

---

# **Wrench Documentation**

*Release 2.0.0-beta*

**Dominic Scheirlinck and Contributors**

**Apr 20, 2017**



---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	php-websocket . . . . .	3
<b>2</b>	<b>Installing Wrench</b>	<b>5</b>
2.1	composer . . . . .	5
<b>3</b>	<b>Getting Started</b>	<b>7</b>
3.1	Starting a Server . . . . .	7
<b>4</b>	<b>Performance</b>	<b>9</b>
<b>5</b>	<b>API Documentation</b>	<b>11</b>
5.1	Wrench\Application . . . . .	11
5.1.1	Wrench\Application\Application . . . . .	11
5.1.2	Wrench\Application\EchoApplication . . . . .	11
5.2	Wrench\BasicServer . . . . .	12
5.3	Wrench\Client . . . . .	14
5.4	Wrench\Connection . . . . .	16
5.5	Wrench\ConnectionManager . . . . .	19
5.6	Wrench\Exception . . . . .	21
5.6.1	Wrench\Exception\BadRequestException . . . . .	21
5.6.2	Wrench\Exception\CloseException . . . . .	22
5.6.3	Wrench\Exception\ConnectionException . . . . .	22
5.6.4	Wrench\Exception\Exception . . . . .	23
5.6.5	Wrench\Exception\FrameException . . . . .	24
5.6.6	Wrench\Exception\HandshakeException . . . . .	24
5.6.7	Wrench\Exception\InvalidOriginException . . . . .	25
5.6.8	Wrench\Exception\PayloadException . . . . .	26
5.6.9	Wrench\Exception\RateLimiterException . . . . .	27
5.6.10	Wrench\Exception\SocketException . . . . .	27
5.7	Wrench\Frame . . . . .	28
5.7.1	Wrench\Frame\Frame . . . . .	28
5.7.2	Wrench\Frame\HybiFrame . . . . .	30
5.8	Wrench\Listener . . . . .	32
5.8.1	Wrench\Listener\HandshakeRequestListener . . . . .	32
5.8.2	Wrench\Listener\Listener . . . . .	33
5.8.3	Wrench\Listener\OriginPolicy . . . . .	33

5.8.4	Wrench\Listener\RateLimiter . . . . .	33
5.9	Wrench\Payload . . . . .	35
5.9.1	Wrench\Payload\HybiPayload . . . . .	35
5.9.2	Wrench\Payload\Payload . . . . .	37
5.10	Wrench\Protocol . . . . .	38
5.10.1	Wrench\Protocol\Hybi10Protocol . . . . .	38
5.10.2	Wrench\Protocol\HybiProtocol . . . . .	42
5.10.3	Wrench\Protocol\Protocol . . . . .	46
5.10.4	Wrench\Protocol\Rfc6455Protocol . . . . .	51
5.11	Wrench\Resource . . . . .	55
5.12	Wrench\Server . . . . .	55
5.13	Wrench\Socket . . . . .	57
5.13.1	Wrench\Socket\ClientSocket . . . . .	57
5.13.2	Wrench\Socket\ServerClientSocket . . . . .	60
5.13.3	Wrench\Socket\ServerSocket . . . . .	62
5.13.4	Wrench\Socket\Socket . . . . .	64
5.13.5	Wrench\Socket\UriSocket . . . . .	66
5.14	Wrench\Util . . . . .	69
5.14.1	Wrench\Util\Configurable . . . . .	69
5.14.2	Wrench\Util\Ssl . . . . .	69
<b>6</b>	<b>Authors</b>	<b>71</b>

Wrench is a WebSockets library for PHP 7.1



Wrench is a simple websocket server and client package for PHP 7.1

### **php-websocket**

Wrench was previously known as php-websocket. Why the name change? See [Frequently Asked Questions about the PHP License](#). Also, the namespace WebSocket is too generic; it denotes a common functionality, and may already be in use by application code. The BC break of a new [major version](#) was a good time to introduce this move to best practices.





---

### Installing Wrench

---

The library is PSR-4 compatible, with a vendor name of **Wrench**.

#### **composer**

Wrench is available on Packagist as [wrench/wrench](#).

Here's what it looks like in your `composer.json`

```
{
    ...
    "require": {
        "wrench/wrench": "~3.0"
    }
}
```



## Starting a Server

The first thing you'll want to do to serve WebSockets from PHP is start a WebSockets server. Wrench provides a simple Server class that implements the most recent version of the WebSockets protocol. Subclassing the Server class is encouraged: see `WebSocketBasicServer` for an example.

When you're ready for your server to start responding to requests, call `$server->run()`:

```
use Wrench\BasicServer;

$server = new BasicServer('ws://localhost:8000', array(
    'allowed_origins' => array(
        'mysite.com',
        'mysite.dev.localdomain'
    )
));

// Logging is via PSR3
$logger = new Monolog\Logger('name');
$server->setLogger($logger);

// Register your applications here
$server->registerApplication('echo', new \Wrench\Examples\EchoApplication());
$server->registerApplication('chat', new \My\ChatApplication());

$server->run();
```



## CHAPTER 4

---

### Performance

---

Wrench uses a single-process server, without threads, and blocks while processing data from any client. This means it has little hope of scaling in production.

You might like to use some middleware between your PHP application code and WebSocket clients in production. For example, you might use something like [RabbitMQ's STOMP + WebSockets Plugin](#). In any case, if you're hoping to serve large numbers of clients, you should probably look into one of the evented IO based servers.



## Wrench\Application

### Wrench\Application\Application

#### class **Application**

Wrench Server Application

**onData** (*\$payload*, *\$connection*)

Handle data received from a client

#### Parameters

- **\$payload** (*Payload*) – A payload object, that supports `__toString()`
- **\$connection** (*Connection*) –

### Wrench\Application\EchoApplication

#### class **EchoApplication**

Example application for Wrench: echo server

**onData** (*\$data*, *\$client*)

#### Parameters

- **\$data** –
- **\$client** –

## Wrench\BasicServer

class **BasicServer**

**constant** **EVENT\_SOCKET\_CONNECT**

Events

**property** **rateLimiter**

protected

**property** **originPolicy**

protected

**property** **uri**

protected string

The URI of the server

**property** **options**

protected array

Options

**property** **logger**

protected Closure

A logging callback

The default callback simply prints to stdout. You can pass your own logger in the options array. It should take a string message and string priority as parameters.

**property** **listeners**

protected array<string

Event listeners

Add listeners using the `addListener()` method.

**property** **connectionManager**

protected ConnectionManager

Connection manager

**property** **applications**

protected array<string

Applications

**property** **protocol**

protected Protocol

\_\_\_ **construct** (*\$uri*, *\$options = array()*)

Constructor

### Parameters

- ***\$uri*** (*string*) –
- ***\$options*** (*array*) –

**configure** (*\$options*)

### Parameters

- ***\$options*** –



**configureRateLimiter** ()

**configureOriginPolicy** ()

Configures the origin policy

**addAllowedOrigin** (*\$origin*)

Adds an allowed origin

**Parameters**

- **\$origin** (*array*) –

**configureLogger** ()

Configures the logger

**Returns** void

**configureConnectionManager** ()

Configures the connection manager

**Returns** void

**getConnectionManager** ()

Gets the connection manager

**Returns** WrenchConnectionManager

**getUri** ()

**Returns** string

**setLogger** (*\$logger*)

Sets a logger

**Parameters**

- **\$logger** (*Closure*) –

**Returns** void

**run** ()

Main server loop

**Returns** void This method does not return!

**log** (*\$message*, *\$priority* = 'info')

Logs a message to the server log

The default logger simply prints the message to stdout. You can provide a logging closure. This is useful, for instance, if you've daemonized and closed STDOUT.

**Parameters**

- **\$message** (*string*) – Message to display.
- **\$priority** –

**Returns** void

**notify** (*\$event*, *\$arguments* = *array()*)

Notifies listeners of an event

**Parameters**

- **\$event** (*string*) –
- **\$arguments** (*array*) – Event arguments

**Returns** void

**addListener** (*\$event*, *\$callback*)

Adds a listener

Provide an event (see the `Server::EVENT_*` constants) and a callback closure. Some arguments may be provided to your callback, such as the connection the caused the event.

**Parameters**

- **\$event** (*string*) –
- **\$callback** (*Closure*) –

**Returns** void

**getApplication** (*\$key*)

Returns a server application.

**Parameters**

- **\$key** (*string*) – Name of application.

**Returns** Application The application object.

**registerApplication** (*\$key*, *\$application*)

Adds a new application object to the application storage.

**Parameters**

- **\$key** (*string*) – Name of application.
- **\$application** (*object*) – The application object

**Returns** void

**configureProtocol** ()

Configures the protocol option

## Wrench\Client

**class Client**

Client class

Represents a Wrench client

**constant MAX\_HANDSHAKE\_RESPONSE**

**property uri**

protected

**property origin**

protected

**property socket**

protected

**property headers**

protected array

Request headers

**property protocol**

protected Protocol

Protocol instance

**property options**

protected array

Options

**property connected**

protected boolean

Whether the client is connected

**\_\_construct** (*\$uri*, *\$origin*, *\$options = array()*)

Constructor

**Parameters**

- **\$uri** (*string*) –
- **\$origin** (*string*) – The origin to include in the handshake (required in later versions of the protocol)
- **\$options** –

**configure** (*\$options*)

Configure options

**Parameters**

- **\$options** (*array*) –

**Returns** void**\_\_destruct** ()

Destructor

**addRequestHeader** (*\$name*, *\$value*)

Adds a request header to be included in the initial handshake

For example, to include a Cookie header

**Parameters**

- **\$name** (*string*) –
- **\$value** (*string*) –

**Returns** void**sendData** (*\$data*, *\$type = 'text'*, *\$masked = true*)

Sends data to the socket

**Parameters**

- **\$data** (*string*) –
- **\$type** (*string*) – Payload type
- **\$masked** (*boolean*) –

**Returns** int bytes written**connect** ()

Connect to the Wrench server

**Returns** boolean Whether a new connection was made

**isConnected** ()

Whether the client is currently connected

**Returns** boolean

**disconnect** ()

## Wrench\Connection

**class Connection**

Represents a client connection on the server side

i.e. the *Server* manages a bunch of 'Connection's

**property manager**

protected WrenchConnectionManager

The connection manager

**property socket**

protected Socket

Socket object

Wraps the client connection resource

**property handshaked**

protected boolean

Whether the connection has successfully handshaken

**property application**

protected Application

The application this connection belongs to

**property ip**

protected string

The IP address of the client

**property port**

protected int

The port of the client

**property headers**

protected array

The array of headers included with the original request (like Cookie for example). The headers specific to the web sockets handshaking have been stripped out.

**property queryParams**

protected array

The array of query parameters included in the original request. The array is in the format 'key' => 'value'.

**property id**

protected string|null

Connection ID

**property payload**

protected

The current payload

**property options**

protected array

**property protocol**

protected Protocol

**\_\_construct** (*ConnectionManager \$manager, ServerClientSocket \$socket, \$options = array()*)

Constructor

**Parameters**

- **\$manager** (*ConnectionManager*) –
- **\$socket** (*ServerClientSocket*) –
- **\$options** (*array*) –

**getConnectionManager** ()

Gets the connection manager of this connection

**Returns** WrenchConnectionManager**configure** (*\$options*)**Parameters**

- **\$options** –

**configureClientInformation** ()**configureClientId** ()

Configures the client ID

We hash the client ID to prevent leakage of information if another client happens to get a hold of an ID. The secret *must* be lengthy, and must be kept secret for this to work: otherwise it's trivial to search the space of possible IP addresses/ports (well, if not trivial, at least very fast).

**onData** (*\$data*)

Data receiver

Called by the connection manager when the connection has received data

**Parameters**

- **\$data** (*string*) –

**handshake** (*\$data*)

Performs a websocket handshake

**Parameters**

- **\$data** (*string*) –

**export** (*\$data*)

Returns a string export of the given binary data

**Parameters**

- **\$data** (*string*) –

**Returns** string

**handle** (*\$data*)

Handle data received from the client

The data passed in may belong to several different frames across one or more protocols. It may not even contain a single complete frame. This method manages slotting the data into separate payload objects.

**Parameters**

- **\$data** (*string*) –

**handlePayload** (*Payload \$payload*)

Handle a complete payload received from the client

**Parameters**

- **\$payload** (*Payload*) –

**send** (*\$data*, *\$type = Protocol::TYPE\_TEXT*)

Sends the payload to the connection

**Parameters**

- **\$data** –
- **\$type** (*string*) –

**Returns** boolean

**process** ()

Processes data on the socket

**close** (*\$code = Protocol::CLOSE\_NORMAL*)

Closes the connection according to the WebSocket protocol

**Parameters**

- **\$code** –

**Returns** boolean

**log** (*\$message*, *\$priority = 'info'*)

Logs a message

**Parameters**

- **\$message** (*string*) –
- **\$priority** (*string*) –

**getIp** ()

Gets the IP address of the connection

**Returns** string Usually dotted quad notation

**getPort** ()

Gets the port of the connection

**Returns** int

**getHeaders** ()

Gets the non-web-sockets headers included with the original request

**Returns** array

**getQueryParams** ()

Gets the query parameters included with the original request

**Returns** array

**getId()**  
Gets the connection ID  
**Returns** string

**getSocket()**  
Gets the socket object  
**Returns** SocketServerClientSocket

**getClientApplication()**  
Gets the client application  
**Returns** Application

**configureProtocol()**  
Configures the protocol option

## Wrench\ConnectionManager

class **ConnectionManager**

**property server**  
protected Server

**property socket**  
protected Socket  
Master socket

**property connections**  
protected array<int  
An array of client connections

**property resources**  
protected array<int  
An array of raw socket resources, corresponding to connections, roughly

**property options**  
protected array

**property protocol**  
protected Protocol

**\_\_construct** (*Server \$server, \$options = array()*)  
Constructor

**Parameters**

- **\$server** (*Server*) –
- **\$options** (*array*) –

**count** ()

**configure** (*\$options*)

**Parameters**

- **\$options** –

**getApplicationForPath** (*\$path*)

Gets the application associated with the given path

**Parameters**

- **\$path** (*string*) –

**configureMasterSocket** ()

Configures the main server socket

**listen** ()

Listens on the main socket

**Returns** void

**getAllResources** ()

Gets all resources

**Returns** array<int => resource)

**getConnectionForClientSocket** (*\$socket*)

Returns the Connection associated with the specified socket resource

**Parameters**

- **\$socket** (*resource*) –

**Returns** Connection

**selectAndProcess** ()

Select and process an array of resources

**processMasterSocket** ()

Process events on the master socket (*\$this->socket*)

**Returns** void

**createConnection** (*\$resource*)

Creates a connection from a socket resource

The create connection object is based on the options passed into the constructor ('connection\_class', 'connection\_options'). This connection instance and its associated socket resource are then stored in the manager.

**Parameters**

- **\$resource** (*resource*) – A socket resource

**Returns** Connection

**processClientSocket** (*\$socket*)

Process events on a client socket

**Parameters**

- **\$socket** (*resource*) –

**resourceId** (*\$resource*)

This server makes an explicit assumption: PHP resource types may be cast to a integer. Furthermore, we assume this is bijective. Both seem to be true in most circumstances, but may not be guaranteed.

This method (and *\$this->getResourceId()*) exist to make this assumption explicit.

This is needed on the connection manager as well as on resources

**Parameters**



- **\$resource** (*resource*) –

**getUri** ()  
Gets the connection manager’s listening URI

**Returns** string

**log** (*\$message*, *\$priority = 'info'*)  
Logs a message

**Parameters**

- **\$message** (*string*) –
- **\$priority** (*string*) –

**getServer** ()

**Returns** WrenchServer

**removeConnection** (*Connection \$connection*)  
Removes a connection

**Parameters**

- **\$connection** (*Connection*) –

**configureProtocol** ()  
Configures the protocol option

## Wrench\Exception

### Wrench\Exception\BadRequestException

**class** `BadRequestException`

**property** `message`  
protected

**property** `code`  
protected

**property** `file`  
protected

**property** `line`  
protected

**\_\_construct** (*\$message = null*, *\$code = null*, *\$previous = null*)

**Parameters**

- **\$message** –
- **\$code** –
- **\$previous** (*Exception*) –

**\_\_clone** ()

**getMessage** ()

**getCode** ()

```
getFile ()
getLine ()
getTrace ()
getPrevious ()
getTraceAsString ()
__toString ()
```

## Wrench\Exception\CloseException

**class** `CloseException`

Close connection exception

**property message**  
protected

**property code**  
protected

**property file**  
protected

**property line**  
protected

**\_\_construct** (*\$message = null, \$code = null, \$previous = null*)

### Parameters

- **\$message** –
- **\$code** –
- **\$previous** (`Exception`) –

```
__clone ()
getMessage ()
getCode ()
getFile ()
getLine ()
getTrace ()
getPrevious ()
getTraceAsString ()
__toString ()
```

## Wrench\Exception\ConnectionException

**class** `ConnectionException`

**property message**  
protected

**property code**  
 protected

**property file**  
 protected

**property line**  
 protected

**\_\_clone ()**

**\_\_construct** (*\$message*, *\$code*, *\$previous*)

**Parameters**

- *\$message* –
- *\$code* –
- *\$previous* –

**getMessage ()**

**getCode ()**

**getFile ()**

**getLine ()**

**getTrace ()**

**getPrevious ()**

**getTraceAsString ()**

**\_\_toString ()**

## Wrench\Exception\Exception

class **Exception**

**property message**  
 protected

**property code**  
 protected

**property file**  
 protected

**property line**  
 protected

**\_\_clone ()**

**\_\_construct** (*\$message*, *\$code*, *\$previous*)

**Parameters**

- *\$message* –
- *\$code* –
- *\$previous* –

**getMessage ()**

```
getCode ()
getFile ()
getLine ()
getTrace ()
getPrevious ()
getTraceAsString ()
__toString ()
```

## Wrench\Exception\FrameException

class **FrameException**

```
property message
    protected
property code
    protected
property file
    protected
property line
    protected
__clone ()
__construct ($message, $code, $previous)
```

### Parameters

- ***\$message*** –
- ***\$code*** –
- ***\$previous*** –

```
getMessage ()
getCode ()
getFile ()
getLine ()
getTrace ()
getPrevious ()
getTraceAsString ()
__toString ()
```

## Wrench\Exception\HandshakeException

class **HandshakeException**

**property message**

protected

**property code**

protected

**property file**

protected

**property line**

protected

**\_\_construct** (*\$message = null, \$code = null, \$previous = null*)**Parameters**

- **\$message** –
- **\$code** –
- **\$previous** (*Exception*) –

**\_\_clone** ()**getMessage** ()**getCode** ()**getFile** ()**getLine** ()**getTrace** ()**getPrevious** ()**getTraceAsString** ()**\_\_toString** ()

## Wrench\Exception\InvalidOriginException

**class InvalidOriginException**

Invalid origin exception

**property message**

protected

**property code**

protected

**property file**

protected

**property line**

protected

**\_\_construct** (*\$message = null, \$code = null, \$previous = null*)**Parameters**

- **\$message** –
- **\$code** –
- **\$previous** (*Exception*) –

```
__clone ()
getMessage ()
getCode ()
getFile ()
getLine ()
getTrace ()
getPrevious ()
getTraceAsString ()
__toString ()
```

## Wrench\Exception\PayloadException

class PayloadException

```
property message
    protected
property code
    protected
property file
    protected
property line
    protected
__clone ()
__construct ($message, $code, $previous)
```

### Parameters

- *\$message* –
- *\$code* –
- *\$previous* –

```
getMessage ()
getCode ()
getFile ()
getLine ()
getTrace ()
getPrevious ()
getTraceAsString ()
__toString ()
```

## Wrench\Exception\RateLimiterException

class `RateLimiterException`

**property message**

protected

**property code**

protected

**property file**

protected

**property line**

protected

**\_\_construct** (*\$message = null, \$code = null, \$previous = null*)

### Parameters

- **\$message** –
- **\$code** –
- **\$previous** (`Exception`) –

**\_\_clone** ()

**getMessage** ()

**getCode** ()

**getFile** ()

**getLine** ()

**getTrace** ()

**getPrevious** ()

**getTraceAsString** ()

**\_\_toString** ()

## Wrench\Exception\SocketException

class `SocketException`

**property message**

protected

**property code**

protected

**property file**

protected

**property line**

protected

**\_\_clone** ()

**\_\_construct** (*\$message, \$code, \$previous*)

### Parameters

- `$message` –
- `$code` –
- `$previous` –

`getMessage()`

`getCode()`

`getFile()`

`getLine()`

`getTrace()`

`getPrevious()`

`getTraceAsString()`

`__toString()`

## Wrench\Frame

### Wrench\Frame\Frame

#### class **Frame**

Represents a WebSocket frame

#### **property length**

protected int

The frame data length

#### **property type**

protected int

The type of this payload

#### **property buffer**

protected string

The buffer

May not be a complete payload, because this frame may still be receiving data. See

#### **property payload**

protected string

The enclosed frame payload

May not be a complete payload, because this frame might indicate a continuation frame. See `isFinal()` versus `isComplete()`

#### **getLength()**

Gets the length of the payload

**Returns** int

#### **encode** (*\$data*, *\$type = Protocol::TYPE\_TEXT*, *\$masked = false*)

Resets the frame and encodes the given data into it



**Parameters**

- **\$data** (*string*) –
- **\$type** (*int*) –
- **\$masked** (*boolean*) –

**Returns** Frame**isFinal** ()

Whether the frame is the final one in a continuation

**Returns** boolean**getType** ()**Returns** int**decodeFramePayloadFromBuffer** ()

Decodes a frame payload from the buffer

**Returns** void**getExpectedBufferLength** ()

Gets the expected length of the buffer once all the data has been received

**Returns** int**isComplete** ()

Whether the frame is complete

**Returns** boolean**receiveData** (*\$data*)

Receives data into the frame

**Parameters**

- **\$data** –

**getRemainingData** ()

Gets the remaining number of bytes before this frame will be complete

**Returns** number**isWaitingForData** ()

Whether this frame is waiting for more data

**Returns** boolean**getFramePayload** ()

Gets the contents of the frame payload

The frame must be complete to call this method.

**Returns** string**getFrameBuffer** ()

Gets the contents of the frame buffer

This is the encoded value, received into the frame with receiveData().

**Returns** string binary**getBufferLength** ()

Gets the expected length of the frame payload

Returns int

## Wrench\Frame\HybiFrame

class HybiFrame

**property masked**

protected boolean

Whether the payload is masked

**property mask**

protected string

Masking key

**property offset\_payload**

protected int

Byte offsets

**property offset\_mask**

protected

**property length**

protected int

The frame data length

**property type**

protected int

The type of this payload

**property buffer**

protected string

The buffer

May not be a complete payload, because this frame may still be receiving data. See

**property payload**

protected string

The enclosed frame payload

May not be a complete payload, because this frame might indicate a continuation frame. See isFinal() versus isComplete()

**encode** (*\$payload*, *\$type* = *Protocol::TYPE\_TEXT*, *\$masked* = *false*)

**Parameters**

- *\$payload* –
- *\$type* –
- *\$masked* –

**mask** (*\$payload*)

Masks/Unmasks the frame

**Parameters**

- *\$payload* (*string*) –

**Returns** string

**unmask** (*\$payload*)

Masks a payload

**Parameters**

- **\$payload** (*string*) –

**Returns** string

**receiveData** (*\$data*)

**Parameters**

- **\$data** –

**getMask** ()

Gets the mask

**Returns** string

**generateMask** ()

Generates a suitable masking key

**Returns** string

**isMasked** ()

Whether the frame is masked

**Returns** boolean

**getExpectedBufferLength** ()

**getPayloadOffset** ()

Gets the offset of the payload in the frame

**Returns** int

**getMaskOffset** ()

Gets the offset in the frame to the masking bytes

**Returns** int

**getLength** ()

**getInitialLength** ()

Gets the initial length value, stored in the first length byte

This determines how the rest of the length value is parsed out of the frame.

**Returns** int

**getLengthSize** ()

Returns the byte size of the length part of the frame

Not including the initial 7 bit part

**Returns** int

**getMaskSize** ()

Returns the byte size of the mask part of the frame

**Returns** int

**decodeFramePayloadFromBuffer** ()

**isFinal** ()

**getType ()**

**isComplete ()**

Whether the frame is complete

**Returns** boolean

**getRemainingData ()**

Gets the remaining number of bytes before this frame will be complete

**Returns** number

**isWaitingForData ()**

Whether this frame is waiting for more data

**Returns** boolean

**getFramePayload ()**

Gets the contents of the frame payload

The frame must be complete to call this method.

**Returns** string

**getFrameBuffer ()**

Gets the contents of the frame buffer

This is the encoded value, received into the frame with `receiveData()`.

**Returns** string binary

**getBufferLength ()**

Gets the expected length of the frame payload

**Returns** int

## Wrench\Listener

### Wrench\Listener\HandshakeRequestListener

**class HandshakeRequestListener**

**onHandshakeRequest** (*Connection \$connection, \$path, \$origin, \$key, \$extensions*)

Handshake request listener

**Parameters**

- **\$connection** (*Connection*) –
- **\$path** (*string*) –
- **\$origin** (*string*) –
- **\$key** (*string*) –
- **\$extensions** (*array*) –

## Wrench\Listener\Listener

class `Listener`

`listen` (*Server* `$server`)

**Parameters**

- `$server` (*Server*) –

## Wrench\Listener\OriginPolicy

class `OriginPolicy`

property `allowed`

protected

`__construct` (*\$allowed*)

**Parameters**

- `$allowed` –

`onHandshakeRequest` (*Connection* `$connection`, *\$path*, *\$origin*, *\$key*, *\$extensions*)

Handshake request listener

Closes the connection on handshake from an origin that isn't allowed

**Parameters**

- `$connection` (*Connection*) –
- `$path` (*string*) –
- `$origin` (*string*) –
- `$key` (*string*) –
- `$extensions` (*array*) –

`isAllowed` (*\$origin*)

Whether the specified origin is allowed under this policy

**Parameters**

- `$origin` (*string*) –

**Returns** boolean

`listen` (*Server* `$server`)

**Parameters**

- `$server` (*Server*) –

## Wrench\Listener\RateLimiter

class `RateLimiter`

**property server**

protected Server

The server being limited

**property ips**

protected array<int>

Connection counts per IP address

**property requests**

protected array<array<int>>

Request tokens per IP address

**property options**

protected array

**property protocol**

protected Protocol

— **construct** (*\$options = array()*)

Constructor

**Parameters**

- **\$options** (*array*) –

**configure** (*\$options*)

**Parameters**

- **\$options** (*array*) –

**listen** (*Server \$server*)

**Parameters**

- **\$server** (*Server*) –

**onSocketConnect** (*\$socket, \$connection*)

Event listener

**Parameters**

- **\$socket** (*resource*) –
- **\$connection** (*Connection*) –

**onSocketDisconnect** (*\$socket, \$connection*)

Event listener

**Parameters**

- **\$socket** (*resource*) –
- **\$connection** (*Connection*) –

**onClientData** (*\$socket, \$connection*)

Event listener

**Parameters**

- **\$socket** (*resource*) –
- **\$connection** (*Connection*) –

**checkConnections** (*\$connection*)

Idempotent

**Parameters**

- **\$connection** (*Connection*) –

**checkConnectionsPerIp** (*\$connection*)

NOT idempotent, call once per connection

**Parameters**

- **\$connection** (*Connection*) –

**releaseConnection** (*\$connection*)

NOT idempotent, call once per disconnection

**Parameters**

- **\$connection** (*Connection*) –

**checkRequestsPerMinute** (*\$connection*)

NOT idempotent, call once per data

**Parameters**

- **\$connection** (*Connection*) –

**limit** (*\$connection*, *\$limit*)

Limits the given connection

**Parameters**

- **\$connection** (*Connection*) –
- **\$limit** (*string*) – Reason

**log** (*\$message*, *\$priority* = 'info')

Logger

**Parameters**

- **\$message** (*string*) –
- **\$priority** (*string*) –

**configureProtocol** ()

Configures the protocol option

## Wrench\Payload

### Wrench\Payload\HybiPayload

**class HybiPayload**

Gets a HyBi payload

**property frames**

protected array<Frame>

A payload may consist of one or more frames

**getFrame** ()

**getCurrentFrame** ()

Gets the current frame for the payload

**Returns** mixed

**getReceivingFrame** ()

Gets the frame into which data should be received

**Returns** Frame

**isComplete** ()

Whether the payload is complete

**Returns** boolean

**encode** (*\$data*, *\$type = Protocol::TYPE\_TEXT*, *\$masked = false*)

Encodes a payload

**Parameters**

- *\$data* (*string*) –
- *\$type* (*int*) –
- *\$masked* (*boolean*) –

**Returns** Payload

**getRemainingData** ()

Gets the number of remaining bytes before this payload will be complete

May return 0 (no more bytes required) or null (unknown number of bytes required).

**Returns** number|NULL

**isWaitingForData** ()

Whether this payload is waiting for more data

**Returns** boolean

**sendToSocket** (*Socket \$socket*)

**Parameters**

- *\$socket* (*Socket*) –

**Returns** boolean

**receiveData** (*\$data*)

Receive raw data into the payload

**Parameters**

- *\$data* (*string*) –

**Returns** void

**getPayload** ()

**Returns** string

**\_\_toString** ()

**Returns** string

**getType** ()

Gets the type of the payload

The type of a payload is taken from its first frame



**Returns** int

## Wrench\Payload\Payload

### class **Payload**

Payload class

Represents a WebSocket protocol payload, which may be made up of multiple frames.

#### **property frames**

protected array<Frame>

A payload may consist of one or more frames

#### **getCurrentFrame ()**

Gets the current frame for the payload

**Returns** mixed

#### **getReceivingFrame ()**

Gets the frame into which data should be received

**Returns** Frame

#### **getFrame ()**

Get a frame object

**Returns** Frame

#### **isComplete ()**

Whether the payload is complete

**Returns** boolean

#### **encode (\$data, \$type = Protocol::TYPE\_TEXT, \$masked = false)**

Encodes a payload

##### **Parameters**

- **\$data** (*string*) –
- **\$type** (*int*) –
- **\$masked** (*boolean*) –

**Returns** Payload

#### **getRemainingData ()**

Gets the number of remaining bytes before this payload will be complete

May return 0 (no more bytes required) or null (unknown number of bytes required).

**Returns** number|NULL

#### **isWaitingForData ()**

Whether this payload is waiting for more data

**Returns** boolean

#### **sendToSocket (Socket \$socket)**

##### **Parameters**

- **\$socket** (*Socket*) –

**Returns** boolean

**receiveData** (*\$data*)

Receive raw data into the payload

**Parameters**

- **\$data** (*string*) –

**Returns** void

**getPayload** ()

**Returns** string

**\_\_toString** ()

**Returns** string

**getType** ()

Gets the type of the payload

The type of a payload is taken from its first frame

**Returns** int

## Wrench\Protocol

### Wrench\Protocol\Hybi10Protocol

**class Hybi10Protocol**

<http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-10>

**constant SCHEME\_WEBSOCKET**

Relevant schemes

**constant HEADER\_HOST**

HTTP headers

**constant HTTP\_SWITCHING\_PROTOCOLS**

HTTP error statuses

**constant CLOSE\_NORMAL**

Close statuses

**constant TYPE\_CONTINUATION**

Frame types

**%x0 denotes a continuation frame** %x1 denotes a text frame %x2 denotes a binary frame %x3-7 are reserved for further non-control frames %x8 denotes a connection close %x9 denotes a ping %xA denotes a pong %xB-F are reserved for further control frames

**constant MAGIC\_GUID**

Magic GUID

Used in the WebSocket accept header

**constant UPGRADE\_VALUE**

**The request MUST contain an `Upgrade` header field whose value MUST include the “websocket” keyword.**

**constant CONNECTION\_VALUE**

The request **MUST** contain a **Connection** header field whose value **MUST** include the “Upgrade” token.

**constant REQUEST\_LINE\_FORMAT**

Request line format

**constant REQUEST\_LINE\_REGEX**

Request line regex

Used for parsing requested path

**constant RESPONSE\_LINE\_FORMAT**

Response line format

**constant HEADER\_LINE\_FORMAT**

Header line format

**property schemes**

protected array<string>

Valid schemes

**property closeReasons**

array<int>

Close status codes

**property frameTypes**

array<string>

Frame types

**property httpResponses**

array<int>

HTTP errors

**getVersion ()**

**acceptsVersion (\$version)**

This is our most recent protocol class

**Parameters**

- **\$version** –

**getPayload ()**

**generateKey ()**

Generates a key suitable for use in the protocol

This base implementation returns a 16-byte (128 bit) random key as a binary string.

**Returns** string

**getRequestHandshake (\$uri, \$key, \$origin, \$headers = array())**

Gets request handshake string

The leading line from the client follows the Request-Line format. The leading line from the server follows the Status-Line format. The Request-Line and Status-Line productions are defined in [RFC2616].

An unordered set of header fields comes after the leading line in both cases. The meaning of these header fields is specified in Section 4 of this document. Additional header fields may also be present, such as cookies [RFC6265]. The format and parsing of headers is as defined in [RFC2616].

**Parameters**

- **\$uri** (*string*) – WebSocket URI, e.g. ws://example.org:8000/chat
- **\$key** (*string*) – 16 byte binary string key
- **\$origin** (*string*) – Origin of the request
- **\$headers** –

**Returns** string

**getResponseHandshake** (*\$key*, *\$headers = array()*)

Gets a handshake response body

**Parameters**

- **\$key** (*string*) –
- **\$headers** (*array*) –

**getResponseError** (*\$e*, *\$headers = array()*)

Gets a response to an error in the handshake

**Parameters**

- **\$e** (*int* | *Exception*) – Exception or HTTP error
- **\$headers** (*array*) –

**getHttpResponse** (*\$status*, *\$headers = array()*)

Gets an HTTP response

**Parameters**

- **\$status** (*int*) –
- **\$headers** (*array*) –

**validateResponseHandshake** (*\$response*, *\$key*)

**Parameters**

- **\$response** (*unknown\_type*) –
- **\$key** (*unknown\_type*) –

**Returns** boolean

**getEncodedHash** (*\$key*)

Gets an encoded hash for a key

**Parameters**

- **\$key** (*string*) –

**Returns** string

**validateRequestHandshake** (*\$request*)

Validates a request handshake

**Parameters**

- **\$request** (*string*) –

**getCloseFrame** (*\$e*)

Gets a suitable WebSocket close frame

**Parameters**

- **\$e** (*Exception* | *int*) –

**validateUri** (*\$uri*)

Validates a WebSocket URI

**Parameters**

- **\$uri** (*string*) –

**Returns** array(string \$scheme, string \$host, int \$port, string \$path)**validateSocketUri** (*\$uri*)

Validates a socket URI

**Parameters**

- **\$uri** (*string*) –

**Returns** array(string \$scheme, string \$host, string \$port)**validateOriginUri** (*\$origin*)

Validates an origin URI

**Parameters**

- **\$origin** (*string*) –

**Returns** string**validateRequestLine** (*\$line*)

Validates a request line

**Parameters**

- **\$line** (*string*) –

**getAcceptValue** (*\$encoded\_key*)

Gets the expected accept value for a handshake response

Note that the protocol calls for the base64 encoded value to be hashed, not the original 16 byte random key.

**Parameters**

- **\$encoded\_key** –

**getHeaders** (*\$response*, *\$request\_line = null*)

Gets the headers from a full response

**Parameters**

- **\$response** (*string*) –
- **\$request\_line** –

**Returns** array()**getRequestHeaders** (*\$response*)

Gets request headers

**Parameters**

- **\$response** (*string*) –

**Returns** array<string, array<string>> The request line, and an array of headers**validateScheme** (*\$scheme*)

Validates a scheme

**Parameters**

- **\$scheme** (*string*) –

**Returns** string Underlying scheme

**getDefaultRequestHeaders** (*\$host*, *\$key*, *\$origin*)

Gets the default request headers

**Parameters**

- **\$host** (*string*) –
- **\$key** (*string*) –
- **\$origin** (*string*) –

**Returns** multitype:unknown string NULL

**getSuccessResponseHeaders** (*\$key*)

Gets the default response headers

**Parameters**

- **\$key** (*string*) –

**getPort** (*\$scheme*)

Gets the default port for a scheme

By default, the WebSocket Protocol uses port 80 for regular WebSocket connections and port 443 for WebSocket connections tunneled over Transport Layer Security

**Parameters**

- **\$scheme** –

**Returns** int

## Wrench\Protocol\HybiProtocol

class **HybiProtocol**

**constant SCHEME\_WEBSOCKET**

Relevant schemes

**constant HEADER\_HOST**

HTTP headers

**constant HTTP\_SWITCHING\_PROTOCOLS**

HTTP error statuses

**constant CLOSE\_NORMAL**

Close statuses

**constant TYPE\_CONTINUATION**

Frame types

**%x0 denotes a continuation frame** %x1 denotes a text frame %x2 denotes a binary frame %x3-7 are reserved for further non-control frames %x8 denotes a connection close %x9 denotes a ping %xA denotes a pong %xB-F are reserved for further control frames

**constant MAGIC\_GUID**

Magic GUID

Used in the WebSocket accept header

**constant UPGRADE\_VALUE**

The request **MUST** contain an **Upgrade** header field whose value **MUST** include the “websocket” keyword.

**constant CONNECTION\_VALUE**

The request **MUST** contain a **Connection** header field whose value **MUST** include the “Upgrade” token.

**constant REQUEST\_LINE\_FORMAT**

Request line format

**constant REQUEST\_LINE\_REGEX**

Request line regex

Used for parsing requested path

**constant RESPONSE\_LINE\_FORMAT**

Response line format

**constant HEADER\_LINE\_FORMAT**

Header line format

**property schemes**

protected array<string>

Valid schemes

**property closeReasons**

array<int

Close status codes

**property frameTypes**

array<string

Frame types

**property httpResponses**

array<int

HTTP errors

**getPayload()****getVersion()**

Gets a version number

**acceptsVersion(\$version)**

Subclasses should implement this method and return a boolean to the given version string, as to whether they would like to accept requests from user agents that specify that version.

**Parameters**

- **\$version** –

**Returns** boolean

**generateKey()**

Generates a key suitable for use in the protocol

This base implementation returns a 16-byte (128 bit) random key as a binary string.

**Returns** string

**getRequestHandshake** (*\$uri*, *\$key*, *\$origin*, *\$headers = array()*)

Gets request handshake string

The leading line from the client follows the Request-Line format. The leading line from the server follows the Status-Line format. The Request-Line and Status-Line productions are defined in [RFC2616].

An unordered set of header fields comes after the leading line in both cases. The meaning of these header fields is specified in Section 4 of this document. Additional header fields may also be present, such as cookies [RFC6265]. The format and parsing of headers is as defined in [RFC2616].

**Parameters**

- **\$uri** (*string*) – WebSocket URI, e.g. `ws://example.org:8000/chat`
- **\$key** (*string*) – 16 byte binary string key
- **\$origin** (*string*) – Origin of the request
- **\$headers** –

**Returns** string

**getResponseHandshake** (*\$key*, *\$headers = array()*)

Gets a handshake response body

**Parameters**

- **\$key** (*string*) –
- **\$headers** (*array*) –

**getResponseError** (*\$e*, *\$headers = array()*)

Gets a response to an error in the handshake

**Parameters**

- **\$e** (*int* | *Exception*) – Exception or HTTP error
- **\$headers** (*array*) –

**getHttpResponse** (*\$status*, *\$headers = array()*)

Gets an HTTP response

**Parameters**

- **\$status** (*int*) –
- **\$headers** (*array*) –

**validateResponseHandshake** (*\$response*, *\$key*)

**Parameters**

- **\$response** (*unknown\_type*) –
- **\$key** (*unknown\_type*) –

**Returns** boolean

**getEncodedHash** (*\$key*)

Gets an encoded hash for a key

**Parameters**

- **\$key** (*string*) –

**Returns** string



**validateRequestHandshake** (*\$request*)

Validates a request handshake

**Parameters**

- **\$request** (*string*) –

**getCloseFrame** (*\$e*)

Gets a suitable WebSocket close frame

**Parameters**

- **\$e** (*Exception|int*) –

**validateUri** (*\$uri*)

Validates a WebSocket URI

**Parameters**

- **\$uri** (*string*) –

**Returns** array(string \$scheme, string \$host, int \$port, string \$path)

**validateSocketUri** (*\$uri*)

Validates a socket URI

**Parameters**

- **\$uri** (*string*) –

**Returns** array(string \$scheme, string \$host, string \$port)

**validateOriginUri** (*\$origin*)

Validates an origin URI

**Parameters**

- **\$origin** (*string*) –

**Returns** string

**validateRequestLine** (*\$line*)

Validates a request line

**Parameters**

- **\$line** (*string*) –

**getAcceptValue** (*\$encoded\_key*)

Gets the expected accept value for a handshake response

Note that the protocol calls for the base64 encoded value to be hashed, not the original 16 byte random key.

**Parameters**

- **\$encoded\_key** –

**getHeaders** (*\$response, \$request\_line = null*)

Gets the headers from a full response

**Parameters**

- **\$response** (*string*) –
- **\$request\_line** –

**Returns** array()

**getRequestHeaders** (*\$response*)

Gets request headers

**Parameters**

- **\$response** (*string*) –

**Returns** array<string, array<string>> The request line, and an array of headers

**validateScheme** (*\$scheme*)

Validates a scheme

**Parameters**

- **\$scheme** (*string*) –

**Returns** string Underlying scheme

**getDefaultRequestHeaders** (*\$host, \$key, \$origin*)

Gets the default request headers

**Parameters**

- **\$host** (*string*) –
- **\$key** (*string*) –
- **\$origin** (*string*) –

**Returns** multitype:unknown string NULL

**getSuccessResponseHeaders** (*\$key*)

Gets the default response headers

**Parameters**

- **\$key** (*string*) –

**getPort** (*\$scheme*)

Gets the default port for a scheme

By default, the WebSocket Protocol uses port 80 for regular WebSocket connections and port 443 for WebSocket connections tunneled over Transport Layer Security

**Parameters**

- **\$scheme** –

**Returns** int

## Wrench\Protocol\Protocol

**class Protocol**

Definitions and implementation helpers for the Wrenchs protocol

Based on RFC 6455: <http://tools.ietf.org/html/rfc6455>

**constant SCHEME\_WEBSOCKET**

Relevant schemes

**constant HEADER\_HOST**

HTTP headers

**constant HTTP\_SWITCHING\_PROTOCOLS**

HTTP error statuses

**constant CLOSE\_NORMAL**

Close statuses

**constant TYPE\_CONTINUATION**

Frame types

**%x0 denotes a continuation frame** %x1 denotes a text frame %x2 denotes a binary frame %x3-7 are reserved for further non-control frames %x8 denotes a connection close %x9 denotes a ping %xA denotes a pong %xB-F are reserved for further control frames

**constant MAGIC\_GUID**

Magic GUID

Used in the WebSocket accept header

**constant UPGRADE\_VALUE**

The request **MUST** contain an **Upgrade** header field whose value **MUST** include the “websocket” keyword.

**constant CONNECTION\_VALUE**

The request **MUST** contain a **Connection** header field whose value **MUST** include the “Upgrade” token.

**constant REQUEST\_LINE\_FORMAT**

Request line format

**constant REQUEST\_LINE\_REGEX**

Request line regex

Used for parsing requested path

**constant RESPONSE\_LINE\_FORMAT**

Response line format

**constant HEADER\_LINE\_FORMAT**

Header line format

**property schemes**

protected array<string>

Valid schemes

**property closeReasons**

array<int>

Close status codes

**property frameTypes**

array<string>

Frame types

**property httpResponses**

array<int>

HTTP errors

**getVersion ()**

Gets a version number

**acceptsVersion (\$version)**

Subclasses should implement this method and return a boolean to the given version string, as to whether they would like to accept requests from user agents that specify that version.

**Parameters**

- **\$version** –

**Returns** boolean

**getPayload()**

Gets a payload instance, suitable for use in decoding/encoding protocol frames

**Returns** Payload

**generateKey()**

Generates a key suitable for use in the protocol

This base implementation returns a 16-byte (128 bit) random key as a binary string.

**Returns** string

**getRequestHandshake(\$uri, \$key, \$origin, \$headers = array())**

Gets request handshake string

The leading line from the client follows the Request-Line format. The leading line from the server follows the Status-Line format. The Request-Line and Status-Line productions are defined in [RFC2616].

An unordered set of header fields comes after the leading line in both cases. The meaning of these header fields is specified in Section 4 of this document. Additional header fields may also be present, such as cookies [RFC6265]. The format and parsing of headers is as defined in [RFC2616].

**Parameters**

- **\$uri** (*string*) – WebSocket URI, e.g. ws://example.org:8000/chat
- **\$key** (*string*) – 16 byte binary string key
- **\$origin** (*string*) – Origin of the request
- **\$headers** –

**Returns** string

**getResponseHandshake(\$key, \$headers = array())**

Gets a handshake response body

**Parameters**

- **\$key** (*string*) –
- **\$headers** (*array*) –

**getResponseError(\$e, \$headers = array())**

Gets a response to an error in the handshake

**Parameters**

- **\$e** (*int* | *Exception*) – Exception or HTTP error
- **\$headers** (*array*) –

**getHttpResponse(\$status, \$headers = array())**

Gets an HTTP response

**Parameters**

- **\$status** (*int*) –
- **\$headers** (*array*) –

**validateResponseHandshake(\$response, \$key)**

**Parameters**

- **\$response** (*unknown\_type*) –
- **\$key** (*unknown\_type*) –

**Returns** boolean**getEncodedHash** (*\$key*)

Gets an encoded hash for a key

**Parameters**

- **\$key** (*string*) –

**Returns** string**validateRequestHandshake** (*\$request*)

Validates a request handshake

**Parameters**

- **\$request** (*string*) –

**getCloseFrame** (*\$e*)

Gets a suitable WebSocket close frame

**Parameters**

- **\$e** (*Exception|int*) –

**validateUri** (*\$uri*)

Validates a WebSocket URI

**Parameters**

- **\$uri** (*string*) –

**Returns** array(string \$scheme, string \$host, int \$port, string \$path)**validateSocketUri** (*\$uri*)

Validates a socket URI

**Parameters**

- **\$uri** (*string*) –

**Returns** array(string \$scheme, string \$host, string \$port)**validateOriginUri** (*\$origin*)

Validates an origin URI

**Parameters**

- **\$origin** (*string*) –

**Returns** string**validateRequestLine** (*\$line*)

Validates a request line

**Parameters**

- **\$line** (*string*) –

**getAcceptValue** (*\$encoded\_key*)

Gets the expected accept value for a handshake response

Note that the protocol calls for the base64 encoded value to be hashed, not the original 16 byte random key.

**Parameters**

- **\$encoded\_key** –

**getHeaders** (*\$response*, *\$request\_line = null*)

Gets the headers from a full response

**Parameters**

- **\$response** (*string*) –
- **\$request\_line** –

**Returns** array()

**getRequestHeaders** (*\$response*)

Gets request headers

**Parameters**

- **\$response** (*string*) –

**Returns** array<string, array<string>> The request line, and an array of headers

**validateScheme** (*\$scheme*)

Validates a scheme

**Parameters**

- **\$scheme** (*string*) –

**Returns** string Underlying scheme

**getDefaultRequestHeaders** (*\$host*, *\$key*, *\$origin*)

Gets the default request headers

**Parameters**

- **\$host** (*string*) –
- **\$key** (*string*) –
- **\$origin** (*string*) –

**Returns** multitype:unknown string NULL

**getSuccessResponseHeaders** (*\$key*)

Gets the default response headers

**Parameters**

- **\$key** (*string*) –

**getPort** (*\$scheme*)

Gets the default port for a scheme

By default, the WebSocket Protocol uses port 80 for regular WebSocket connections and port 443 for WebSocket connections tunneled over Transport Layer Security

**Parameters**

- **\$scheme** –

**Returns** int

## Wrench\Protocol\Rfc6455Protocol

### class Rfc6455Protocol

This is the version of websockets used by Chrome versions 17 through 19.

#### constant SCHEME\_WEBSOCKET

Relevant schemes

#### constant HEADER\_HOST

HTTP headers

#### constant HTTP\_SWITCHING\_PROTOCOLS

HTTP error statuses

#### constant CLOSE\_NORMAL

Close statuses

#### constant TYPE\_CONTINUATION

Frame types

**%x0 denotes a continuation frame** %x1 denotes a text frame %x2 denotes a binary frame %x3-7 are reserved for further non-control frames %x8 denotes a connection close %x9 denotes a ping %xA denotes a pong %xB-F are reserved for further control frames

#### constant MAGIC\_GUID

Magic GUID

Used in the WebSocket accept header

#### constant UPGRADE\_VALUE

The request **MUST** contain an **Upgrade** header field whose value **MUST** include the “websocket” keyword.

#### constant CONNECTION\_VALUE

The request **MUST** contain a **Connection** header field whose value **MUST** include the “Upgrade” token.

#### constant REQUEST\_LINE\_FORMAT

Request line format

#### constant REQUEST\_LINE\_REGEX

Request line regex

Used for parsing requested path

#### constant RESPONSE\_LINE\_FORMAT

Response line format

#### constant HEADER\_LINE\_FORMAT

Header line format

#### property schemes

protected array<string>

Valid schemes

#### property closeReasons

array<int>

Close status codes

**property frameTypes**

array<string

Frame types

**property httpResponses**

array<int

HTTP errors

**getVersion ()**

**acceptsVersion (\$version)**

This is our most recent protocol class

**Parameters**

- **\$version** –

**getPayload ()**

**generateKey ()**

Generates a key suitable for use in the protocol

This base implementation returns a 16-byte (128 bit) random key as a binary string.

**Returns** string

**getRequestHandshake (\$uri, \$key, \$origin, \$headers = array())**

Gets request handshake string

The leading line from the client follows the Request-Line format. The leading line from the server follows the Status-Line format. The Request-Line and Status-Line productions are defined in [RFC2616].

An unordered set of header fields comes after the leading line in both cases. The meaning of these header fields is specified in Section 4 of this document. Additional header fields may also be present, such as cookies [RFC6265]. The format and parsing of headers is as defined in [RFC2616].

**Parameters**

- **\$uri** (*string*) – WebSocket URI, e.g. ws://example.org:8000/chat
- **\$key** (*string*) – 16 byte binary string key
- **\$origin** (*string*) – Origin of the request
- **\$headers** –

**Returns** string

**getResponseHandshake (\$key, \$headers = array())**

Gets a handshake response body

**Parameters**

- **\$key** (*string*) –
- **\$headers** (*array*) –

**getResponseError (\$e, \$headers = array())**

Gets a response to an error in the handshake

**Parameters**

- **\$e** (*int | Exception*) – Exception or HTTP error
- **\$headers** (*array*) –



**getHttpResponse** (*\$status*, *\$headers = array()*)

Gets an HTTP response

**Parameters**

- **\$status** (*int*) –
- **\$headers** (*array*) –

**validateResponseHandshake** (*\$response*, *\$key*)

**Parameters**

- **\$response** (*unknown\_type*) –
- **\$key** (*unknown\_type*) –

**Returns** boolean

**getEncodedHash** (*\$key*)

Gets an encoded hash for a key

**Parameters**

- **\$key** (*string*) –

**Returns** string

**validateRequestHandshake** (*\$request*)

Validates a request handshake

**Parameters**

- **\$request** (*string*) –

**getCloseFrame** (*\$e*)

Gets a suitable WebSocket close frame

**Parameters**

- **\$e** (*Exception|int*) –

**validateUri** (*\$uri*)

Validates a WebSocket URI

**Parameters**

- **\$uri** (*string*) –

**Returns** array(string \$scheme, string \$host, int \$port, string \$path)

**validateSocketUri** (*\$uri*)

Validates a socket URI

**Parameters**

- **\$uri** (*string*) –

**Returns** array(string \$scheme, string \$host, string \$port)

**validateOriginUri** (*\$origin*)

Validates an origin URI

**Parameters**

- **\$origin** (*string*) –

**Returns** string

**validateRequestLine** (*\$line*)

Validates a request line

**Parameters**

- **\$line** (*string*) –

**getAcceptValue** (*\$encoded\_key*)

Gets the expected accept value for a handshake response

Note that the protocol calls for the base64 encoded value to be hashed, not the original 16 byte random key.

**Parameters**

- **\$encoded\_key** –

**getHeaders** (*\$response*, *\$request\_line = null*)

Gets the headers from a full response

**Parameters**

- **\$response** (*string*) –
- **\$request\_line** –

**Returns** array()

**getRequestHeaders** (*\$response*)

Gets request headers

**Parameters**

- **\$response** (*string*) –

**Returns** array<string, array<string>> The request line, and an array of headers

**validateScheme** (*\$scheme*)

Validates a scheme

**Parameters**

- **\$scheme** (*string*) –

**Returns** string Underlying scheme

**getDefaultRequestHeaders** (*\$host*, *\$key*, *\$origin*)

Gets the default request headers

**Parameters**

- **\$host** (*string*) –
- **\$key** (*string*) –
- **\$origin** (*string*) –

**Returns** multitype:unknown string NULL

**getSuccessResponseHeaders** (*\$key*)

Gets the default response headers

**Parameters**

- **\$key** (*string*) –

**getPort** (*\$scheme*)

Gets the default port for a scheme

By default, the WebSocket Protocol uses port 80 for regular WebSocket connections and port 443 for WebSocket connections tunneled over Transport Layer Security

**Parameters**

- **\$scheme** –

**Returns** int

## Wrench\Resource

**class Resource**

Resource interface

**getResourceId** ()

**getResource** ()

## Wrench\Server

**class Server**

WebSocket server

The server extends socket, which provides the master socket resource. This resource is listened to, and an array of clients managed.

**constant EVENT\_SOCKET\_CONNECT**

Events

**property uri**

protected string

The URI of the server

**property options**

protected array

Options

**property logger**

protected Closure

A logging callback

The default callback simply prints to stdout. You can pass your own logger in the options array. It should take a string message and string priority as parameters.

**property listeners**

protected array<string

Event listeners

Add listeners using the addListener() method.

**property connectionManager**

protected ConnectionManager

Connection manager

**property applications**

protected array<string>

Applications

**property protocol**

protected Protocol

**\_\_construct** (*\$uri*, *\$options = array()*)

Constructor

**Parameters**

- **\$uri** (*string*) – WebSocket URI, e.g. ws://localhost:8000/, path will be ignored
- **\$options** (*array*) – (optional) See configure

**configure** (*\$options*)

Configure options

Options include - *socket\_class* => The socket class to use, defaults to ServerSocket - *socket\_options* => An array of socket options - *logger* => Closure(*\$message*, *\$priority = 'info'*), used for logging

**Parameters**

- **\$options** (*array*) –

**Returns** void

**configureLogger** ()

Configures the logger

**Returns** void

**configureConnectionManager** ()

Configures the connection manager

**Returns** void

**getConnectionManager** ()

Gets the connection manager

**Returns** WrenchConnectionManager

**getUri** ()

**Returns** string

**setLogger** (*\$logger*)

Sets a logger

**Parameters**

- **\$logger** (*Closure*) –

**Returns** void

**run** ()

Main server loop

**Returns** void This method does not return!

**log** (*\$message*, *\$priority = 'info'*)

Logs a message to the server log

The default logger simply prints the message to stdout. You can provide a logging closure. This is useful, for instance, if you've daemonized and closed STDOUT.

**Parameters**

- **\$message** (*string*) – Message to display.
- **\$priority** –

**Returns** void

**notify** (*\$event*, *\$arguments = array()*)  
Notifies listeners of an event

**Parameters**

- **\$event** (*string*) –
- **\$arguments** (*array*) – Event arguments

**Returns** void

**addListener** (*\$event*, *\$callback*)  
Adds a listener

Provide an event (see the `Server::EVENT_*` constants) and a callback closure. Some arguments may be provided to your callback, such as the connection the caused the event.

**Parameters**

- **\$event** (*string*) –
- **\$callback** (*Closure*) –

**Returns** void

**getApplication** (*\$key*)  
Returns a server application.

**Parameters**

- **\$key** (*string*) – Name of application.

**Returns** Application The application object.

**registerApplication** (*\$key*, *\$application*)  
Adds a new application object to the application storage.

**Parameters**

- **\$key** (*string*) – Name of application.
- **\$application** (*object*) – The application object

**Returns** void

**configureProtocol** ()  
Configures the protocol option

## Wrench\Socket

### Wrench\Socket\ClientSocket

class `ClientSocket`

**Options:**

- `timeout_connect => int, seconds, default 2`

**constant TIMEOUT\_CONNECT**

Default connection timeout

**constant TIMEOUT\_SOCKET**

Default timeout for socket operations (reads, writes)

**constant DEFAULT\_RECEIVE\_LENGTH**

**constant NAME\_PART\_IP**

Socket name parts

**property scheme**

protected

**property host**

protected

**property port**

protected

**property socket**

protected resource

**property context**

protected

Stream context

**property connected**

protected boolean

Whether the socket is connected to a server

Note, the connection may not be ready to use, but the socket is connected at least. See `$handshaked`, and other properties in subclasses.

**property firstRead**

protected boolean

Whether the current read is the first one to the socket

**property name**

protected string

The socket name according to `stream_socket_get_name`

**property options**

protected array

**property protocol**

protected Protocol

**configure** (*\$options*)

**Parameters**

- *\$options* –

**connect** ()

Connects to the given socket

**reconnect** ()

**getSocketStreamContextOptions** ()

**getSslStreamContextOptions** ()

**\_\_construct** (*\$uri*, *\$options = array()*)  
URI Socket constructor

**Parameters**

- **\$uri** (*string*) – WebSocket URI, e.g. `ws://example.org:8000/chat`
- **\$options** –

**getUri** ()  
Gets the canonical/normalized URI for this socket

**Returns** string

**getName** ()

**getHost** ()  
Gets the host name

**getPort** ()

**getStreamContext** (*\$listen = false*)  
Gets a stream context

**Parameters**

- **\$listen** –

**getNamePart** (*\$name*, *\$part*)  
Gets part of the name of the socket

PHP seems to return IPV6 address/port combos like this: `:::1:1234`, where `:::1` is the address and `1234` the port. So, the `part` number here is either the last `:` delimited section (the port) or all the other sections (the whole initial part, the address).

**Parameters**

- **\$name** (*string*) – (from `$this->getName()` usually)
- **\$part** –

**Returns** string

**getIp** ()  
Gets the IP address of the socket

**Returns** string

**getLastError** ()  
Get the last error that occurred on the socket

**Returns** intlstring

**isConnected** ()  
Whether the socket is currently connected

**Returns** boolean

**disconnect** ()  
Disconnect the socket

**Returns** void

**getResource** ()

**getResourceId** ()

**send** (*\$data*)

**Parameters**

- **\$data** (*unknown\_type*) –

**Returns** boolean|int The number of bytes sent or false on error

**receive** (*\$length = self::DEFAULT\_RECEIVE\_LENGTH*)

Recieve data from the socket

**Parameters**

- **\$length** (*int*) –

**Returns** string

**configureProtocol** ()

Configures the protocol option

## Wrench\Socket\ServerClientSocket

class **ServerClientSocket**

**constant TIMEOUT\_SOCKET**

Default timeout for socket operations (reads, writes)

**constant DEFAULT\_RECEIVE\_LENGTH**

**constant NAME\_PART\_IP**

Socket name parts

**property socket**

protected resource

**property context**

protected

Stream context

**property connected**

protected boolean

Whether the socket is connected to a server

Note, the connection may not be ready to use, but the socket is connected at least. See \$handshaked, and other properties in subclasses.

**property firstRead**

protected boolean

Whether the current read is the first one to the socket

**property name**

protected string

The socket name according to stream\_socket\_get\_name

**property options**

protected array

**property protocol**

protected Protocol



**\_\_construct** (*\$accepted\_socket*, *\$options = array()*)

Constructor

A server client socket is accepted from a listening socket, so there's no need to call `->connect()` or whatnot.

**Parameters**

- **\$accepted\_socket** (*resource*) –
- **\$options** (*array*) –

**configure** (*\$options*)

Configure options

Options include - `timeout_connect => int, seconds, default 2` - `timeout_socket => int, seconds, default 5`

**Parameters**

- **\$options** (*array*) –

**Returns** void

**getName** ()

Gets the name of the socket

**getNamePart** (*\$name*, *\$part*)

Gets part of the name of the socket

PHP seems to return IPV6 address/port combos like this: `::1:1234`, where `::1` is the address and `1234` the port. So, the part number here is either the last `:` delimited section (the port) or all the other sections (the whole initial part, the address).

**Parameters**

- **\$name** (*string*) – (from `$this->getName()` usually)
- **\$part** –

**Returns** string

**getIp** ()

Gets the IP address of the socket

**Returns** string

**getPort** ()

Gets the port of the socket

**Returns** int

**getLastError** ()

Get the last error that occurred on the socket

**Returns** intlstring

**isConnected** ()

Whether the socket is currently connected

**Returns** boolean

**disconnect** ()

Disconnect the socket

**Returns** void

**getResource** ()

**getResourceId** ()

**send** (*\$data*)

**Parameters**

- **\$data** (*unknown\_type*) –

**Returns** boolean|int The number of bytes sent or false on error

**receive** (*\$length = self::DEFAULT\_RECEIVE\_LENGTH*)

Receive data from the socket

**Parameters**

- **\$length** (*int*) –

**Returns** string

**configureProtocol** ()

Configures the protocol option

## Wrench\Socket\ServerSocket

**class ServerSocket**

Server socket

Used for a server's "master" socket that binds to the configured interface and listens

**constant TIMEOUT\_SOCKET**

Default timeout for socket operations (reads, writes)

**constant DEFAULT\_RECEIVE\_LENGTH**

**constant NAME\_PART\_IP**

Socket name parts

**property listening**

protected boolean

Whether the socket is listening

**property scheme**

protected

**property host**

protected

**property port**

protected

**property socket**

protected resource

**property context**

protected

Stream context

**property connected**

protected boolean

Whether the socket is connected to a server

Note, the connection may not be ready to use, but the socket is connected at least. See \$handshaked, and other properties in subclasses.

**property firstRead**

protected boolean

Whether the current read is the first one to the socket

**property name**

protected string

The socket name according to stream\_socket\_get\_name

**property options**

protected array

**property protocol**

protected Protocol

**configure** (*\$options*)**Parameters**

- **\$options** –

**listen** ()

Listens

**accept** ()

Accepts a new connection on the socket

**Returns** resource**getSocketStreamContextOptions** ()**getSslStreamContextOptions** ()**\_\_construct** (*\$uri*, *\$options = array()*)

URI Socket constructor

**Parameters**

- **\$uri** (*string*) – WebSocket URI, e.g. ws://example.org:8000/chat
- **\$options** –

**getUri** ()

Gets the canonical/normalized URI for this socket

**Returns** string**getName** ()**getHost** ()

Gets the host name

**getPort** ()**getStreamContext** (*\$listen = false*)

Gets a stream context

**Parameters**

- **\$listen** –

**getNamePart** (*\$name*, *\$part*)

Gets part of the name of the socket

PHP seems to return IPV6 address/port combos like this: `::1:1234`, where `::1` is the address and `1234` the port. So, the part number here is either the last `:` delimited section (the port) or all the other sections (the whole initial part, the address).

### Parameters

- **\$name** (*string*) – (from `$this->getName()` usually)
- **\$part** –

**Returns** string

### `getIp()`

Gets the IP address of the socket

**Returns** string

### `getLastError()`

Get the last error that occurred on the socket

**Returns** `int|string`

### `isConnected()`

Whether the socket is currently connected

**Returns** boolean

### `disconnect()`

Disconnect the socket

**Returns** void

### `getResource()`

### `getResourceId()`

### `send($data)`

#### Parameters

- **\$data** (*unknown\_type*) –

**Returns** `boolean|int` The number of bytes sent or false on error

### `receive($length = self::DEFAULT_RECEIVE_LENGTH)`

Receive data from the socket

#### Parameters

- **\$length** (*int*) –

**Returns** string

### `configureProtocol()`

Configures the protocol option

## Wrench\Socket\Socket

### class `Socket`

Socket class

Implements low level logic for connecting, serving, reading to, and writing from `WebSocket` connections using PHP's streams.

Unlike in previous versions of this library, a `Socket` instance now represents a single underlying socket resource. It's designed to be used by aggregation, rather than inheritance.

**constant TIMEOUT\_SOCKET**

Default timeout for socket operations (reads, writes)

**constant DEFAULT\_RECEIVE\_LENGTH****constant NAME\_PART\_IP**

Socket name parts

**property socket**

protected resource

**property context**

protected

Stream context

**property connected**

protected boolean

Whether the socket is connected to a server

Note, the connection may not be ready to use, but the socket is connected at least. See `$handshaked`, and other properties in subclasses.

**property firstRead**

protected boolean

Whether the current read is the first one to the socket

**property name**

protected string

The socket name according to `stream_socket_get_name`

**property options**

protected array

**property protocol**

protected Protocol

**configure** (*\$options*)

Configure options

Options include - `timeout_connect => int, seconds, default 2` - `timeout_socket => int, seconds, default 5`

**Parameters**

- **\$options** (*array*) –

**Returns** void

**getName** ()

Gets the name of the socket

**getNamePart** (*\$name*, *\$part*)

Gets part of the name of the socket

PHP seems to return IPV6 address/port combos like this: `:::1:1234`, where `:::1` is the address and `1234` the port. So, the part number here is either the last `:` delimited section (the port) or all the other sections (the whole initial part, the address).

**Parameters**

- **\$name** (*string*) – (from `$this->getName()` usually)
- **\$part** –

**Returns** string

**getIp** ()

Gets the IP address of the socket

**Returns** string

**getPort** ()

Gets the port of the socket

**Returns** int

**getLastError** ()

Get the last error that occurred on the socket

**Returns** int|string

**isConnected** ()

Whether the socket is currently connected

**Returns** boolean

**disconnect** ()

Disconnect the socket

**Returns** void

**getResource** ()

**getResourceId** ()

**send** (*\$data*)

**Parameters**

- **\$data** (*unknown\_type*) –

**Returns** boolean|int The number of bytes sent or false on error

**receive** (*\$length = self::DEFAULT\_RECEIVE\_LENGTH*)

Receive data from the socket

**Parameters**

- **\$length** (*int*) –

**Returns** string

**\_\_construct** (*\$options = array()*)

Configurable constructor

**Parameters**

- **\$options** –

**configureProtocol** ()

Configures the protocol option

## Wrench\Socket\UriSocket

class **UriSocket**

**constant TIMEOUT\_SOCKET**

Default timeout for socket operations (reads, writes)

**constant** `DEFAULT_RECEIVE_LENGTH`

**constant** `NAME_PART_IP`

Socket name parts

**property** `scheme`

protected

**property** `host`

protected

**property** `port`

protected

**property** `socket`

protected resource

**property** `context`

protected

Stream context

**property** `connected`

protected boolean

Whether the socket is connected to a server

Note, the connection may not be ready to use, but the socket is connected at least. See `$handshaked`, and other properties in subclasses.

**property** `firstRead`

protected boolean

Whether the current read is the first one to the socket

**property** `name`

protected string

The socket name according to `stream_socket_get_name`

**property** `options`

protected array

**property** `protocol`

protected Protocol

**\_\_construct** (*\$uri*, *\$options* = *array()*)

URI Socket constructor

#### Parameters

- *\$uri* (*string*) – WebSocket URI, e.g. `ws://example.org:8000/chat`
- *\$options* –

**getUri** ()

Gets the canonical/normalized URI for this socket

**Returns** string

**getName** ()

**getHost** ()

Gets the host name

**getPort** ()

**getStreamContext** (*\$listen = false*)

Gets a stream context

**Parameters**

- **\$listen** –

**getSocketStreamContextOptions** ()

Returns an array of socket stream context options

See <http://php.net/manual/en/context.socket.php>

**Returns** array

**getSslStreamContextOptions** ()

Returns an array of ssl stream context options

See <http://php.net/manual/en/context.ssl.php>

**Returns** array

**configure** (*\$options*)

Configure options

Options include - `timeout_connect => int, seconds, default 2` - `timeout_socket => int, seconds, default 5`

**Parameters**

- **\$options** (*array*) –

**Returns** void

**getNamePart** (*\$name, \$part*)

Gets part of the name of the socket

PHP seems to return IPV6 address/port combos like this: `:::1:1234`, where `:::1` is the address and `1234` the port. So, the `part` number here is either the last `:` delimited section (the port) or all the other sections (the whole initial part, the address).

**Parameters**

- **\$name** (*string*) – (from `$this->getName()` usually)
- **\$part** –

**Returns** string

**getIp** ()

Gets the IP address of the socket

**Returns** string

**getLastError** ()

Get the last error that occurred on the socket

**Returns** intlstring

**isConnected** ()

Whether the socket is currently connected

**Returns** boolean

**disconnect** ()

Disconnect the socket

**Returns** void

**getResource** ()



**getResourceId()**

**send(\$data)**

**Parameters**

- **\$data** (*unknown\_type*) –

**Returns** boolean|int The number of bytes sent or false on error

**receive(\$length = self::DEFAULT\_RECEIVE\_LENGTH)**

Recieve data from the socket

**Parameters**

- **\$length** (*int*) –

**Returns** string

**configureProtocol()**

Configures the protocol option

## Wrench\Util

### Wrench\Util\Configurable

**class Configurable**

Configurable base class

**property options**

protected array

**property protocol**

protected Protocol

**\_\_construct(\$options = array())**

Configurable constructor

**Parameters**

- **\$options** –

**configure(\$options)**

Configures the options

**Parameters**

- **\$options** (*array*) –

**configureProtocol()**

Configures the protocol option

### Wrench\Util\Ssl

**class Ssl**

**generatePemFile(\$pem\_file, \$pem\_passphrase, \$country\_name, \$state\_or\_province\_name, \$locality\_name, \$organization\_name, \$organizational\_unit\_name, \$common\_name, \$email\_address)**

Generates a new PEM File given the informations

### Parameters

- **\$pem\_file** (*string*) – the path of the PEM file to create
- **\$pem\_passphrase** (*string*) – the passphrase to protect the PEM file or if you don't want to use a passphrase
- **\$country\_name** (*string*) – the country code of the new PEM file. e.g.: EN
- **\$state\_or\_province\_name** (*string*) – the state or province name of the new PEM file
- **\$locality\_name** (*string*) – the name of the locality
- **\$organization\_name** –
- **\$organizational\_unit\_name** –
- **\$common\_name** –
- **\$email\_address** (*string*) – the email address

## CHAPTER 6

---

### Authors

---

The original maintainer and author was [@nicokaiser](#). Plentiful improvements were contributed by [@lemmingzshadow](#) and [@mzhack](#). Parts of the Socket class were written by Moritz Wutz. The server is licensed under the WTFPL, a free software compatible license.



## Symbols

\_\_clone() (BadRequestException method), **21**  
 \_\_clone() (CloseException method), **22**  
 \_\_clone() (ConnectionException method), **23**  
 \_\_clone() (Exception method), **23**  
 \_\_clone() (FrameException method), **24**  
 \_\_clone() (HandshakeException method), **25**  
 \_\_clone() (InvalidOriginException method), **25**  
 \_\_clone() (PayloadException method), **26**  
 \_\_clone() (RateLimiterException method), **27**  
 \_\_clone() (SocketException method), **27**  
 \_\_construct() (BadRequestException method), **21**  
 \_\_construct() (BasicServer method), **12**  
 \_\_construct() (Client method), **15**  
 \_\_construct() (ClientSocket method), **58**  
 \_\_construct() (CloseException method), **22**  
 \_\_construct() (Configurable method), **69**  
 \_\_construct() (Connection method), **17**  
 \_\_construct() (ConnectionException method), **23**  
 \_\_construct() (ConnectionManager method), **19**  
 \_\_construct() (Exception method), **23**  
 \_\_construct() (FrameException method), **24**  
 \_\_construct() (HandshakeException method), **25**  
 \_\_construct() (InvalidOriginException method), **25**  
 \_\_construct() (OriginPolicy method), **33**  
 \_\_construct() (PayloadException method), **26**  
 \_\_construct() (RateLimiter method), **34**  
 \_\_construct() (RateLimiterException method), **27**  
 \_\_construct() (Server method), **56**  
 \_\_construct() (ServerClientSocket method), **60**  
 \_\_construct() (ServerSocket method), **63**  
 \_\_construct() (Socket method), **66**  
 \_\_construct() (SocketException method), **27**  
 \_\_construct() (UriSocket method), **67**  
 \_\_destruct() (Client method), **15**  
 \_\_toString() (BadRequestException method), **22**  
 \_\_toString() (CloseException method), **22**  
 \_\_toString() (ConnectionException method), **23**  
 \_\_toString() (Exception method), **24**

\_\_toString() (FrameException method), **24**  
 \_\_toString() (HandshakeException method), **25**  
 \_\_toString() (HybiPayload method), **36**  
 \_\_toString() (InvalidOriginException method), **26**  
 \_\_toString() (Payload method), **38**  
 \_\_toString() (PayloadException method), **26**  
 \_\_toString() (RateLimiterException method), **27**  
 \_\_toString() (SocketException method), **28**

## A

accept() (ServerSocket method), **63**  
 acceptsVersion() (Hybi10Protocol method), **39**  
 acceptsVersion() (HybiProtocol method), **43**  
 acceptsVersion() (Protocol method), **47**  
 acceptsVersion() (Rfc6455Protocol method), **52**  
 addAllowedOrigin() (BasicServer method), **13**  
 addListener() (BasicServer method), **14**  
 addListener() (Server method), **57**  
 addRequestHeader() (Client method), **15**  
 allowed (OriginPolicy property), **33**  
 Application (class), **11**  
 application (Connection property), **16**  
 applications (BasicServer property), **12**  
 applications (Server property), **56**

## B

BadRequestException (class), **21**  
 BasicServer (class), **12**  
 BasicServer::EVENT\_SOCKET\_CONNECT (class constant), **12**  
 buffer (Frame property), **28**  
 buffer (HybiFrame property), **30**

## C

checkConnections() (RateLimiter method), **34**  
 checkConnectionsPerIp() (RateLimiter method), **35**  
 checkRequestsPerMinute() (RateLimiter method), **35**  
 Client (class), **14**  
 Client::MAX\_HANDSHAKE\_RESPONSE (class constant), **14**

ClientSocket (class), [57](#)  
ClientSocket::DEFAULT\_RECEIVE\_LENGTH (class constant), [58](#)  
ClientSocket::NAME\_PART\_IP (class constant), [58](#)  
ClientSocket::TIMEOUT\_CONNECT (class constant), [57](#)  
ClientSocket::TIMEOUT\_SOCKET (class constant), [58](#)  
close() (Connection method), [18](#)  
CloseException (class), [22](#)  
closeReasons (Hybi10Protocol property), [39](#)  
closeReasons (HybiProtocol property), [43](#)  
closeReasons (Protocol property), [47](#)  
closeReasons (Rfc6455Protocol property), [51](#)  
code (BadRequestException property), [21](#)  
code (CloseException property), [22](#)  
code (ConnectionException property), [22](#)  
code (Exception property), [23](#)  
code (FrameException property), [24](#)  
code (HandshakeException property), [25](#)  
code (InvalidOriginException property), [25](#)  
code (PayloadException property), [26](#)  
code (RateLimiterException property), [27](#)  
code (SocketException property), [27](#)  
Configurable (class), [69](#)  
configure() (BasicServer method), [12](#)  
configure() (Client method), [15](#)  
configure() (ClientSocket method), [58](#)  
configure() (Configurable method), [69](#)  
configure() (Connection method), [17](#)  
configure() (ConnectionManager method), [19](#)  
configure() (RateLimiter method), [34](#)  
configure() (Server method), [56](#)  
configure() (ServerClientSocket method), [61](#)  
configure() (ServerSocket method), [63](#)  
configure() (Socket method), [65](#)  
configure() (UriSocket method), [68](#)  
configureClientId() (Connection method), [17](#)  
configureClientInformation() (Connection method), [17](#)  
configureConnectionManager() (BasicServer method), [13](#)  
configureConnectionManager() (Server method), [56](#)  
configureLogger() (BasicServer method), [13](#)  
configureLogger() (Server method), [56](#)  
configureMasterSocket() (ConnectionManager method), [20](#)  
configureOriginPolicy() (BasicServer method), [13](#)  
configureProtocol() (BasicServer method), [14](#)  
configureProtocol() (ClientSocket method), [60](#)  
configureProtocol() (Configurable method), [69](#)  
configureProtocol() (Connection method), [19](#)  
configureProtocol() (ConnectionManager method), [21](#)  
configureProtocol() (RateLimiter method), [35](#)  
configureProtocol() (Server method), [57](#)  
configureProtocol() (ServerClientSocket method), [62](#)  
configureProtocol() (ServerSocket method), [64](#)

configureProtocol() (Socket method), [66](#)  
configureProtocol() (UriSocket method), [69](#)  
configureRateLimiter() (BasicServer method), [12](#)  
connect() (Client method), [15](#)  
connect() (ClientSocket method), [58](#)  
connected (Client property), [15](#)  
connected (ClientSocket property), [58](#)  
connected (ServerClientSocket property), [60](#)  
connected (ServerSocket property), [62](#)  
connected (Socket property), [65](#)  
connected (UriSocket property), [67](#)  
Connection (class), [16](#)  
ConnectionException (class), [22](#)  
connectionManager (BasicServer property), [12](#)  
ConnectionManager (class), [19](#)  
connectionManager (Server property), [55](#)  
connections (ConnectionManager property), [19](#)  
context (ClientSocket property), [58](#)  
context (ServerClientSocket property), [60](#)  
context (ServerSocket property), [62](#)  
context (Socket property), [65](#)  
context (UriSocket property), [67](#)  
count() (ConnectionManager method), [19](#)  
createConnection() (ConnectionManager method), [20](#)

## D

decodeFramePayloadFromBuffer() (Frame method), [29](#)  
decodeFramePayloadFromBuffer() (HybiFrame method), [31](#)  
disconnect() (Client method), [16](#)  
disconnect() (ClientSocket method), [59](#)  
disconnect() (ServerClientSocket method), [61](#)  
disconnect() (ServerSocket method), [64](#)  
disconnect() (Socket method), [66](#)  
disconnect() (UriSocket method), [68](#)

## E

EchoApplication (class), [11](#)  
encode() (Frame method), [28](#)  
encode() (HybiFrame method), [30](#)  
encode() (HybiPayload method), [36](#)  
encode() (Payload method), [37](#)  
Exception (class), [23](#)  
export() (Connection method), [17](#)

## F

file (BadRequestException property), [21](#)  
file (CloseException property), [22](#)  
file (ConnectionException property), [23](#)  
file (Exception property), [23](#)  
file (FrameException property), [24](#)  
file (HandshakeException property), [25](#)  
file (InvalidOriginException property), [25](#)  
file (PayloadException property), [26](#)

file (RateLimiterException property), [27](#)  
 file (SocketException property), [27](#)  
 firstRead (ClientSocket property), [58](#)  
 firstRead (ServerClientSocket property), [60](#)  
 firstRead (ServerSocket property), [62](#)  
 firstRead (Socket property), [65](#)  
 firstRead (UriSocket property), [67](#)  
 Frame (class), [28](#)  
 FrameException (class), [24](#)  
 frames (HybiPayload property), [35](#)  
 frames (Payload property), [37](#)  
 frameTypes (Hybi10Protocol property), [39](#)  
 frameTypes (HybiProtocol property), [43](#)  
 frameTypes (Protocol property), [47](#)  
 frameTypes (Rfc6455Protocol property), [51](#)

## G

generateKey() (Hybi10Protocol method), [39](#)  
 generateKey() (HybiProtocol method), [43](#)  
 generateKey() (Protocol method), [48](#)  
 generateKey() (Rfc6455Protocol method), [52](#)  
 generateMask() (HybiFrame method), [31](#)  
 generatePemFile() (Ssl method), [69](#)  
 getAcceptValue() (Hybi10Protocol method), [41](#)  
 getAcceptValue() (HybiProtocol method), [45](#)  
 getAcceptValue() (Protocol method), [49](#)  
 getAcceptValue() (Rfc6455Protocol method), [54](#)  
 getAllResources() (ConnectionManager method), [20](#)  
 getApplication() (BasicServer method), [14](#)  
 getApplication() (Server method), [57](#)  
 getApplicationForPath() (ConnectionManager method),  
[19](#)  
 getBufferLength() (Frame method), [29](#)  
 getBufferLength() (HybiFrame method), [32](#)  
 getClientApplication() (Connection method), [19](#)  
 getCloseFrame() (Hybi10Protocol method), [40](#)  
 getCloseFrame() (HybiProtocol method), [45](#)  
 getCloseFrame() (Protocol method), [49](#)  
 getCloseFrame() (Rfc6455Protocol method), [53](#)  
 getCode() (BadRequestException method), [21](#)  
 getCode() (CloseException method), [22](#)  
 getCode() (ConnectionException method), [23](#)  
 getCode() (Exception method), [24](#)  
 getCode() (FrameException method), [24](#)  
 getCode() (HandshakeException method), [25](#)  
 getCode() (InvalidOriginException method), [26](#)  
 getCode() (PayloadException method), [26](#)  
 getCode() (RateLimiterException method), [27](#)  
 getCode() (SocketException method), [28](#)  
 getConnectionForClientSocket() (ConnectionManager  
 method), [20](#)  
 getConnectionManager() (BasicServer method), [13](#)  
 getConnectionManager() (Connection method), [17](#)  
 getConnectionManager() (Server method), [56](#)

getCurrentFrame() (HybiPayload method), [35](#)  
 getCurrentFrame() (Payload method), [37](#)  
 getDefaultRequestHeaders() (Hybi10Protocol method),  
[42](#)  
 getDefaultRequestHeaders() (HybiProtocol method), [46](#)  
 getDefaultRequestHeaders() (Protocol method), [50](#)  
 getDefaultRequestHeaders() (Rfc6455Protocol method),  
[54](#)  
 getEncodedHash() (Hybi10Protocol method), [40](#)  
 getEncodedHash() (HybiProtocol method), [44](#)  
 getEncodedHash() (Protocol method), [49](#)  
 getEncodedHash() (Rfc6455Protocol method), [53](#)  
 getExpectedBufferLength() (Frame method), [29](#)  
 getExpectedBufferLength() (HybiFrame method), [31](#)  
 getFile() (BadRequestException method), [21](#)  
 getFile() (CloseException method), [22](#)  
 getFile() (ConnectionException method), [23](#)  
 getFile() (Exception method), [24](#)  
 getFile() (FrameException method), [24](#)  
 getFile() (HandshakeException method), [25](#)  
 getFile() (InvalidOriginException method), [26](#)  
 getFile() (PayloadException method), [26](#)  
 getFile() (RateLimiterException method), [27](#)  
 getFile() (SocketException method), [28](#)  
 getFrame() (HybiPayload method), [35](#)  
 getFrame() (Payload method), [37](#)  
 getFrameBuffer() (Frame method), [29](#)  
 getFrameBuffer() (HybiFrame method), [32](#)  
 getFramePayload() (Frame method), [29](#)  
 getFramePayload() (HybiFrame method), [32](#)  
 getHeaders() (Connection method), [18](#)  
 getHeaders() (Hybi10Protocol method), [41](#)  
 getHeaders() (HybiProtocol method), [45](#)  
 getHeaders() (Protocol method), [50](#)  
 getHeaders() (Rfc6455Protocol method), [54](#)  
 getHost() (ClientSocket method), [59](#)  
 getHost() (ServerSocket method), [63](#)  
 getHost() (UriSocket method), [67](#)  
 getHttpResponse() (Hybi10Protocol method), [40](#)  
 getHttpResponse() (HybiProtocol method), [44](#)  
 getHttpResponse() (Protocol method), [48](#)  
 getHttpResponse() (Rfc6455Protocol method), [52](#)  
 getId() (Connection method), [18](#)  
 getInitialLength() (HybiFrame method), [31](#)  
 getIp() (ClientSocket method), [59](#)  
 getIp() (Connection method), [18](#)  
 getIp() (ServerClientSocket method), [61](#)  
 getIp() (ServerSocket method), [64](#)  
 getIp() (Socket method), [66](#)  
 getIp() (UriSocket method), [68](#)  
 getLastError() (ClientSocket method), [59](#)  
 getLastError() (ServerClientSocket method), [61](#)  
 getLastError() (ServerSocket method), [64](#)  
 getLastError() (Socket method), [66](#)

getLastError() (UriSocket method), [68](#)  
getLength() (Frame method), [28](#)  
getLength() (HybiFrame method), [31](#)  
getLengthSize() (HybiFrame method), [31](#)  
getLine() (BadRequestException method), [22](#)  
getLine() (CloseException method), [22](#)  
getLine() (ConnectionException method), [23](#)  
getLine() (Exception method), [24](#)  
getLine() (FrameException method), [24](#)  
getLine() (HandshakeException method), [25](#)  
getLine() (InvalidOriginException method), [26](#)  
getLine() (PayloadException method), [26](#)  
getLine() (RateLimiterException method), [27](#)  
getLine() (SocketException method), [28](#)  
getMask() (HybiFrame method), [31](#)  
getMaskOffset() (HybiFrame method), [31](#)  
getMaskSize() (HybiFrame method), [31](#)  
getMessage() (BadRequestException method), [21](#)  
getMessage() (CloseException method), [22](#)  
getMessage() (ConnectionException method), [23](#)  
getMessage() (Exception method), [23](#)  
getMessage() (FrameException method), [24](#)  
getMessage() (HandshakeException method), [25](#)  
getMessage() (InvalidOriginException method), [26](#)  
getMessage() (PayloadException method), [26](#)  
getMessage() (RateLimiterException method), [27](#)  
getMessage() (SocketException method), [28](#)  
getName() (ClientSocket method), [59](#)  
getName() (ServerClientSocket method), [61](#)  
getName() (ServerSocket method), [63](#)  
getName() (Socket method), [65](#)  
getName() (UriSocket method), [67](#)  
getNamePart() (ClientSocket method), [59](#)  
getNamePart() (ServerClientSocket method), [61](#)  
getNamePart() (ServerSocket method), [63](#)  
getNamePart() (Socket method), [65](#)  
getNamePart() (UriSocket method), [68](#)  
getPayload() (Hybi10Protocol method), [39](#)  
getPayload() (HybiPayload method), [36](#)  
getPayload() (HybiProtocol method), [43](#)  
getPayload() (Payload method), [38](#)  
getPayload() (Protocol method), [48](#)  
getPayload() (Rfc6455Protocol method), [52](#)  
getPayloadOffset() (HybiFrame method), [31](#)  
getPort() (ClientSocket method), [59](#)  
getPort() (Connection method), [18](#)  
getPort() (Hybi10Protocol method), [42](#)  
getPort() (HybiProtocol method), [46](#)  
getPort() (Protocol method), [50](#)  
getPort() (Rfc6455Protocol method), [54](#)  
getPort() (ServerClientSocket method), [61](#)  
getPort() (ServerSocket method), [63](#)  
getPort() (Socket method), [66](#)  
getPort() (UriSocket method), [67](#)  
getPrevious() (BadRequestException method), [22](#)  
getPrevious() (CloseException method), [22](#)  
getPrevious() (ConnectionException method), [23](#)  
getPrevious() (Exception method), [24](#)  
getPrevious() (FrameException method), [24](#)  
getPrevious() (HandshakeException method), [25](#)  
getPrevious() (InvalidOriginException method), [26](#)  
getPrevious() (PayloadException method), [26](#)  
getPrevious() (RateLimiterException method), [27](#)  
getPrevious() (SocketException method), [28](#)  
getQueryParams() (Connection method), [18](#)  
getReceivingFrame() (HybiPayload method), [36](#)  
getReceivingFrame() (Payload method), [37](#)  
getRemainingData() (Frame method), [29](#)  
getRemainingData() (HybiFrame method), [32](#)  
getRemainingData() (HybiPayload method), [36](#)  
getRemainingData() (Payload method), [37](#)  
getRequestHandshake() (Hybi10Protocol method), [39](#)  
getRequestHandshake() (HybiProtocol method), [43](#)  
getRequestHandshake() (Protocol method), [48](#)  
getRequestHandshake() (Rfc6455Protocol method), [52](#)  
getRequestHeaders() (Hybi10Protocol method), [41](#)  
getRequestHeaders() (HybiProtocol method), [45](#)  
getRequestHeaders() (Protocol method), [50](#)  
getRequestHeaders() (Rfc6455Protocol method), [54](#)  
getResource() (ClientSocket method), [59](#)  
getResource() (Resource method), [55](#)  
getResource() (ServerClientSocket method), [61](#)  
getResource() (ServerSocket method), [64](#)  
getResource() (Socket method), [66](#)  
getResource() (UriSocket method), [68](#)  
getResourceId() (ClientSocket method), [59](#)  
getResourceId() (Resource method), [55](#)  
getResourceId() (ServerClientSocket method), [61](#)  
getResourceId() (ServerSocket method), [64](#)  
getResourceId() (Socket method), [66](#)  
getResourceId() (UriSocket method), [68](#)  
getResponseError() (Hybi10Protocol method), [40](#)  
getResponseError() (HybiProtocol method), [44](#)  
getResponseError() (Protocol method), [48](#)  
getResponseError() (Rfc6455Protocol method), [52](#)  
getResponseHandshake() (Hybi10Protocol method), [40](#)  
getResponseHandshake() (HybiProtocol method), [44](#)  
getResponseHandshake() (Protocol method), [48](#)  
getResponseHandshake() (Rfc6455Protocol method), [52](#)  
getServer() (ConnectionManager method), [21](#)  
getSocket() (Connection method), [19](#)  
getSocketStreamContextOptions() (ClientSocket method), [58](#)  
getSocketStreamContextOptions() (ServerSocket method), [63](#)  
getSocketStreamContextOptions() (UriSocket method), [68](#)  
getSslStreamContextOptions() (ClientSocket method), [58](#)



- getSslStreamContextOptions() (ServerSocket method), [63](#)
  - getSslStreamContextOptions() (UriSocket method), [68](#)
  - getStreamContext() (ClientSocket method), [59](#)
  - getStreamContext() (ServerSocket method), [63](#)
  - getStreamContext() (UriSocket method), [67](#)
  - getSuccessResponseHeaders() (Hybi10Protocol method), [42](#)
  - getSuccessResponseHeaders() (HybiProtocol method), [46](#)
  - getSuccessResponseHeaders() (Protocol method), [50](#)
  - getSuccessResponseHeaders() (Rfc6455Protocol method), [54](#)
  - getTrace() (BadRequestException method), [22](#)
  - getTrace() (CloseException method), [22](#)
  - getTrace() (ConnectionException method), [23](#)
  - getTrace() (Exception method), [24](#)
  - getTrace() (FrameException method), [24](#)
  - getTrace() (HandshakeException method), [25](#)
  - getTrace() (InvalidOriginException method), [26](#)
  - getTrace() (PayloadException method), [26](#)
  - getTrace() (RateLimiterException method), [27](#)
  - getTrace() (SocketException method), [28](#)
  - getTraceAsString() (BadRequestException method), [22](#)
  - getTraceAsString() (CloseException method), [22](#)
  - getTraceAsString() (ConnectionException method), [23](#)
  - getTraceAsString() (Exception method), [24](#)
  - getTraceAsString() (FrameException method), [24](#)
  - getTraceAsString() (HandshakeException method), [25](#)
  - getTraceAsString() (InvalidOriginException method), [26](#)
  - getTraceAsString() (PayloadException method), [26](#)
  - getTraceAsString() (RateLimiterException method), [27](#)
  - getTraceAsString() (SocketException method), [28](#)
  - getType() (Frame method), [29](#)
  - getType() (HybiFrame method), [31](#)
  - getType() (HybiPayload method), [36](#)
  - getType() (Payload method), [38](#)
  - getUri() (BasicServer method), [13](#)
  - getUri() (ClientSocket method), [59](#)
  - getUri() (ConnectionManager method), [21](#)
  - getUri() (Server method), [56](#)
  - getUri() (ServerSocket method), [63](#)
  - getUri() (UriSocket method), [67](#)
  - getVersion() (Hybi10Protocol method), [39](#)
  - getVersion() (HybiProtocol method), [43](#)
  - getVersion() (Protocol method), [47](#)
  - getVersion() (Rfc6455Protocol method), [52](#)
- ## H
- handle() (Connection method), [17](#)
  - handlePayload() (Connection method), [18](#)
  - handshake() (Connection method), [17](#)
  - handshaked (Connection property), [16](#)
  - HandshakeException (class), [24](#)
  - HandshakeRequestListener (class), [32](#)
  - headers (Client property), [14](#)
  - headers (Connection property), [16](#)
  - host (ClientSocket property), [58](#)
  - host (ServerSocket property), [62](#)
  - host (UriSocket property), [67](#)
  - httpResponses (Hybi10Protocol property), [39](#)
  - httpResponses (HybiProtocol property), [43](#)
  - httpResponses (Protocol property), [47](#)
  - httpResponses (Rfc6455Protocol property), [52](#)
  - Hybi10Protocol (class), [38](#)
  - Hybi10Protocol::CLOSE\_NORMAL (class constant), [38](#)
  - Hybi10Protocol::CONNECTION\_VALUE (class constant), [38](#)
  - Hybi10Protocol::HEADER\_HOST (class constant), [38](#)
  - Hybi10Protocol::HEADER\_LINE\_FORMAT (class constant), [39](#)
  - Hybi10Protocol::HTTP\_SWITCHING\_PROTOCOLS (class constant), [38](#)
  - Hybi10Protocol::MAGIC\_GUID (class constant), [38](#)
  - Hybi10Protocol::REQUEST\_LINE\_FORMAT (class constant), [39](#)
  - Hybi10Protocol::REQUEST\_LINE\_REGEX (class constant), [39](#)
  - Hybi10Protocol::RESPONSE\_LINE\_FORMAT (class constant), [39](#)
  - Hybi10Protocol::SCHEME\_WEBSOCKET (class constant), [38](#)
  - Hybi10Protocol::TYPE\_CONTINUATION (class constant), [38](#)
  - Hybi10Protocol::UPGRADE\_VALUE (class constant), [38](#)
  - HybiFrame (class), [30](#)
  - HybiPayload (class), [35](#)
  - HybiProtocol (class), [42](#)
  - HybiProtocol::CLOSE\_NORMAL (class constant), [42](#)
  - HybiProtocol::CONNECTION\_VALUE (class constant), [43](#)
  - HybiProtocol::HEADER\_HOST (class constant), [42](#)
  - HybiProtocol::HEADER\_LINE\_FORMAT (class constant), [43](#)
  - HybiProtocol::HTTP\_SWITCHING\_PROTOCOLS (class constant), [42](#)
  - HybiProtocol::MAGIC\_GUID (class constant), [42](#)
  - HybiProtocol::REQUEST\_LINE\_FORMAT (class constant), [43](#)
  - HybiProtocol::REQUEST\_LINE\_REGEX (class constant), [43](#)
  - HybiProtocol::RESPONSE\_LINE\_FORMAT (class constant), [43](#)
  - HybiProtocol::SCHEME\_WEBSOCKET (class constant), [42](#)
  - HybiProtocol::TYPE\_CONTINUATION (class constant), [42](#)

HybiProtocol::UPGRADE\_VALUE (class constant), [42](#)

## I

id (Connection property), [16](#)  
InvalidOriginException (class), [25](#)  
ip (Connection property), [16](#)  
ips (RateLimiter property), [34](#)  
isAllowed() (OriginPolicy method), [33](#)  
isComplete() (Frame method), [29](#)  
isComplete() (HybiFrame method), [32](#)  
isComplete() (HybiPayload method), [36](#)  
isComplete() (Payload method), [37](#)  
isConnected() (Client method), [16](#)  
isConnected() (ClientSocket method), [59](#)  
isConnected() (ServerClientSocket method), [61](#)  
isConnected() (ServerSocket method), [64](#)  
isConnected() (Socket method), [66](#)  
isConnected() (UriSocket method), [68](#)  
isFinal() (Frame method), [29](#)  
isFinal() (HybiFrame method), [31](#)  
isMasked() (HybiFrame method), [31](#)  
isWaitingForData() (Frame method), [29](#)  
isWaitingForData() (HybiFrame method), [32](#)  
isWaitingForData() (HybiPayload method), [36](#)  
isWaitingForData() (Payload method), [37](#)

## L

length (Frame property), [28](#)  
length (HybiFrame property), [30](#)  
limit() (RateLimiter method), [35](#)  
line (BadRequestException property), [21](#)  
line (CloseException property), [22](#)  
line (ConnectionException property), [23](#)  
line (Exception property), [23](#)  
line (FrameException property), [24](#)  
line (HandshakeException property), [25](#)  
line (InvalidOriginException property), [25](#)  
line (PayloadException property), [26](#)  
line (RateLimiterException property), [27](#)  
line (SocketException property), [27](#)  
listen() (ConnectionManager method), [20](#)  
listen() (Listener method), [33](#)  
listen() (OriginPolicy method), [33](#)  
listen() (RateLimiter method), [34](#)  
listen() (ServerSocket method), [63](#)  
Listener (class), [33](#)  
listeners (BasicServer property), [12](#)  
listeners (Server property), [55](#)  
listening (ServerSocket property), [62](#)  
log() (BasicServer method), [13](#)  
log() (Connection method), [18](#)  
log() (ConnectionManager method), [21](#)  
log() (RateLimiter method), [35](#)  
log() (Server method), [56](#)

logger (BasicServer property), [12](#)  
logger (Server property), [55](#)

## M

manager (Connection property), [16](#)  
mask (HybiFrame property), [30](#)  
mask() (HybiFrame method), [30](#)  
masked (HybiFrame property), [30](#)  
message (BadRequestException property), [21](#)  
message (CloseException property), [22](#)  
message (ConnectionException property), [22](#)  
message (Exception property), [23](#)  
message (FrameException property), [24](#)  
message (HandshakeException property), [24](#)  
message (InvalidOriginException property), [25](#)  
message (PayloadException property), [26](#)  
message (RateLimiterException property), [27](#)  
message (SocketException property), [27](#)

## N

name (ClientSocket property), [58](#)  
name (ServerClientSocket property), [60](#)  
name (ServerSocket property), [63](#)  
name (Socket property), [65](#)  
name (UriSocket property), [67](#)  
notify() (BasicServer method), [13](#)  
notify() (Server method), [57](#)

## O

offset\_mask (HybiFrame property), [30](#)  
offset\_payload (HybiFrame property), [30](#)  
onClientData() (RateLimiter method), [34](#)  
onData() (Application method), [11](#)  
onData() (Connection method), [17](#)  
onData() (EchoApplication method), [11](#)  
onHandshakeRequest() (HandshakeRequestListener method), [32](#)  
onHandshakeRequest() (OriginPolicy method), [33](#)  
onSocketConnect() (RateLimiter method), [34](#)  
onSocketDisconnect() (RateLimiter method), [34](#)  
options (BasicServer property), [12](#)  
options (Client property), [15](#)  
options (ClientSocket property), [58](#)  
options (Configurable property), [69](#)  
options (Connection property), [17](#)  
options (ConnectionManager property), [19](#)  
options (RateLimiter property), [34](#)  
options (Server property), [55](#)  
options (ServerClientSocket property), [60](#)  
options (ServerSocket property), [63](#)  
options (Socket property), [65](#)  
options (UriSocket property), [67](#)  
origin (Client property), [14](#)  
originPolicy (BasicServer property), [12](#)

OriginPolicy (class), [33](#)

## P

Payload (class), [37](#)

payload (Connection property), [16](#)

payload (Frame property), [28](#)

payload (HybiFrame property), [30](#)

PayloadException (class), [26](#)

port (ClientSocket property), [58](#)

port (Connection property), [16](#)

port (ServerSocket property), [62](#)

port (UriSocket property), [67](#)

process() (Connection method), [18](#)

processClientSocket() (ConnectionManager method), [20](#)

processMasterSocket() (ConnectionManager method), [20](#)

protocol (BasicServer property), [12](#)

Protocol (class), [46](#)

protocol (Client property), [14](#)

protocol (ClientSocket property), [58](#)

protocol (Configurable property), [69](#)

protocol (Connection property), [17](#)

protocol (ConnectionManager property), [19](#)

protocol (RateLimiter property), [34](#)

protocol (Server property), [56](#)

protocol (ServerClientSocket property), [60](#)

protocol (ServerSocket property), [63](#)

protocol (Socket property), [65](#)

protocol (UriSocket property), [67](#)

Protocol::CLOSE\_NORMAL (class constant), [46](#)

Protocol::CONNECTION\_VALUE (class constant), [47](#)

Protocol::HEADER\_HOST (class constant), [46](#)

Protocol::HEADER\_LINE\_FORMAT (class constant), [47](#)

Protocol::HTTP\_SWITCHING\_PROTOCOLS (class constant), [46](#)

Protocol::MAGIC\_GUID (class constant), [47](#)

Protocol::REQUEST\_LINE\_FORMAT (class constant), [47](#)

Protocol::REQUEST\_LINE\_REGEX (class constant), [47](#)

Protocol::RESPONSE\_LINE\_FORMAT (class constant), [47](#)

Protocol::SCHEME\_WEBSOCKET (class constant), [46](#)

Protocol::TYPE\_CONTINUATION (class constant), [47](#)

Protocol::UPGRADE\_VALUE (class constant), [47](#)

## Q

queryParams (Connection property), [16](#)

## R

rateLimiter (BasicServer property), [12](#)

RateLimiter (class), [33](#)

RateLimiterException (class), [27](#)

receive() (ClientSocket method), [60](#)

receive() (ServerClientSocket method), [62](#)

receive() (ServerSocket method), [64](#)

receive() (Socket method), [66](#)

receive() (UriSocket method), [69](#)

receiveData() (Frame method), [29](#)

receiveData() (HybiFrame method), [31](#)

receiveData() (HybiPayload method), [36](#)

receiveData() (Payload method), [37](#)

reconnect() (ClientSocket method), [58](#)

registerApplication() (BasicServer method), [14](#)

registerApplication() (Server method), [57](#)

releaseConnection() (RateLimiter method), [35](#)

removeConnection() (ConnectionManager method), [21](#)

requests (RateLimiter property), [34](#)

Resource (class), [55](#)

resourceId() (ConnectionManager method), [20](#)

resources (ConnectionManager property), [19](#)

Rfc6455Protocol (class), [51](#)

Rfc6455Protocol::CLOSE\_NORMAL (class constant), [51](#)

Rfc6455Protocol::CONNECTION\_VALUE (class constant), [51](#)

Rfc6455Protocol::HEADER\_HOST (class constant), [51](#)

Rfc6455Protocol::HEADER\_LINE\_FORMAT (class constant), [51](#)

Rfc6455Protocol::HTTP\_SWITCHING\_PROTOCOLS (class constant), [51](#)

Rfc6455Protocol::MAGIC\_GUID (class constant), [51](#)

Rfc6455Protocol::REQUEST\_LINE\_FORMAT (class constant), [51](#)

Rfc6455Protocol::REQUEST\_LINE\_REGEX (class constant), [51](#)

Rfc6455Protocol::RESPONSE\_LINE\_FORMAT (class constant), [51](#)

Rfc6455Protocol::SCHEME\_WEBSOCKET (class constant), [51](#)

Rfc6455Protocol::TYPE\_CONTINUATION (class constant), [51](#)

Rfc6455Protocol::UPGRADE\_VALUE (class constant), [51](#)

run() (BasicServer method), [13](#)

run() (Server method), [56](#)

## S

scheme (ClientSocket property), [58](#)

scheme (ServerSocket property), [62](#)

scheme (UriSocket property), [67](#)

schemes (Hybi10Protocol property), [39](#)

schemes (HybiProtocol property), [43](#)

schemes (Protocol property), [47](#)

schemes (Rfc6455Protocol property), [51](#)

selectAndProcess() (ConnectionManager method), [20](#)

send() (ClientSocket method), [59](#)

send() (Connection method), [18](#)

send() (ServerClientSocket method), [61](#)

send() (ServerSocket method), [64](#)  
send() (Socket method), [66](#)  
send() (UriSocket method), [69](#)  
sendData() (Client method), [15](#)  
sendToSocket() (HybiPayload method), [36](#)  
sendToSocket() (Payload method), [37](#)  
Server (class), [55](#)  
server (ConnectionManager property), [19](#)  
server (RateLimiter property), [33](#)  
Server::EVENT\_SOCKET\_CONNECT (class constant), [55](#)  
ServerClientSocket (class), [60](#)  
ServerClientSocket::DEFAULT\_RECEIVE\_LENGTH (class constant), [60](#)  
ServerClientSocket::NAME\_PART\_IP (class constant), [60](#)  
ServerClientSocket::TIMEOUT\_SOCKET (class constant), [60](#)  
ServerSocket (class), [62](#)  
ServerSocket::DEFAULT\_RECEIVE\_LENGTH (class constant), [62](#)  
ServerSocket::NAME\_PART\_IP (class constant), [62](#)  
ServerSocket::TIMEOUT\_SOCKET (class constant), [62](#)  
setLogger() (BasicServer method), [13](#)  
setLogger() (Server method), [56](#)  
Socket (class), [64](#)  
socket (Client property), [14](#)  
socket (ClientSocket property), [58](#)  
socket (Connection property), [16](#)  
socket (ConnectionManager property), [19](#)  
socket (ServerClientSocket property), [60](#)  
socket (ServerSocket property), [62](#)  
socket (Socket property), [65](#)  
socket (UriSocket property), [67](#)  
Socket::DEFAULT\_RECEIVE\_LENGTH (class constant), [65](#)  
Socket::NAME\_PART\_IP (class constant), [65](#)  
Socket::TIMEOUT\_SOCKET (class constant), [64](#)  
SocketException (class), [27](#)  
Ssl (class), [69](#)

## T

type (Frame property), [28](#)  
type (HybiFrame property), [30](#)

## U

unmask() (HybiFrame method), [31](#)  
uri (BasicServer property), [12](#)  
uri (Client property), [14](#)  
uri (Server property), [55](#)  
UriSocket (class), [66](#)  
UriSocket::DEFAULT\_RECEIVE\_LENGTH (class constant), [66](#)  
UriSocket::NAME\_PART\_IP (class constant), [67](#)

UriSocket::TIMEOUT\_SOCKET (class constant), [66](#)

## V

validateOriginUri() (Hybi10Protocol method), [41](#)  
validateOriginUri() (HybiProtocol method), [45](#)  
validateOriginUri() (Protocol method), [49](#)  
validateOriginUri() (Rfc6455Protocol method), [53](#)  
validateRequestHandshake() (Hybi10Protocol method), [40](#)  
validateRequestHandshake() (HybiProtocol method), [44](#)  
validateRequestHandshake() (Protocol method), [49](#)  
validateRequestHandshake() (Rfc6455Protocol method), [53](#)  
validateRequestLine() (Hybi10Protocol method), [41](#)  
validateRequestLine() (HybiProtocol method), [45](#)  
validateRequestLine() (Protocol method), [49](#)  
validateRequestLine() (Rfc6455Protocol method), [53](#)  
validateResponseHandshake() (Hybi10Protocol method), [40](#)  
validateResponseHandshake() (HybiProtocol method), [44](#)  
validateResponseHandshake() (Protocol method), [48](#)  
validateResponseHandshake() (Rfc6455Protocol method), [53](#)  
validateScheme() (Hybi10Protocol method), [41](#)  
validateScheme() (HybiProtocol method), [46](#)  
validateScheme() (Protocol method), [50](#)  
validateScheme() (Rfc6455Protocol method), [54](#)  
validateSocketUri() (Hybi10Protocol method), [41](#)  
validateSocketUri() (HybiProtocol method), [45](#)  
validateSocketUri() (Protocol method), [49](#)  
validateSocketUri() (Rfc6455Protocol method), [53](#)  
validateUri() (Hybi10Protocol method), [40](#)  
validateUri() (HybiProtocol method), [45](#)  
validateUri() (Protocol method), [49](#)  
validateUri() (Rfc6455Protocol method), [53](#)