PHP Beanstalkd Client Documentation

Release 1.0.0-beta1

Josh Dechant

September 07, 2015

Contents

A beanstalkd client for PHP 5.5+

Comparison of PHP beanstalkd clients

Client	Pool Support	PHP Version	HHVM	Beanstalkd Version
php-beanstalk		5.5+		latest
c		5.0+		?
Pheanstalk		5.3+		latest

CHAPTER 1

Contents

1.1 Quickstart

1.1.1 As a Producer

```
// returns BeanstalkPool instance
$bean = (new Beanstalk\Pool)
     ->addServer('localhost', 11300)
     ->useTube('my-tube')
     ->put('Hello World!');
```

1.1.2 As a Consumer

```
$bean = (new Beanstalk\Pool)
     ->addServer('localhost', 11300)
     ->watchTube('my-tube');

while (true)
{
    try
    {
        $job = $bean->reserve($timeout = 10);

        /* process job ... */

        $job->delete();
    }
    catch (Beanstalk\Exception $e)
    {
        switch ($e->getCode())
        {
            case Beanstalk\Exception::TIMED_OUT:
                echo "Timed out waiting for a job.  Retrying in 1 second."
                sleep(1);
                continue;
                break;

            default:
                throw $e;
                break;
```

```
    }
}
```

1.1.3 Built-in JSON support

Objects are automatically converted

```
$bean = (new Beanstalk\Pool)
    ->addServer('localhost', 11300)
    ->useTube('my-tube');

$obj = new stdClass;
$obj->content = 'Hello World!';
$bean->put($obj); // stored in beanstalkd as '{"content":"Hello World!"}'

$bean->watchTube('my-tube');
$job = $bean->reserve();
print_r($job->getMessage());
```

Outputs:

```
stdClass Object
(
    [content] => Hello World!
)
```

Send a custom JSON string

```
$bean = (new Beanstalk\Pool)
    ->addServer('localhost', 11300)
    ->useTube('my-tube')
$bean->put('[123,456,789]');

$bean->watchTube('my-tube');
$job = $bean->reserve();
print_r($job->getMessage());
```

Outputs:

```
Array
(
    [0] => 123
    [1] => 456
    [2] => 789
)
```

1.2 Beanstalkd Client Class Refs

1.2.1 Beanstalk\Command Class Ref

class Beanstalk**Command**

Description Abstract beanstalk command.

Author Joshua Dechant <jdechant@shapeup.com>

All commands must extends this class

Class Methods

- *Command::getCommand* – Get the command to send to the beanstalkd server
- *Command::getData* – Get data, if any, to send with the command.
- *Command::parseResponse* – Parse the response for success or failure.
- *Command::returnsData* – Does the command return data?

Beanstalk\Command::**getCommand**()

Description Get the command to send to the beanstalkd server

Returns *string*

Beanstalk\Command::**getData**()

Description Get data, if any, to send with the command.

Returns *mixed* Data string to send with command or boolean false if none

Not all commands have data; in fact, most do not.

Beanstalk\Command::**parseResponse**(*$response*[, *$data* = *null*, *$conn* = *null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *mixed* On success

Throws *BeanstalkException* On failure

Failures should throw a BeanstalkException with the error message.

Beanstalk\Command::**returnsData**()

Description Does the command return data?

Returns *boolean*

1.2.2 Beanstalk\Connection Class Ref

class Beanstalk**Connection**

Description Beanstalkd connection

Author Joshua Dechant <jdechant@shapeup.com>

Class Methods

- `Connection::__construct` – Constructor; establishes connection stream
- `Connection::bury` – Bury command
- `Connection::close` – Close the connection
- `Connection::connect` – Connect to the beanstalkd server
- `Connection::delete` – Delete command
- `Connection::getServer` – Get the Beanstalkd server address
- `Connection::getStream` – Get the connect's stream
- `Connection::getTimeout` – Get the connection timeout
- `Connection::ignoreTube` – Ignore command
- `Connection::isTimedOut` – Has the connection timed out?
- `Connection::kick` – Kick command
- `Connection::listTubes` – The list-tubes command returns a list of all existing tubes
- `Connection::pauseTube` – The pause-tube command can delay any new job being reserved for a given time
- `Connection::peek` – Return job $id
- `Connection::peekBuried` – Return the next job in the list of buried jobs
- `Connection::peekDelayed` – Return the delayed job with the shortest delay left
- `Connection::peekReady` – Return the next ready job
- `Connection::put` – The "put" command is for any process that wants to insert a job into the queue
- `Connection::release` – Release command
- `Connection::reserve` – Reserve command
- `Connection::setTimeout` – Set the connection timeout
- `Connection::stats` – The stats command gives statistical information about the system as a whole.
- `Connection::statsJob` – The stats-job command gives statistical information about the specified job if it exists.
- `Connection::statsTube` – The stats-tube command gives statistical information about the specified tube if it exists.
- `Connection::touch` – Touch command
- `Connection::useTube` – Use command
- `Connection::validateResponse` – Generic validation for all responses from beanstalkd
- `Connection::watchTube` – Watch command

`Beanstalk\Connection::__construct` (*$address*, *$stream*[, *$timeout = 500*])

Description Constructor; establishes connection stream

Parameters

- **`$address`** (*string*) – Beanstalkd server address in the format "host:port"
- **`$stream`** (*BeanstalkConnectionStream*) – Stream to use for connection
- **`$timeout`** (*float*) – Connection timeout in milliseconds

Throws *BeanstalkException* When a connection cannot be established

`Beanstalk\Connection::bury` (*$id*, *$priority*)

Description Bury command

Parameters

- **`$id`** (*integer*) – The job id to bury
- **`$priority`** (*integer*) – A new priority to assign to the job

The bury command puts a job into the "buried" state. Buried jobs are put into a FIFO linked list and will not be touched by the server again until a client kicks them with the "kick" command.

`Beanstalk\Connection::close()`

Description Close the connection

`Beanstalk\Connection::connect()`

Description Connect to the beanstalkd server

Returns *boolean*

Throws *BeanstalkException* When a connection cannot be established

`Beanstalk\Connection::delete(`*$id*`)`

Description Delete command

Parameters

- **$id** (*integer*) – The job id to delete

Returns *boolean*

Throws *BeanstalkException*

The delete command removes a job from the server entirely. It is normally used by the client when the job has successfully run to completion. A client can delete jobs that it has reserved, ready jobs, and jobs that are buried.

`Beanstalk\Connection::getServer()`

Description Get the Beanstalkd server address

Returns *string* Beanstalkd server address in the format "host:port"

`Beanstalk\Connection::getStream()`

Description Get the connect's stream

Returns *BeanstalkConnectionStream*

`Beanstalk\Connection::getTimeout()`

Description Get the connection timeout

Returns *float* Connection timeout

`Beanstalk\Connection::ignoreTube(`*$tube*`)`

Description Ignore command

Parameters

- **$tube** (*string*) – Tube to remove from the watch list

The "ignore" command is for consumers. It removes the named tube from the watch list for the current connection.

`Beanstalk\Connection::isTimedOut()`

Description Has the connection timed out?

Returns *boolean*

`Beanstalk\Connection::kick(`*$bound*`)`

Description Kick command

Parameters

- **$bound** (*integer*) – Upper bound on the number of jobs to kick. The server will kick no more than $bound jobs.

Returns *integer* The number of jobs actually kicked

The kick command applies only to the currently used tube. It moves jobs into the ready queue. If there are any buried jobs, it will only kick buried jobs. Otherwise it will kick delayed jobs

`Beanstalk\Connection::listTubes()`

Description The list-tubes command returns a list of all existing tubes

`Beanstalk\Connection::pauseTube(`*$tube*, *$delay*`)`

Description The pause-tube command can delay any new job being reserved for a given time

Parameters

- **$tube** (*string*) – The tube to pause
- **$delay** (*integer*) – Number of seconds to wait before reserving any more jobs from the queue

Returns *boolean*

Throws *BeanstalkException*

`Beanstalk\Connection::peek(`*$id*`)`

Description Return job $id

Parameters

- **$id** (*integer*) – Id of job to return

Returns *BeanstalkJob*

Throws *BeanstalkException* When job cannot be found

`Beanstalk\Connection::peekBuried()`

Description Return the next job in the list of buried jobs

Returns *BeanstalkJob*

Throws *BeanstalkException* When no jobs in buried state

`Beanstalk\Connection::peekDelayed()`

Description Return the delayed job with the shortest delay left

Returns *BeanstalkJob*

Throws *BeanstalkException* When no jobs in delayed state

`Beanstalk\Connection::peekReady()`

Description Return the next ready job

Returns *BeanstalkJob*

Throws *BeanstalkException* When no jobs in ready state

`Beanstalk\Connection::put(`*$message*`[`, *$priority = 65536*, *$delay = 0*, *$ttr = 120* `]`)

Description The “put” command is for any process that wants to insert a job into the queue

Parameters

- **$message** (*mixed*) – Description

- **$priority** (*integer*) – Job priority.
- **$delay** (*integer*) – Number of seconds to wait before putting the job in the ready queue.
- **$ttr** (*integer*) – Time to run. The number of seconds to allow a worker to run this job.

Beanstalk\Connection::**release**(*$id*, *$priority*, *$delay*)

Description Release command

Parameters

- **$id** (*integer*) – The job id to release
- **$priority** (*integer*) – A new priority to assign to the job
- **$delay** (*integer*) – Number of seconds to wait before putting the job in the ready queue. The job will be in the "delayed" state during this time

The release command puts a reserved job back into the ready queue (and marks its state as "ready") to be run by any client. It is normally used when the job fails because of a transitory error.

Beanstalk\Connection::**reserve**([*$timeout = null*])

Description Reserve command

Parameters

- **$timeout** (*integer*) – Wait timeout in seconds

This will return a newly-reserved job. If no job is available to be reserved, beanstalkd will wait to send a response until one becomes available. Once a job is reserved for the client, the client has limited time to run (TTR) the job before the job times out. When the job times out, the server will put the job back into the ready queue. Both the TTR and the actual time left can be found in response to the stats-job command.

A timeout value of 0 will cause the server to immediately return either a response or TIMED_OUT. A positive value of timeout will limit the amount of time the client will block on the reserve request until a job becomes available.

Beanstalk\Connection::**setTimeout**(*$timeout*)

Description Set the connection timeout

Parameters

- **$timeout** (*float*) – Connection timeout in milliseconds

Beanstalk\Connection::**stats**()

Description The stats command gives statistical information about the system as a whole.

Beanstalk\Connection::**statsJob**(*$id*)

Description The stats-job command gives statistical information about the specified job if it exists.

Parameters

- **$id** (*integer*) – The job id to get stats on

Returns *BeanstalkStats*

Throws *BeanstalkException* When the job does not exist

Beanstalk\Connection::**statsTube**(*$tube*)

Description The stats-tube command gives statistical information about the specified tube if it exists.

Parameters

- **$tube** (*string*) – is a name at most 200 bytes. Stats will be returned for this tube.

Returns *BeanstalkStats*

Throws *BeanstalkException* When the tube does not exist

`Beanstalk\Connection::touch`(*$id*)

Description Touch command

Parameters

- **$id** (*integer*) – The job id to touch

Returns *boolean*

Throws *BeanstalkException*

The "touch" command allows a worker to request more time to work on a job. This is useful for jobs that potentially take a long time, but you still want the benefits of a TTR pulling a job away from an unresponsive worker. A worker may periodically tell the server that it's still alive and processing a job (e.g. it may do this on DEADLINE_SOON).

`Beanstalk\Connection::useTube`(*$tube*)

Description Use command

Parameters

- **$tube** (*string*) – The tube to use. If the tube does not exist, it will be created.

The "use" command is for producers. Subsequent put commands will put jobs into the tube specified by this command. If no use command has been issued, jobs will be put into the tube named "default".

`Beanstalk\Connection::validateResponse`(*$response*)

Description Generic validation for all responses from beanstalkd

Parameters

- **$response** (*string*) –

Returns *boolean* true when response is valid

Throws *BeanstalkException* When response is invalid

`Beanstalk\Connection::watchTube`(*$tube*)

Description Watch command

Parameters

- **$tube** (*string*) – Tube to add to the watch list. If the tube doesn't exist, it will be created

The "watch" command adds the named tube to the watch list for the current connection. A reserve command will take a job from any of the tubes in the watch list. For each new connection, the watch list initially consists of one tube, named "default".

1.2.3 Beanstalk\Exception Class Ref

class `Beanstalk\Exception`

Extends *Exception*

Description Beanstalk Exceptions

Author Joshua Dechant <jdechant@shapeup.com>

Class Constants

constant Beanstalk\Exception::**OUT_OF_MEMORY**
constant Beanstalk\Exception::**INTERNAL_ERROR**
constant Beanstalk\Exception::**BAD_FORMAT**
constant Beanstalk\Exception::**UNKNOWN_COMMAND**
constant Beanstalk\Exception::**BURIED**
constant Beanstalk\Exception::**NOT_FOUND**
constant Beanstalk\Exception::**EXPECTED_CRLF**
constant Beanstalk\Exception::**JOB_TOO_BIG**
constant Beanstalk\Exception::**DEADLINE_SOON**
constant Beanstalk\Exception::**TIMED_OUT**
constant Beanstalk\Exception::**TUBE_NAME_TOO_LONG**
constant Beanstalk\Exception::**NOT_IGNORED**
constant Beanstalk\Exception::**UNKNOWN**
constant Beanstalk\Exception::**SERVER_OFFLINE**
constant Beanstalk\Exception::**SERVER_READ**
constant Beanstalk\Exception::**SERVER_WRITE**

Class Methods

- *Exception::__construct*
- *Exception::getCodeAsString* – Get a string representation of a given code

Beanstalk\Exception::**__construct**(*$message*[, *$code = 0*, *$previous = null*])

Parameters

- **$message** (*mixed*) –
- **$code** (*mixed*) –
- **$previous** (*BeanstalkException*) –

Beanstalk\Exception::**getCodeAsString**()

Description Get a string representation of a given code

Returns *string*

1.2.4 Beanstalk\Job Class Ref

class Beanstalk**Job**

Description A Beanstalkd job

Author Joshua Dechant <jdechant@shapeup.com>

Class Methods

Beanstalk\Job::**__construct**(*$conn*, *$id*, *$message*)

Description Constructor

Parameters

- **$conn** (*BeanstalkConnection*) – Connection for the job
- **$id** (*integer*) – Job id
- **$message** (*string*) – Job body. If the body is JSON, it will be converted to an object

Beanstalk\Job::**bury**([*$priority = 2048*])

Description Bury the job

Parameters

- **$priority** (*integer*) – A new priority to assign to the job

The bury command puts a job into the "buried" state. Buried jobs are put into a FIFO linked list and will not be touched by the server again until a client kicks them with the "kick" command.

Beanstalk\Job::**delete**()

Description Delete the job

Returns *boolean*

Throws *BeanstalkException*

The delete command removes a job from the server entirely. It is normally used by the client when the job has successfully run to completion.

Beanstalk\Job::**getConnection**()

Description Get the beanstalkd connection for the job

Returns *BeanstalkConnection*

Beanstalk\Job::**getId**()

Description Get the job id

Returns *integer*

Beanstalk\Job::**getMessage**()

Description Get the job body/message

Returns *mixed* String of body for simple message; stdClass for JSON messages

Beanstalk\Job::**release**([*$delay = 10*, *$priority = 5*])

Description Release the job

Parameters

- **$delay** (*integer*) – Number of seconds to wait before putting the job in the ready queue.
- **$priority** (*integer*) – A new priority to assign to the job

Returns *boolean*

Throws *BeanstalkException*

The release command puts a reserved job back into the ready queue (and marks its state as "ready") to be run by any client. It is normally used when the job fails because of a transitory error.

`Beanstalk\Job::stats()`

Description Get stats on the job

Returns *BeanstalkStats*

Throws *BeanstalkException* When the job does not exist

The stats-job command gives statistical information about the specified job if it exists.

`Beanstalk\Job::touch()`

Description Touch the job

Returns *boolean*

Throws *BeanstalkException*

The "touch" command allows a worker to request more time to work on a job. This is useful for jobs that potentially take a long time, but you still want the benefits of a TTR pulling a job away from an unresponsive worker. A worker may periodically tell the server that it's still alive and processing a job (e.g. it may do this on DEADLINE_SOON).

1.2.5 Beanstalk\Pool Class Ref

class `Beanstalk\Pool`

Description Beanstalkd connection pool

Author Joshua Dechant <jdechant@shapeup.com>

```
$beanstalk = (new \Beanstalk\Pool)
    ->addServer('localhost', 11300)
    ->useTube('my-tube');
$beanstalk->put('Hello World!');
```

Class Methods

- `Pool::addServer` – Add a beanstalkd server to the pool
- `Pool::close` – Close all connections in the pool
- `Pool::connect` – Establish a connection to all servers in the pool
- `Pool::getConnections`
- `Pool::getLastConnection`
- `Pool::getServers` – Get the Beanstalkd server addresses in the pool
- `Pool::getTimeout` – Get the current connection timeout
- `Pool::ignoreTube` – Ignore command
- `Pool::kick` – Kick command
- `Pool::listTubes` – The list-tubes command returns a list of all existing tubes
- `Pool::pauseTube` – The pause-tube command can delay any new job being reserved for a given time
- `Pool::put` – The "put" command is for any process that wants to insert a job into the queue
- `Pool::reserve` – Reserve command
- `Pool::setStream` – Sets the stream class to use for the connections in the pool
- `Pool::setTimeout` – Set the connection timeout for attempting to connect to servers in the pool
- `Pool::stats` – The stats command gives statistical information about the system as a whole
- `Pool::useTube` – Use command
- `Pool::watchTube` – Watch command

Beanstalk\Pool::**addServer**(*$host*[, *$port = 11300*])

Description Add a beanstalkd server to the pool

Parameters

- **$host** (*string*) – Server host
- **$port** (*integer*) – Server port

Returns *self*

Beanstalk\Pool::**close**()

Description Close all connections in the pool

Beanstalk\Pool::**connect**()

Description Establish a connection to all servers in the pool

Beanstalk\Pool::**getConnections**()

Beanstalk\Pool::**getLastConnection**()

Beanstalk\Pool::**getServers**()

Description Get the Beanstalkd server addresses in the pool

Returns *array* Beanstalkd server addresses in the format "host:port"

Beanstalk\Pool::**getTimeout**()

Description Get the current connection timeout

Returns *float* Current connection timeout

Beanstalk\Pool::**ignoreTube**(*$tube*)

Description Ignore command

Parameters

- **$tube** (*string*) – Tube to remove from the watch list

Returns *self*

The "ignore" command is for consumers. It removes the named tube from the watch list for the current connection.

`Beanstalk\Pool::kick` (*$bound*)

Description Kick command

Parameters

- **$bound** (*integer*) – Upper bound on the number of jobs to kick. Each server will kick no more than $bound jobs.

Returns *integer* The number of jobs actually kicked

The kick command applies only to the currently used tube. It moves jobs into the ready queue. If there are any buried jobs, it will only kick buried jobs. Otherwise it will kick delayed jobs

`Beanstalk\Pool::listTubes` ()

Description The list-tubes command returns a list of all existing tubes

`Beanstalk\Pool::pauseTube` (*$tube*, *$delay*)

Description The pause-tube command can delay any new job being reserved for a given time

Parameters

- **$tube** (*string*) – The tube to pause
- **$delay** (*integer*) – Number of seconds to wait before reserving any more jobs from the queue

Returns *boolean*

Throws *BeanstalkException*

`Beanstalk\Pool::put` (*$message*[, *$priority = 65536*, *$delay = 0*, *$ttr = 120*])

Description The "put" command is for any process that wants to insert a job into the queue

Parameters

- **$message** (*mixed*) – Description
- **$priority** (*integer*) – Job priority.
- **$delay** (*integer*) – Number of seconds to wait before putting the job in the ready queue.
- **$ttr** (*integer*) – Time to run. The number of seconds to allow a worker to run this job.

`Beanstalk\Pool::reserve` ([*$timeout = null*])

Description Reserve command

Parameters

- **$timeout** (*integer*) – Wait timeout in seconds

This will return a newly-reserved job. If no job is available to be reserved, beanstalkd will wait to send a response until one becomes available. Once a job is reserved for the client, the client has limited time to run (TTR) the job before the job times out. When the job times out, the server will put the job back into the ready queue. Both the TTR and the actual time left can be found in response to the stats-job command.

A timeout value of 0 will cause the server to immediately return either a response or TIMED_OUT. A positive value of timeout will limit the amount of time the client will block on the reserve request until a job becomes available.

Beanstalk\Pool::**setStream**(*$class*)

Description Sets the stream class to use for the connections in the pool

Parameters

- **$class** (*string*) – Name of stream class

Returns *self*

Beanstalk\Pool::**setTimeout**(*$timeout*)

Description Set the connection timeout for attempting to connect to servers in the pool

Parameters

- **$timeout** (*float*) – Connection timeout in milliseconds

Returns *self*

Beanstalk\Pool::**stats**()

Description The stats command gives statistical information about the system as a whole

Beanstalk\Pool::**useTube**(*$tube*)

Description Use command

Parameters

- **$tube** (*string*) – The tube to use. If the tube does not exist, it will be created.

Returns *self*

The "use" command is for producers. Subsequent put commands will put jobs into the tube specified by this command. If no use command has been issued, jobs will be put into the tube named "default".

Beanstalk\Pool::**watchTube**(*$tube*)

Description Watch command

Parameters

- **$tube** (*string*) – Tube to add to the watch list. If the tube doesn't exist, it will be created

Returns *self*

The "watch" command adds the named tube to the watch list for the connection pool. A reserve command will take a job from any of the tubes in the watch list. For each new connection, the watch list initially consists of one tube, named "default".

1.2.6 Beanstalk\Stats Class Ref

class Beanstalk**Stats**

Class Methods

- *Stats::__construct* – Constructor
- *Stats::getStat* – Get the value of a given stat
- *Stats::getStats* – Get all the stats as an array
- *Stats::setStat* – Set a stat to a given value

Beanstalk\Stats::**__construct**([*$data = null*])

Description Constructor

Parameters

- **$data** (*string*) – Set stats from input data in the form "stat-1: valuenstat-2: value"

`Beanstalk\Stats::getStat` (*$stat*)

Description Get the value of a given stat

Parameters

- **$stat** (*string*) – Stat name to get the value of

Returns *string* Stat's value

Returns *boolean* false when value not set

`Beanstalk\Stats::getStats` ()

Description Get all the stats as an array

Returns *array* ('stat1-name' => 'value' , 'stat2-name' => 'value' , ...)

`Beanstalk\Stats::setStat` (*$stat*, *$value*)

Description Set a stat to a given value

Parameters

- **$stat** (*string*) –
- **$value** (*string*) –

Returns *null*

1.2.7 Beanstalk\Command\Bury Class Ref

class `Beanstalk\Command\Bury`

Extends *Beanstalk\Command*

Description Bury command

Author Joshua Dechant <jdechant@shapeup.com>

The bury command puts a job into the "buried" state. Buried jobs are put into a FIFO linked list and will not be touched by the server again until a client kicks them with the "kick" command.

Class Methods

- *Bury::__construct* – Constructor
- *Bury::getCommand* – Get the bury command to send to the beanstalkd server
- *Bury::parseResponse* – Parse the response for success or failure.

`Beanstalk\Command\Bury::__construct` (*$id*, *$priority*)

Description Constructor

Parameters

- **$id** (*integer*) – The job id to bury
- **$priority** (*integer*) – A new priority to assign to the job

Beanstalk\Command\Bury::**getCommand**()

Description Get the bury command to send to the beanstalkd server

Returns *string*

Beanstalk\Command\Bury::**parseResponse**(*$response*[, *$data* = *null*, *$conn* = *null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *boolean* True if command was successful

Throws *BeanstalkException* When the job cannot be found

Throws *BeanstalkException* When any other error occurs

1.2.8 Beanstalk\Command\Delete Class Ref

class Beanstalk\Command**Delete**

Extends *Beanstalk\Command*

Description Delete command

Author Joshua Dechant <jdechant@shapeup.com>

The delete command removes a job from the server entirely. It is normally used by the client when the job has successfully run to completion. A client can delete jobs that it has reserved, ready jobs, and jobs that are buried.

Class Methods

- *Delete::__construct* – Constructor
- *Delete::getCommand* – Get the delete command to send to the beanstalkd server
- *Delete::parseResponse* – Parse the response for success or failure.

Beanstalk\Command\Delete::**__construct**(*$id*)

Description Constructor

Parameters

- **$id** (*integer*) – The job id to delete

Beanstalk\Command\Delete::**getCommand**()

Description Get the delete command to send to the beanstalkd server

Returns *string*

Beanstalk\Command\Delete::**parseResponse**(*$response*[, *$data* = *null*, *$conn* = *null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response

- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *boolean* True if command was successful

Throws *BeanstalkException* When the job cannot be found or has already timed out

Throws *BeanstalkException* When any other error occurs

1.2.9 Beanstalk\Command\IgnoreTube Class Ref

class `Beanstalk\Command\`**`IgnoreTube`**

Extends *`Beanstalk\Command`*

Description Ignore command

Author Joshua Dechant <jdechant@shapeup.com>

The "ignore" command is for consumers. It removes the named tube from the watch list for the current connection.

Class Methods

- *`IgnoreTube::__construct`* – Constructor
- *`IgnoreTube::getCommand`* – Get the command to send to the beanstalkd server
- *`IgnoreTube::parseResponse`* – Parse the response for success or failure.

`Beanstalk\Command\IgnoreTube::`**`__construct`**(*$tube*)

Description Constructor

Parameters

- **$tube** (*string*) – Tube to remove from the watch list

`Beanstalk\Command\IgnoreTube::`**`getCommand`**()

Description Get the command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\IgnoreTube::`**`parseResponse`**(*$response*[, *$data = null*, *$conn = null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data received with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *integer* The number of tubes being watched

Throws *BeanstalkException* When the requested tube cannot be ignored

Throws *BeanstalkException* When any error occurs

1.2.10 Beanstalk\Command\Kick Class Ref

class `Beanstalk\Command\`**`Kick`**

Extends *`Beanstalk\Command`*

Description Kick command

Author Joshua Dechant <jdechant@shapeup.com>

The kick command applies only to the currently used tube. It moves jobs into the ready queue. If there are any buried jobs, it will only kick buried jobs. Otherwise it will kick delayed jobs

Class Methods

- *`Kick::__construct`* – Constructor
- *`Kick::getCommand`* – Get the delete command to send to the beanstalkd server
- *`Kick::parseResponse`* – Parse the response for success or failure.

`Beanstalk\Command\Kick::`**`__construct`**`(`*$bound*`)`

Description Constructor

Parameters

- **`$bound`** (*integer*) – Upper bound on the number of jobs to kick. The server will kick no more than $bound jobs.

`Beanstalk\Command\Kick::`**`getCommand`**`()`

Description Get the delete command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\Kick::`**`parseResponse`**`(`*$response*`[`, *$data = null*, *$conn = null* `]`)

Description Parse the response for success or failure.

Parameters

- **`$response`** (*string*) – Response line, i.e, first line in response
- **`$data`** (*string*) – Data recieved with reponse, if any, else null
- **`$conn`** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *integer* The number of jobs actually kicked

Throws *BeanstalkException* When any error occurs

1.2.11 Beanstalk\Command\ListTubes Class Ref

class `Beanstalk\Command\`**`ListTubes`**

Extends *`Beanstalk\Command`*

Description The list-tubes command returns a list of all existing tubes

Author Joshua Dechant <jdechant@shapeup.com>

Class Methods

- *ListTubes::getCommand* – Get the command to send to the beanstalkd server
- *ListTubes::parseResponse* – Parse the response for success or failure.
- *ListTubes::returnsData* – Does the command return data?

Beanstalk\Command\ListTubes::**getCommand**()

Description Get the command to send to the beanstalkd server

Returns *string*

Beanstalk\Command\ListTubes::**parseResponse**(*$response*[, *$data = null*, *$conn = null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *array* List of all existing tubes

Throws *BeanstalkException* When any error occurs

Beanstalk\Command\ListTubes::**returnsData**()

Description Does the command return data?

Returns *boolean*

1.2.12 Beanstalk\Command\PauseTube Class Ref

class Beanstalk\Command**PauseTube**

Extends *Beanstalk\Command*

Description The pause-tube command can delay any new job being reserved for a given time

Author Joshua Dechant <jdechant@shapeup.com>

Class Methods

- *PauseTube::__construct* – Constructor
- *PauseTube::getCommand* – Get the command to send to the beanstalkd server
- *PauseTube::parseResponse* – Parse the response for success or failure.

Beanstalk\Command\PauseTube::**__construct**(*$tube*, *$delay*)

Description Constructor

Parameters

- **$tube** (*string*) – The tube to pause
- **$delay** (*integer*) – Number of seconds to wait before reserving any more jobs from the queue

Beanstalk\Command\PauseTube::**getCommand**()

Description Get the command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\PauseTube::parseResponse` (*$response*[, *$data = null*, *$conn = null*])

Description Parse the response for success or failure.

Parameters

- **`$response`** (*string*) – Response line, i.e, first line in response
- **`$data`** (*string*) – Data recieved with reponse, if any, else null
- **`$conn`** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *boolean* True if command was successful

Throws *BeanstalkException* When the tube does not exist

Throws *BeanstalkException* When any other error occurs

1.2.13 Beanstalk\Command\Peek Class Ref

class `Beanstalk\Command\Peek`

Extends *`Beanstalk\Command`*

Description The peek commands let the client inspect a job in the system

Author Joshua Dechant <jdechant@shapeup.com>

There are four variations. All but the first operate only on the currently used tube.

- peek $id - return job $id
- ready - return the next ready job
- delayed - return the delayed job with the shortest delay left
- buried - return the next job in the list of buried jobs

Class Methods

- *`Peek::__construct`* – Constructor
- *`Peek::getCommand`* – Get the command to send to the beanstalkd server
- *`Peek::parseResponse`* – Parse the response for success or failure.
- *`Peek::returnsData`* – Does the command return data?

`Beanstalk\Command\Peek::__construct` (*$what*)

Description Constructor

Parameters

- **`$what`** (*mixed*) – What to peek. One of job id, “ready”, “delayed”, or “buried”

`Beanstalk\Command\Peek::getCommand` ()

Description Get the command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\Peek::parseResponse` (*$response*[, *$data = null*, *$conn = null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *BeanstalkJob*

Throws *BeanstalkException* When the job doesn't exist or there are no jobs in the requested state

Throws *BeanstalkException* When any other error occurs

`Beanstalk\Command\Peek::returnsData()`

Description Does the command return data?

Returns *boolean*

1.2.14 Beanstalk\Command\Put Class Ref

class `Beanstalk\Command\Put`

Extends *Beanstalk\Command*

Description The "put" command is for any process that wants to insert a job into the queue

Author Joshua Dechant <jdechant@shapeup.com>

Class Methods

- *Put::__construct* – Constructor
- *Put::getCommand* – Get the command to send to the beanstalkd server
- *Put::getData* – Get the data to send to the beanstalkd server with the command
- *Put::parseResponse* – Parse the response for success or failure.

`Beanstalk\Command\Put::__construct(`*$message*[, *$priority = 65536*, *$delay = 0*, *$ttr = 120*]`)`

Description Constructor

Parameters

- **$message** (*mixed*) – Message to put in the beanstalkd queue
- **$priority** (*integer*) – Job priority.
- **$delay** (*integer*) – Number of seconds to wait before putting the job in the ready queue.
- **$ttr** (*integer*) – Time to run. The number of seconds to allow a worker to run this job.

`Beanstalk\Command\Put::getCommand()`

Description Get the command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\Put::getData()`

Description Get the data to send to the beanstalkd server with the command

Returns *string*

Beanstalk\Command\Put::**parseResponse**(*$response*[, *$data* = *null*, *$conn* = *null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *integer* Id of the inserted job

Throws *BeanstalkException* When the server runs out of memory

Throws *BeanstalkException* When the job body is malformed

Throws *BeanstalkException* When the job body is larger than max-job-size in the server

Throws *BeanstalkException* When any other error occurs

1.2.15 Beanstalk\Command\Release Class Ref

class Beanstalk\Command**Release**

Extends *Beanstalk\Command*

Description Release command

Author Joshua Dechant <jdechant@shapeup.com>

The release command puts a reserved job back into the ready queue (and marks its state as "ready") to be run by any client. It is normally used when the job fails because of a transitory error.

Class Methods

- *Release::__construct* – Constructor
- *Release::getCommand* – Get the command to send to the beanstalkd server
- *Release::parseResponse* – Parse the response for success or failure.

Beanstalk\Command\Release::**__construct**(*$id*, *$priority*, *$delay*)

Description Constructor

Parameters

- **$id** (*integer*) – The job id to release
- **$priority** (*integer*) – A new priority to assign to the job
- **$delay** (*integer*) – Number of seconds to wait before putting the job in the ready queue.

Beanstalk\Command\Release::**getCommand**()

Description Get the command to send to the beanstalkd server

Returns *string*

Beanstalk\Command\Release::**parseResponse**(*$response*[, *$data* = *null*, *$conn* = *null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *boolean* True if command was successful

Throws *BeanstalkException* When the server runs out of memory

Throws *BeanstalkException* When the job cannot be found or has already timed out

Throws *BeanstalkException* When any other error occurs

1.2.16 Beanstalk\Command\Reserve Class Ref

class `Beanstalk\Command\`**`Reserve`**

Extends *`Beanstalk\Command`*

Description Reserve command

Author Joshua Dechant <jdechant@shapeup.com>

This will return a newly-reserved job. If no job is available to be reserved, beanstalkd will wait to send a response until one becomes available. Once a job is reserved for the client, the client has limited time to run (TTR) the job before the job times out. When the job times out, the server will put the job back into the ready queue. Both the TTR and the actual time left can be found in response to the stats-job command.

A timeout value of 0 will cause the server to immediately return either a response or TIMED_OUT. A positive value of timeout will limit the amount of time the client will block on the reserve request until a job becomes available.

Class Methods

- *`Reserve::__construct`* – Constructor
- *`Reserve::getCommand`* – Get the command to send to the beanstalkd server
- *`Reserve::parseResponse`* – Parse the response for success or failure.
- *`Reserve::returnsData`* – Does the command return data?

`Beanstalk\Command\Reserve::`**`__construct`**`(`[*$timeout = null*]`)`

Description Constructor

Parameters

- **$timeout** (*integer*) – Wait timeout in seconds

`Beanstalk\Command\Reserve::`**`getCommand`**`()`

Description Get the command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\Reserve::`**`parseResponse`**`(`*$response*[, *$data = null*, *$conn = null*]`)`

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null

- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *BeanstalkJob*

Throws *BeanstalkException* When trying to reserve another job and the TTR of the current job ends soon

Throws *BeanstalkException* When the wait timeout exceeded before a job became available

Throws *BeanstalkException* When any other error occurs

`Beanstalk\Command\Reserve::returnsData()`

Description Does the command return data?

Returns *boolean*

1.2.17 Beanstalk\Command\Stats Class Ref

class `Beanstalk\Command\Stats`

Extends *Beanstalk\Command*

Description The stats command gives statistical information about the system as a whole

Author Joshua Dechant <jdechant@shapeup.com>

Class Methods

- *Stats::getCommand* – Get the command to send to the beanstalkd server
- *Stats::parseResponse* – Parse the response for success or failure.
- *Stats::returnsData* – Does the command return data?

`Beanstalk\Command\Stats::getCommand()`

Description Get the command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\Stats::parseResponse(`*$response*[, *$data* = *null*, *$conn* = *null*]`)`

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *BeanstalkStats*

Throws *BeanstalkException* When any error occurs

`Beanstalk\Command\Stats::returnsData()`

Description Does the command return data?

Returns *boolean*

1.2.18 Beanstalk\Command\StatsJob Class Ref

class `Beanstalk\Command\`**`StatsJob`**

Extends *`Beanstalk\Command`*

Description The stats-job command gives statistical information about the specified job if it exists

Author Joshua Dechant <jdechant@shapeup.com>

Returned stats available:

- id: The job id
- tube: The name of the tube that contains this job
- state: One of "ready" or "delayed" or "reserved" or "buried"
- pri: The priority value set by the put, release, or bury commands.
- age: The time in seconds since the put command that created this job.
- **time-left: The number of seconds left until the server puts this job** into the ready queue. This number is only meaningful if the job is reserved or delayed. If the job is reserved and this amount of time elapses before its state changes, it is considered to have timed out.
- reserves: The number of times this job has been reserved.
- timeouts: The number of times this job has timed out during a reservation.
- releases: The number of times a client has released this job from a reservation.
- buries The number of times this job has been buried.
- kicks: The number of times this job has been kicked.

Class Methods

- *`StatsJob::__construct`* – Constructor
- *`StatsJob::getCommand`* – Get the command to send to the beanstalkd server
- *`StatsJob::parseResponse`* – Parse the response for success or failure.
- *`StatsJob::returnsData`* – Does the command return data?

`Beanstalk\Command\StatsJob::`**`__construct`**`(`*$id*`)`

Description Constructor

Parameters

- **`$id`** (*integer*) – Job id

`Beanstalk\Command\StatsJob::`**`getCommand`**`()`

Description Get the command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\StatsJob::`**`parseResponse`**`(`*$response*`[`, *$data = null*, *$conn = null*`]`)

Description Parse the response for success or failure.

Parameters

- **`$response`** (*string*) – Response line, i.e, first line in response
- **`$data`** (*string*) – Data recieved with reponse, if any, else null

- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *BeanstalkStats*

Throws *BeanstalkException* When the job does not exist

Throws *BeanstalkException* When any other error occurs

`Beanstalk\Command\StatsJob::returnsData()`

Description Does the command return data?

Returns *boolean*

1.2.19 Beanstalk\Command\StatsTube Class Ref

class `Beanstalk\Command\StatsTube`

Extends *Beanstalk\Command*

Description The stats-tube command gives statistical information about the specified tube if it exists

Class Methods

- *StatsTube::__construct* – Constructor
- *StatsTube::getCommand* – Get the command to send to the beanstalkd server
- *StatsTube::parseResponse* – Parse the response for success or failure.
- *StatsTube::returnsData* – Does the command return data?

`Beanstalk\Command\StatsTube::__construct(`*$tube*`)`

Description Constructor

Parameters

- **$tube** (*string*) – Stats will be returned for this tube.

Throws *BeanstalkException* When $tube exceeds 200 bytes

`Beanstalk\Command\StatsTube::getCommand()`

Description Get the command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\StatsTube::parseResponse(`*$response*[, *$data = null*, *$conn = null*]`)`

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *BeanstalkStats*

Throws *BeanstalkException* When the job does not exist

Throws *BeanstalkException* When any other error occurs

`Beanstalk\Command\StatsTube::returnsData()`

Description Does the command return data?

Returns *boolean*

1.2.20 Beanstalk\Command\Touch Class Ref

class `Beanstalk\Command\`**`Touch`**

Extends *`Beanstalk\Command`*

Description Touch command

Author Joshua Dechant <jdechant@shapeup.com>

The "touch" command allows a worker to request more time to work on a job. This is useful for jobs that potentially take a long time, but you still want the benefits of a TTR pulling a job away from an unresponsive worker. A worker may periodically tell the server that it's still alive and processing a job (e.g. it may do this on DEADLINE_SOON).

Class Methods

- *`Touch::__construct`* – Constructor
- *`Touch::getCommand`* – Get the command to send to the beanstalkd server
- *`Touch::parseResponse`* – Parse the response for success or failure.

`Beanstalk\Command\Touch::`**`__construct`**`(`*$id*`)`

Description Constructor

Parameters

- **`$id`** (*integer*) – The job id to touch

`Beanstalk\Command\Touch::`**`getCommand`**`()`

Description Get the command to send to the beanstalkd server

Returns *string*

`Beanstalk\Command\Touch::`**`parseResponse`**`(`*$response*`[`, *$data* = *null*, *$conn* = *null* `]`)

Description Parse the response for success or failure.

Parameters

- **`$response`** (*string*) – Response line, i.e, first line in response
- **`$data`** (*string*) – Data recieved with reponse, if any, else null
- **`$conn`** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *boolean* True if command was successful

Throws *BeanstalkException* When the job cannot be found or has already timed out

Throws *BeanstalkException* When any other error occurs

1.2.21 Beanstalk\Command\UseTube Class Ref

class `Beanstalk\Command\`**`UseTube`**

Extends *`Beanstalk\Command`*

Description Use command

Author Joshua Dechant <jdechant@shapeup.com>

The "use" command is for producers. Subsequent put commands will put jobs into the tube specified by this command. If no use command has been issued, jobs will be put into the tube named "default".

Class Methods

- *UseTube::__construct* – Constructor
- *UseTube::getCommand* – Get the command to send to the beanstalkd server
- *UseTube::parseResponse* – Parse the response for success or failure.

Beanstalk\Command\UseTube::**__construct**(*$tube*)

Description Constructor

Parameters

- **$tube** (*string*) – The tube to use. If the tube does not exist, it will be created.

Throws *BeanstalkException* When $tube exceeds 200 bytes

Beanstalk\Command\UseTube::**getCommand**()

Description Get the command to send to the beanstalkd server

Returns *string*

Beanstalk\Command\UseTube::**parseResponse**(*$response*[, *$data = null*, *$conn = null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *string* The name of the tube now being used

Throws *BeanstalkException* When any error occurs

1.2.22 Beanstalk\Command\WatchTube Class Ref

class Beanstalk\Command**WatchTube**

Extends *Beanstalk\Command*

Description Watch command

Author Joshua Dechant <jdechant@shapeup.com>

The "watch" command adds the named tube to the watch list for the current connection. A reserve command will take a job from any of the tubes in the watch list. For each new connection, the watch list initially consists of one tube, named "default".

Class Methods

- *WatchTube::__construct* – Constructor
- *WatchTube::getCommand* – Get the command to send to the beanstalkd server
- *WatchTube::parseResponse* – Parse the response for success or failure.

Beanstalk\Command\WatchTube::**__construct**(*$tube*)

Description Constructor

Parameters

- **$tube** (*string*) – Tube to add to the watch list. If the tube doesn't exist, it will be created

Throws *BeanstalkException* When $tube exceeds 200 bytes

Beanstalk\Command\WatchTube::**getCommand**()

Description Get the command to send to the beanstalkd server

Returns *string*

Beanstalk\Command\WatchTube::**parseResponse**(*$response*[, *$data = null*, *$conn = null*])

Description Parse the response for success or failure.

Parameters

- **$response** (*string*) – Response line, i.e, first line in response
- **$data** (*string*) – Data recieved with reponse, if any, else null
- **$conn** (*BeanstalkConnection*) – BeanstalkConnection use to send the command

Returns *integer* The number of tubes being watched

Throws *BeanstalkException* When any error occurs

1.2.23 Beanstalk\Connection\Stream Interface Ref

interface Beanstalk\Connection**Stream**

Class Methods

- *Stream::close* – Close the stream connection
- *Stream::isTimedOut* – Has the connection timed out or otherwise gone away?
- *Stream::open* – Open the stream
- *Stream::read* – Read the next $bytes bytes from the stream
- *Stream::readLine* – Read the next line from the stream
- *Stream::write* – Write data to the stream

Beanstalk\Connection\Stream::**close**()

Description Close the stream connection

Returns *null*

Beanstalk\Connection\Stream::**isTimedOut**()

Description Has the connection timed out or otherwise gone away?

Returns *boolean*

`Beanstalk\Connection\Stream::`**`open`**(*$host*, *$port*, *$timeout*)

Description Open the stream

Parameters

- **`$host`** (*string*) – Host or IP address to connect to
- **`$port`** (*integer*) – Port to connect on
- **`$timeout`** (*float*) – Connection timeout in milliseconds

Returns *boolean*

`Beanstalk\Connection\Stream::`**`read`**(*$bytes*)

Description Read the next $bytes bytes from the stream

Parameters

- **`$bytes`** (*integer*) – Number of bytes to read

Returns *string*

`Beanstalk\Connection\Stream::`**`readLine`**()

Description Read the next line from the stream

Returns *string*

`Beanstalk\Connection\Stream::`**`write`**(*$data*)

Description Write data to the stream

Parameters

- **`$data`** (*string*) –

Returns *integer* Number of bytes written

1.2.24 Beanstalk\Connection\Stream\Socket Class Ref

class `Beanstalk\Connection\Stream\`**`Socket`**

Implements `BeanstalkConnectionStream`

Description Connection stream using PHP native sockets

Author Joshua Dechant <jdechant@shapeup.com>

Class Methods

- *`Socket::close`* – Close the stream connection
- *`Socket::isTimedOut`* – Has the connection timed out or otherwise gone away?
- *`Socket::open`* – Open the stream
- *`Socket::read`* – Read the next $bytes bytes from the stream
- *`Socket::readLine`* – Read the next line from the stream
- *`Socket::write`* – Write data to the stream

`Beanstalk\Connection\Stream\Socket::`**`close`**()

Description Close the stream connection

Returns *null*

Beanstalk\Connection\Stream\Socket::**isTimedOut**()

Description Has the connection timed out or otherwise gone away?

Returns *boolean*

Beanstalk\Connection\Stream\Socket::**open**(*$host*, *$port*, *$timeout*)

Description Open the stream

Parameters

- **$host** (*string*) – Host or IP address to connect to
- **$port** (*integer*) – Port to connect on
- **$timeout** (*float*) – Connection timeout in milliseconds

Returns *boolean*

Beanstalk\Connection\Stream\Socket::**read**(*$bytes*)

Description Read the next $bytes bytes from the stream

Parameters

- **$bytes** (*integer*) – Number of bytes to read

Returns *string*

Beanstalk\Connection\Stream\Socket::**readLine**()

Description Read the next line from the stream

Returns *string*

Beanstalk\Connection\Stream\Socket::**write**(*$data*)

Description Write data to the stream

Parameters

- **$data** (*string*) –

Returns *integer* Number of bytes written

CHAPTER 2

Indices

- genindex
- search

b

Symbols

A

B

C

D

E

G

I

J

K

L

O

P

R

S

T

U

V

W