: Permission denied.

EX_CONFIG

Code: 78

Definition: Configuration error.

EX_TIMEOUT

Code: 79

Definition: Read timeout.

1.3.4 Body

SpamAssassin will generally want the body of a request to be in a supported RFC email format. The response body will differ depending on the type of request that was sent.

1.3.5 References

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Z