
aiospamc Documentation

Release 0.4.0

Michael Caley

Feb 16, 2018

Contents:

1	Contents	3
1.1	User Guide	3
1.2	aiospamc API Reference	4
1.3	SPAMC/SPAMD Protocol As Implemented by SpamAssassin	29
2	Indices and tables	41
	Python Module Index	43

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

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:

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:

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.

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 local-host.
- **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 retry.
- `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 retry.
- *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 service 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 retry.
- *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 retry.
- `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 retry.
- `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.IOExceptionException` – Input/output error.
- `aiospamc.exceptions.TemporaryFailureException` – Temporary failure, may retry.
- `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.IOExceptionException` – Input/output error.
- `aiospamc.exceptions.TemporaryFailureException` – Temporary failure, may retry.

- *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 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 retry.
- `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.AbstractEventLoop – 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.AbstractEventLoop – 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.AbstractEventLoop – 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 object for SPAMC/SPAMD requests and responses.

exception aiospamc.parser.**ParseError** (*index, message*)

Bases: Exception

An exception occurring when parsing.

index

int – Index in the stream the exception occurred.

message

str – User readable message.

__init__ (*index, message*)

ParseError constructor.

Parameters

- **index** (int) – Index in the stream the exception occurred.
- **message** (str) – User readable message.

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class aiospamc.parser.**Parser** (*string=None*)

Bases: object

Parser object for requests and responses.

__init__ (*string=None*)

Parser constructor.

Parameters **string** (str, optional) – The string to parse.

advance (*by*)

Advance the current index by number of bytes.

Parameters **by** (int) – Number of bytes in the stream to advance.

body ()

Consumes the rest of the message and returns the contents.

Returns

Return type bytes

compress_value ()

Consumes the Compression header value.

Returns

Return type str

consume (*pattern*)

If the pattern matches, advances the index the length of the match. Returns the regular expression match.

Parameters **pattern** (bytes) –

Returns

Return type Regular expression match

Raises *aiospamc.parser.ParseError*

content_length_value()

Consumes the Content-length header value.

Returns

Return type `int`

current()

The remainder of the string that hasn't been parsed.

Returns

Return type `str`

end()

Whether the parser has parsed the entire string.

Returns

Return type `bool`

header()

Consumes the string and returns an instance of *aiospamc.headers.Header*.

Returns

Return type *aiospamc.headers.Header*

headers()

Consumes all headers.

Returns

Return type list of *aiospamc.headers.Header*

match(pattern)

Returns the regular expression matches string at the current index.

Parameters **pattern** (bytes) –

Returns

Return type Regular expression match

message()

Consumes a string until it matches a newline.

Returns

Return type `str`

message_class_value()

Consumes the Message-class header value.

Returns

Return type *aiospamc.options.MessageClassOption*

method()

Consumes the method name in a request.

Returns

Return type `str`

newline()

Consumes a newline sequence (carriage return and line feed).

request()

Consumes a SPAMC request.

Returns

Return type `aiospamc.requests.Request`

response()

Consumes a SPAMD response.

Returns

Return type `aiospamc.responses.Response`

set_remove_value()

Consumes the value for the DidRemove, DidSet, Remove and Set headers.

Returns

Return type `aiospamc.options.ActionOption`

static skip(func)

Makes the parser function optional by ignore whether it raises a `aiospamc.parser.ParseError` exception or not.

Parameters **func** (*function*) – Function to execute.

spam_value()

Consumes the Spam header value.

Returns Has the keys *value*, *score*, and *threshold*.

Return type dict

spamc_protocol()

Consumes the string “SPAMC”.

Returns

Return type str

spamd_protocol()

Consumes the string “SPAMD”.

Returns

Return type str

status_code()

Consumes the status code.

Returns

Return type `aiospamc.responses.StatusCode` or int

user_value()

Consumes the User header value.

Returns

Return type str

version()

Consumes a version pattern. For example, “1.5”.

Returns**Return type** `str`**whitespace()**

Consumes spaces or tabs.

`aiospamc.parser.checkpoint` (*func*)

A decorator to restore the index if an exception occurred.

`aiospamc.parser.parse` (*string*)

Parses a request or response.

Returns**Return type** `aiospamc.requests.Request` or `aiospamc.responses.Response`**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.responses.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 the first 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.

CHECK

Instruct SpamAssassin to process the included message.

Request

Required Headers

- *Content-length*

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

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.D4EFFF7"

This is a multi-part message in MIME format.

-----=_5875044E.D4EFFF7
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.D4EFFFFD7
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.D4EFFFFD7--
```

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* and/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* and/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 repository².

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

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.

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`

a

- `aiosпамc`, [29](#)
- `aiosпамc.client`, [4](#)
- `aiosпамc.common`, [13](#)
- `aiosпамc.connections`, [16](#)
- `aiosпамc.connections.tcp_connection`, [14](#)
- `aiosпамc.connections.unix_connection`,
[15](#)
- `aiosпамc.exceptions`, [17](#)
- `aiosпамc.headers`, [19](#)
- `aiosпамc.options`, [22](#)
- `aiosпамc.parser`, [23](#)
- `aiosпамc.requests`, [26](#)
- `aiosпамc.responses`, [27](#)

Symbols

- `__init__()` (aiospamc.client.Client method), 5
 - `__init__()` (aiospamc.common.RequestResponseBase method), 13
 - `__init__()` (aiospamc.connections.Connection method), 16
 - `__init__()` (aiospamc.connections.tcp_connection.TcpConnection method), 14
 - `__init__()` (aiospamc.connections.tcp_connection.TcpConnectionManager method), 15
 - `__init__()` (aiospamc.connections.unix_connection.UnixConnection method), 15
 - `__init__()` (aiospamc.connections.unix_connection.UnixConnectionManager method), 16
 - `__init__()` (aiospamc.headers.ContentLength method), 20
 - `__init__()` (aiospamc.headers.DidRemove method), 20
 - `__init__()` (aiospamc.headers.DidSet method), 20
 - `__init__()` (aiospamc.headers.MessageClass method), 20
 - `__init__()` (aiospamc.headers.Remove method), 21
 - `__init__()` (aiospamc.headers.Set method), 21
 - `__init__()` (aiospamc.headers.Spam method), 21
 - `__init__()` (aiospamc.headers.User method), 22
 - `__init__()` (aiospamc.headers.XHeader method), 22
 - `__init__()` (aiospamc.parser.ParseError method), 23
 - `__init__()` (aiospamc.parser.Parser method), 23
 - `__init__()` (aiospamc.requests.Request method), 26
 - `__init__()` (aiospamc.responses.Response method), 27
- ## A
- action (aiospamc.headers.DidRemove attribute), 20
 - action (aiospamc.headers.DidSet attribute), 20
 - action (aiospamc.headers.Remove attribute), 21
 - action (aiospamc.headers.Set attribute), 21
 - ActionOption (class in aiospamc.options), 22
 - add_header() (aiospamc.common.RequestResponseBase method), 13
 - add_header() (aiospamc.requests.Request method), 26
 - add_header() (aiospamc.responses.Response method), 27
 - advance() (aiospamc.parser.Parser method), 23
 - aiospamc (module), 29
 - aiospamc.client (module), 4
 - aiospamc.common (module), 13
 - aiospamc.connections (module), 16
 - aiospamc.connections.tcp_connection (module), 14
 - aiospamc.connections.unix_connection (module), 15
 - aiospamc.exceptions (module), 17
 - aiospamc.headers (module), 19
 - aiospamc.options (module), 22
 - aiospamc.parser (module), 23
 - aiospamc.requests (module), 26
 - aiospamc.responses (module), 27
 - AIOSpamcConnectionException, 17
 - AIOSpamcConnectionFailed, 17
 - args (aiospamc.parser.ParseError attribute), 23
- ## B
- BadRequest, 17
 - BadResponse, 17
 - body (aiospamc.common.RequestResponseBase attribute), 13
 - body (aiospamc.requests.Request attribute), 26
 - body (aiospamc.responses.Response attribute), 27
 - body() (aiospamc.parser.Parser method), 23
- ## C
- CantCreateException, 17
 - check() (aiospamc.client.Client method), 5
 - checkpoint() (in module aiospamc.parser), 26
 - Client (class in aiospamc.client), 4
 - ClientException, 17
 - close() (aiospamc.connections.Connection method), 16
 - code (aiospamc.exceptions.CantCreateException attribute), 17
 - code (aiospamc.exceptions.ConfigException attribute), 18
 - code (aiospamc.exceptions.DataErrorException attribute), 18
 - code (aiospamc.exceptions.InternalSoftwareException attribute), 18

code (aiospamc.exceptions.IOException attribute), 18
code (aiospamc.exceptions.NoHostException attribute), 18
code (aiospamc.exceptions.NoInputException attribute), 18
code (aiospamc.exceptions.NoPermissionException attribute), 18
code (aiospamc.exceptions.NoUserException attribute), 18
code (aiospamc.exceptions.OSErrorException attribute), 18
code (aiospamc.exceptions.OSFileException attribute), 19
code (aiospamc.exceptions.ProtocolException attribute), 19
code (aiospamc.exceptions.TemporaryFailureException attribute), 19
code (aiospamc.exceptions.TimeoutException attribute), 19
code (aiospamc.exceptions.UnavailableException attribute), 19
code (aiospamc.exceptions.UsageException attribute), 19
compress (aiospamc.client.Client attribute), 4
Compress (class in aiospamc.headers), 19
compress_value() (aiospamc.parser.Parser method), 23
ConfigException, 18
connected (aiospamc.connections.Connection attribute), 16
connection (aiospamc.client.Client attribute), 4
Connection (class in aiospamc.connections), 16
connection_string (aiospamc.connections.Connection attribute), 16
connection_string (aiospamc.connections.tcp_connection.TcpConnection attribute), 14
connection_string (aiospamc.connections.unix_connection.UnixConnection attribute), 15
ConnectionManager (class in aiospamc.connections), 17
consume() (aiospamc.parser.Parser method), 23
content_length_value() (aiospamc.parser.Parser method), 24
ContentLength (class in aiospamc.headers), 19
count() (aiospamc.options.ActionOption method), 22
current() (aiospamc.parser.Parser method), 24

D

DataErrorException, 18
delete_header() (aiospamc.common.RequestResponseBase method), 13
delete_header() (aiospamc.requests.Request method), 27
delete_header() (aiospamc.responses.Response method), 28
DidRemove (class in aiospamc.headers), 20
DidSet (class in aiospamc.headers), 20

E

end() (aiospamc.parser.Parser method), 24
EX_CANTCREAT (aiospamc.responses.Status attribute), 28
EX_CONFIG (aiospamc.responses.Status attribute), 28
EX_DATAERR (aiospamc.responses.Status attribute), 28
EX_IOERR (aiospamc.responses.Status attribute), 28
EX_NOHOST (aiospamc.responses.Status attribute), 28
EX_NOINPUT (aiospamc.responses.Status attribute), 28
EX_NOPERM (aiospamc.responses.Status attribute), 28
EX_NOUSER (aiospamc.responses.Status attribute), 28
EX_OK (aiospamc.responses.Status attribute), 28
EX_OSERR (aiospamc.responses.Status attribute), 28
EX_OSFILE (aiospamc.responses.Status attribute), 28
EX_PROTOCOL (aiospamc.responses.Status attribute), 28
EX_SOFTWARE (aiospamc.responses.Status attribute), 28
EX_TEMPFAIL (aiospamc.responses.Status attribute), 28
EX_TIMEOUT (aiospamc.responses.Status attribute), 28
EX_UNAVAILABLE (aiospamc.responses.Status attribute), 28
EX_USAGE (aiospamc.responses.Status attribute), 28

F

field_name() (aiospamc.headers.Compress method), 19
field_name() (aiospamc.headers.ContentLength method), 20
field_name() (aiospamc.headers.DidRemove method), 20
field_name() (aiospamc.headers.DidSet method), 20
field_name() (aiospamc.headers.Header method), 20
field_name() (aiospamc.headers.MessageClass method), 21
field_name() (aiospamc.headers.Remove method), 21
field_name() (aiospamc.headers.Set method), 21
field_name() (aiospamc.headers.Spam method), 21
field_name() (aiospamc.headers.User method), 22
field_name() (aiospamc.headers.XHeader method), 22

G

get_header() (aiospamc.common.RequestResponseBase method), 13
get_header() (aiospamc.requests.Request method), 27
get_header() (aiospamc.responses.Response method), 28

H

ham (aiospamc.options.MessageClassOption attribute), 22
Header (class in aiospamc.headers), 20
header() (aiospamc.parser.Parser method), 24
headers() (aiospamc.client.Client method), 6
headers() (aiospamc.parser.Parser method), 24

host (aiospamc.connections.tcp_connection.TcpConnection attribute), 14
 host (aiospamc.connections.tcp_connection.TcpConnectionManager attribute), 14
 host (aiospamc.connections.unix_connection.UnixConnection attribute), 15

I

index (aiospamc.parser.ParseError attribute), 23
 index() (aiospamc.options.ActionOption method), 22
 InternalSoftwareException, 18
 IOErrorException, 18

L

length (aiospamc.headers.ContentLength attribute), 19
 local (aiospamc.options.ActionOption attribute), 22
 logger (aiospamc.client.Client attribute), 5
 logger (aiospamc.connections.Connection attribute), 16
 loop (aiospamc.client.Client attribute), 4
 loop (aiospamc.connections.Connection attribute), 16
 loop (aiospamc.connections.ConnectionManager attribute), 17
 loop (aiospamc.connections.tcp_connection.TcpConnection attribute), 14
 loop (aiospamc.connections.unix_connection.UnixConnection attribute), 15

M

match() (aiospamc.parser.Parser method), 24
 message (aiospamc.parser.ParseError attribute), 23
 message (aiospamc.responses.Response attribute), 27
 message() (aiospamc.parser.Parser method), 24
 message_class_value() (aiospamc.parser.Parser method), 24
 MessageClass (class in aiospamc.headers), 20
 MessageClassOption (class in aiospamc.options), 22
 method() (aiospamc.parser.Parser method), 24

N

name (aiospamc.headers.User attribute), 22
 name (aiospamc.headers.XHeader attribute), 22
 new_connection() (aiospamc.connections.ConnectionManager method), 17
 new_connection() (aiospamc.connections.tcp_connection.TcpConnectionManager method), 15
 new_connection() (aiospamc.connections.unix_connection.UnixConnectionManager method), 16
 newline() (aiospamc.parser.Parser method), 24
 NoHostException, 18
 NoInputException, 18
 NoPermissionException, 18
 NoUserException, 18

O

open() (aiospamc.connections.Connection method), 16

OSErrException, 18
 OSFileException, 18

P

parse() (in module aiospamc.parser), 26
 ParseError, 23
 Parser (class in aiospamc.parser), 23
 path (aiospamc.connections.unix_connection.UnixConnection attribute), 15
 path (aiospamc.connections.unix_connection.UnixConnectionManager attribute), 16
 ping() (aiospamc.client.Client method), 7
 port (aiospamc.connections.tcp_connection.TcpConnection attribute), 14
 port (aiospamc.connections.tcp_connection.TcpConnectionManager attribute), 15
 process() (aiospamc.client.Client method), 7
 protocol_version (aiospamc.responses.Response attribute), 27
 ProtocolException, 19

R

receive() (aiospamc.connections.Connection method), 17
 remote (aiospamc.options.ActionOption attribute), 22
 Remove (class in aiospamc.headers), 21
 report() (aiospamc.client.Client method), 8
 report_if_spam() (aiospamc.client.Client method), 9
 Request (class in aiospamc.requests), 26
 request() (aiospamc.parser.Parser method), 25
 RequestResponseBase (class in aiospamc.common), 13
 Response (class in aiospamc.responses), 27
 response() (aiospamc.parser.Parser method), 25
 ResponseException, 19
 RFC
 RFC 5322, 30, 32, 34–36

S

score (aiospamc.headers.Spam attribute), 21
 send() (aiospamc.client.Client method), 10
 send() (aiospamc.connections.Connection method), 17
 Set (class in aiospamc.headers), 21
 set_remove_value() (aiospamc.parser.Parser method), 25
 skip() (aiospamc.parser.Parser static method), 25
 spam (aiospamc.options.MessageClassOption attribute), 22
 Spam (class in aiospamc.headers), 21
 spam_value() (aiospamc.parser.Parser method), 25
 spamc_protocol() (aiospamc.parser.Parser method), 25
 spamd_protocol() (aiospamc.parser.Parser method), 25

ssl (aiospamc.connections.tcp_connection.TcpConnection attribute), [14](#)
ssl (aiospamc.connections.tcp_connection.TcpConnectionManager attribute), [15](#)
Status (class in aiospamc.responses), [28](#)
status_code (aiospamc.responses.Response attribute), [27](#)
status_code() (aiospamc.parser.Parser method), [25](#)
symbols() (aiospamc.client.Client method), [11](#)

T

TcpConnection (class in aiospamc.connections.tcp_connection), [14](#)
TcpConnectionManager (class in aiospamc.connections.tcp_connection), [14](#)
tell() (aiospamc.client.Client method), [12](#)
TemporaryFailureException, [19](#)
threshold (aiospamc.headers.Spam attribute), [21](#)
TimeoutException, [19](#)

U

UnavailableException, [19](#)
UnixConnection (class in aiospamc.connections.unix_connection), [15](#)
UnixConnectionManager (class in aiospamc.connections.unix_connection), [16](#)
UsageException, [19](#)
user (aiospamc.client.Client attribute), [4](#)
User (class in aiospamc.headers), [22](#)
user_value() (aiospamc.parser.Parser method), [25](#)

V

value (aiospamc.headers.MessageClass attribute), [20](#)
value (aiospamc.headers.Spam attribute), [21](#)
value (aiospamc.headers.XHeader attribute), [22](#)
verb (aiospamc.requests.Request attribute), [26](#)
version (aiospamc.requests.Request attribute), [26](#)
version() (aiospamc.parser.Parser method), [25](#)

W

whitespace() (aiospamc.parser.Parser method), [26](#)
with_traceback() (aiospamc.parser.ParseError method), [23](#)

X

XHeader (class in aiospamc.headers), [22](#)

Z

zlib (aiospamc.headers.Compress attribute), [19](#)