

Wave

0.1

Generated by Doxygen 1.9.1



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Ghoti::Wave::Blob Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 Blob() [1/2]	8
4.1.2.2 Blob() [2/2]	9
4.1.3 Member Function Documentation	9
4.1.3.1 append()	9
4.1.3.2 convertToFile()	9
4.1.3.3 getFile()	10
4.1.3.4 getText()	10
4.1.3.5 getType()	10
4.1.3.6 length()	11
4.1.3.7 operator==()	11
4.1.3.8 set() [1/2]	11
4.1.3.9 set() [2/2]	12
4.1.3.10 size()	12
4.1.3.11 truncate()	12
4.2 Ghoti::Wave::Client Class Reference	13
4.2.1 Detailed Description	15
4.2.2 Member Function Documentation	15
4.2.2.1 dispatchLoop()	15
4.2.2.2 getAllParameters()	15
4.2.2.3 getParameter()	15
4.2.2.4 getParameterAny()	16
4.2.2.5 getParameterDefault() [1/2]	16
4.2.2.6 getParameterDefault() [2/2]	17
4.2.2.7 isRunning()	17
4.2.2.8 sendRequest()	17
4.2.2.9 setParameter()	18
4.2.2.10 start()	18
4.2.2.11 stop()	19
4.2.3 Member Data Documentation	19
4.2.3.1 domains	19

4.3 Ghoti::Wave::ClientSession Class Reference	19
4.3.1 Detailed Description	21
4.3.2 Member Enumeration Documentation	21
4.3.2.1 Parameter	21
4.3.3 Constructor & Destructor Documentation	22
4.3.3.1 ClientSession()	22
4.3.4 Member Function Documentation	22
4.3.4.1 enqueue()	22
4.3.4.2 getAllParameters()	23
4.3.4.3 getParameter()	23
4.3.4.4 getParameterAny()	23
4.3.4.5 getParameterDefault() [1/2]	24
4.3.4.6 getParameterDefault() [2/2]	24
4.3.4.7 hasReadDataWaiting()	24
4.3.4.8 hasWriteDataWaiting()	25
4.3.4.9 isFinished()	25
4.3.4.10 read()	25
4.3.4.11 setParameter()	25
4.3.4.12 write()	26
4.3.5 Member Data Documentation	26
4.3.5.1 messages	26
4.3.5.2 readSequence	26
4.3.5.3 requestSequence	26
4.3.5.4 writeSequence	27
4.4 Ghoti::Wave::HasClientParameters Class Reference	27
4.4.1 Detailed Description	28
4.4.2 Member Function Documentation	28
4.4.2.1 getAllParameters()	28
4.4.2.2 getParameter()	28
4.4.2.3 getParameterAny()	29
4.4.2.4 getParameterDefault() [1/2]	29
4.4.2.5 getParameterDefault() [2/2]	30
4.4.2.6 setParameter()	30
4.5 Ghoti::Wave::HasParameters< T > Class Template Reference	30
4.5.1 Detailed Description	31
4.5.2 Member Function Documentation	32
4.5.2.1 getAllParameters()	32
4.5.2.2 getParameter()	32
4.5.2.3 getParameterAny()	33
4.5.2.4 getParameterDefault()	33
4.5.2.5 setParameter()	34
4.6 Ghoti::Wave::HasServerParameters Class Reference	34

4.6.1 Detailed Description	35
4.6.2 Member Function Documentation	35
4.6.2.1 getAllParameters()	36
4.6.2.2 getParameter()	36
4.6.2.3 getParameterAny()	36
4.6.2.4 getParameterDefault() [1/2]	37
4.6.2.5 getParameterDefault() [2/2]	37
4.6.2.6 setParameter()	37
4.7 Ghoti::Wave::Message Class Reference	38
4.7.1 Detailed Description	41
4.7.2 Member Enumeration Documentation	41
4.7.2.1 Transport	41
4.7.2.2 Type	41
4.7.3 Constructor & Destructor Documentation	42
4.7.3.1 Message()	42
4.7.4 Member Function Documentation	42
4.7.4.1 addFieldValue()	42
4.7.4.2 adoptContents()	42
4.7.4.3 getContentLength()	44
4.7.4.4 getDomain()	44
4.7.4.5 getFields()	44
4.7.4.6 getId()	44
4.7.4.7 getMessage()	45
4.7.4.8 getMessageBody()	45
4.7.4.9 getMethod()	45
4.7.4.10 getPort()	45
4.7.4.11 getReadySemaphore()	46
4.7.4.12 getRenderedHeader1()	46
4.7.4.13 getStatusCode()	47
4.7.4.14 getTarget()	47
4.7.4.15 getTransport()	47
4.7.4.16 getType()	47
4.7.4.17 getVersion()	48
4.7.4.18 hasError()	48
4.7.4.19 isFinished()	48
4.7.4.20 setDomain()	48
4.7.4.21 setErrorMessage()	49
4.7.4.22 setId()	49
4.7.4.23 setMessage()	49
4.7.4.24 setMessageBody()	50
4.7.4.25 setMethod()	50
4.7.4.26 setPort()	51

4.7.4.27 setReady()	51
4.7.4.28 setStatusCode()	52
4.7.4.29 setTarget()	52
4.7.4.30 setTransport()	52
4.7.4.31 setVersion()	53
4.7.5 Member Data Documentation	53
4.7.5.1 headers	53
4.7.5.2 parsingIsFinished	53
4.8 Ghoti::Wave::Parser Class Reference	54
4.8.1 Detailed Description	56
4.8.2 Member Enumeration Documentation	56
4.8.2.1 ReadStateMajor	56
4.8.2.2 ReadStateMinor	56
4.8.2.3 Type	57
4.8.3 Constructor & Destructor Documentation	57
4.8.3.1 Parser()	57
4.8.4 Member Function Documentation	58
4.8.4.1 createNewMessage()	58
4.8.4.2 getMEMCHUNKSIZELIMIT()	58
4.8.4.3 processChunk()	58
4.8.4.4 registerMessage()	59
4.8.5 Member Data Documentation	59
4.8.5.1 messageRegister	59
4.8.5.2 messages	59
4.9 Ghoti::Wave::RequestParser Class Reference	60
4.9.1 Detailed Description	62
4.9.2 Member Enumeration Documentation	62
4.9.2.1 ReadStateMajor	62
4.9.2.2 ReadStateMinor	63
4.9.2.3 Type	64
4.9.3 Member Function Documentation	64
4.9.3.1 createNewMessage()	64
4.9.3.2 getAllParameters()	64
4.9.3.3 getMEMCHUNKSIZELIMIT()	65
4.9.3.4 getParameter()	65
4.9.3.5 getParameterAny()	65
4.9.3.6 getParameterDefault() [1/2]	66
4.9.3.7 getParameterDefault() [2/2]	66
4.9.3.8 processChunk()	66
4.9.3.9 registerMessage()	67
4.9.3.10 setParameter()	67
4.9.4 Member Data Documentation	67

4.9.4.1 messageRegister . . . . .	68
4.9.4.2 messages . . . . .	68
4.10 Ghoti::Wave::ResponseParser Class Reference . . . . .	68
4.10.1 Detailed Description . . . . .	71
4.10.2 Member Enumeration Documentation . . . . .	71
4.10.2.1 ReadStateMajor . . . . .	71
4.10.2.2 ReadStateMinor . . . . .	71
4.10.2.3 Type . . . . .	72
4.10.3 Member Function Documentation . . . . .	72
4.10.3.1 createNewMessage() . . . . .	72
4.10.3.2 getAllParameters() . . . . .	73
4.10.3.3 getMEMCHUNKSIZELIMIT() . . . . .	73
4.10.3.4 getParameter() . . . . .	73
4.10.3.5 getParameterAny() . . . . .	74
4.10.3.6 getParameterDefault() [1/2] . . . . .	74
4.10.3.7 getParameterDefault() [2/2] . . . . .	74
4.10.3.8 processChunk() . . . . .	75
4.10.3.9 registerMessage() . . . . .	75
4.10.3.10 setParameter() . . . . .	75
4.10.4 Member Data Documentation . . . . .	76
4.10.4.1 messageRegister . . . . .	76
4.10.4.2 messages . . . . .	76
4.11 Ghoti::Wave::Server Class Reference . . . . .	77
4.11.1 Detailed Description . . . . .	79
4.11.2 Member Enumeration Documentation . . . . .	79
4.11.2.1 ErrorCode . . . . .	79
4.11.3 Constructor & Destructor Documentation . . . . .	79
4.11.3.1 Server() . . . . .	79
4.11.3.2 ~Server() . . . . .	80
4.11.4 Member Function Documentation . . . . .	80
4.11.4.1 clearError() . . . . .	80
4.11.4.2 dispatchLoop() . . . . .	80
4.11.4.3 getAddress() . . . . .	81
4.11.4.4 getAllParameters() . . . . .	81
4.11.4.5 getErrorCode() . . . . .	81
4.11.4.6 getErrorMessage() . . . . .	82
4.11.4.7 getParameter() . . . . .	82
4.11.4.8 getParameterAny() . . . . .	82
4.11.4.9 getParameterDefault() [1/2] . . . . .	83
4.11.4.10 getParameterDefault() [2/2] . . . . .	83
4.11.4.11 getPort() . . . . .	83
4.11.4.12 getSocketHandle() . . . . .	84

4.11.4.13 isRunning()	84
4.11.4.14 setAddress()	84
4.11.4.15 setParameter()	85
4.11.4.16 setPort()	85
4.11.4.17 start()	86
4.11.4.18 stop()	86
4.11.5 Member Data Documentation	86
4.11.5.1 sessions	87
4.12 Ghoti::Wave::ServerSession Class Reference	87
4.12.1 Detailed Description	89
4.12.2 Constructor & Destructor Documentation	89
4.12.2.1 ServerSession()	89
4.12.3 Member Function Documentation	89
4.12.3.1 getAllParameters()	89
4.12.3.2 getParameter()	90
4.12.3.3 getParameterAny()	90
4.12.3.4 getParameterDefault() [1/2]	90
4.12.3.5 getParameterDefault() [2/2]	91
4.12.3.6 hasReadDataWaiting()	91
4.12.3.7 hasWriteDataWaiting()	92
4.12.3.8 isFinished()	92
4.12.3.9 read()	92
4.12.3.10 setParameter()	92
4.12.3.11 write()	93
4.12.4 Member Data Documentation	93
4.12.4.1 messages	93
<b>5 File Documentation</b>	<b>95</b>
5.1 include/wave.hpp File Reference	95
5.1.1 Detailed Description	96
5.2 include/wave/blob.hpp File Reference	96
5.2.1 Detailed Description	97
5.2.2 Function Documentation	97
5.2.2.1 operator<<()	97
5.3 include/wave/client.hpp File Reference	98
5.3.1 Detailed Description	99
5.4 include/wave/clientSession.hpp File Reference	99
5.4.1 Detailed Description	100
5.5 include/wave/hasClientParameters.hpp File Reference	100
5.5.1 Detailed Description	101
5.5.2 Enumeration Type Documentation	101
5.5.2.1 ClientParameter	101



5.6 include/wave/hasParameters.hpp File Reference . . . . .	101
5.6.1 Detailed Description . . . . .	102
5.7 include/wave/hasServerParameters.hpp File Reference . . . . .	102
5.7.1 Detailed Description . . . . .	103
5.7.2 Enumeration Type Documentation . . . . .	103
5.7.2.1 ServerParameter . . . . .	103
5.8 include/wave/macros.hpp File Reference . . . . .	104
5.8.1 Detailed Description . . . . .	104
5.9 include/wave/message.hpp File Reference . . . . .	104
5.9.1 Detailed Description . . . . .	105
5.9.2 Function Documentation . . . . .	105
5.9.2.1 operator<<() . . . . .	105
5.10 include/wave/parser.hpp File Reference . . . . .	106
5.10.1 Detailed Description . . . . .	107
5.11 include/wave/parsing.hpp File Reference . . . . .	107
5.11.1 Detailed Description . . . . .	109
5.11.2 Function Documentation . . . . .	109
5.11.2.1 fieldValueEscape() . . . . .	109
5.11.2.2 fieldValueQuotesNeeded() . . . . .	110
5.11.2.3 isCRLFChar() . . . . .	110
5.11.2.4 isFieldContentChar() . . . . .	111
5.11.2.5 isFieldNameChar() . . . . .	112
5.11.2.6 isListField() . . . . .	112
5.11.2.7 isObsoleteTextChar() . . . . .	113
5.11.2.8 isQuotedChar() . . . . .	114
5.11.2.9 isTokenChar() . . . . .	115
5.11.2.10 isVisibleChar() . . . . .	115
5.11.2.11 isWhitespaceChar() . . . . .	116
5.12 include/wave/response.hpp File Reference . . . . .	116
5.12.1 Detailed Description . . . . .	117
5.13 include/wave/server.hpp File Reference . . . . .	117
5.13.1 Detailed Description . . . . .	118
5.14 include/wave/serverSession.hpp File Reference . . . . .	118
5.14.1 Detailed Description . . . . .	119
5.15 src/blob.cpp File Reference . . . . .	119
5.15.1 Detailed Description . . . . .	120
5.16 src/client.cpp File Reference . . . . .	120
5.16.1 Detailed Description . . . . .	121
5.17 src/clientSession.cpp File Reference . . . . .	121
5.17.1 Detailed Description . . . . .	121
5.18 src/hasClientParameters.cpp File Reference . . . . .	121
5.18.1 Detailed Description . . . . .	122

5.19 src/hasServerParameters.cpp File Reference . . . . .	122
5.19.1 Detailed Description . . . . .	123
5.20 src/message.cpp File Reference . . . . .	123
5.20.1 Detailed Description . . . . .	124
5.21 src/parser.cpp File Reference . . . . .	124
5.21.1 Detailed Description . . . . .	124
5.21.2 Macro Definition Documentation . . . . .	124
5.21.2.1 READ_CRLF_OPTIONAL . . . . .	125
5.21.2.2 READ_CRLF_REQUIRED . . . . .	125
5.21.2.3 READ_WHITESPACE_OPTIONAL . . . . .	125
5.21.2.4 READ_WHITESPACE_REQUIRED . . . . .	126
5.21.2.5 SET_MAJOR_STATE . . . . .	126
5.21.2.6 SET_MINOR_STATE . . . . .	126
5.21.2.7 SET_NEW_HEADER . . . . .	126
5.21.2.8 START_NEW_INPUT . . . . .	127
5.22 src/parsing.cpp File Reference . . . . .	127
5.22.1 Detailed Description . . . . .	128
5.22.2 Function Documentation . . . . .	128
5.22.2.1 isCRLFChar() . . . . .	128
5.22.2.2 isFieldContentChar() . . . . .	128
5.22.2.3 isFieldNameChar() . . . . .	129
5.22.2.4 isObsoleteTextChar() . . . . .	130
5.22.2.5 isQuotedChar() . . . . .	130
5.22.2.6 isTokenChar() . . . . .	131
5.22.2.7 isVisibleChar() . . . . .	132
5.22.2.8 isWhitespaceChar() . . . . .	132
5.23 src/response.cpp File Reference . . . . .	133
5.23.1 Detailed Description . . . . .	133
5.24 src/server.cpp File Reference . . . . .	133
5.24.1 Detailed Description . . . . .	134
5.25 src/serverSession.cpp File Reference . . . . .	134
5.25.1 Detailed Description . . . . .	134
5.26 test/test-hasParameters.cpp File Reference . . . . .	135
5.26.1 Detailed Description . . . . .	135
5.27 test/test.cpp File Reference . . . . .	135
5.27.1 Detailed Description . . . . .	136

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Ghoti::Wave::Blob . . . . .	7
Ghoti::Wave::HasParameters< T > . . . . .	30
Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter > . . . . .	30
Ghoti::Wave::HasClientParameters . . . . .	27
Ghoti::Wave::Client . . . . .	13
Ghoti::Wave::ClientSession . . . . .	19
Ghoti::Wave::ResponseParser . . . . .	68
Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter > . . . . .	30
Ghoti::Wave::HasServerParameters . . . . .	34
Ghoti::Wave::RequestParser . . . . .	60
Ghoti::Wave::Server . . . . .	77
Ghoti::Wave::ServerSession . . . . .	87
Ghoti::Wave::Message . . . . .	38
Ghoti::Wave::Parser . . . . .	54
Ghoti::Wave::RequestParser . . . . .	60
Ghoti::Wave::ResponseParser . . . . .	68



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Ghoti::Wave::Blob</a>	Generic container which may reference text (binary or otherwise) either in-memory or on-disk (e.g., in a file) . . . . .	7
<a href="#">Ghoti::Wave::Client</a>	Represents a client and all of its HTTP connections . . . . .	13
<a href="#">Ghoti::Wave::ClientSession</a>	Represents a connection to a particular domain/port pair . . . . .	19
<a href="#">Ghoti::Wave::HasClientParameters</a>	Base class to provide consistent defaults to <a href="#">Server</a> and <a href="#">ServerSession</a> classes . . . . .	27
<a href="#">Ghoti::Wave::HasParameters&lt; T &gt;</a>	Serves as a base class for any other class to have settings parameters . . . . .	30
<a href="#">Ghoti::Wave::HasServerParameters</a>	Base class to provide consistent defaults to <a href="#">Server</a> and <a href="#">ServerSession</a> classes . . . . .	34
<a href="#">Ghoti::Wave::Message</a>	Represents a HTTP message . . . . .	38
<a href="#">Ghoti::Wave::Parser</a>	Parses a HTTP/1.1 data stream into discrete messages . . . . .	54
<a href="#">Ghoti::Wave::RequestParser</a>	Specialized class for handling the request parser, to make it easier to pass in ServerParameters . . . . .	60
<a href="#">Ghoti::Wave::ResponseParser</a>	Specialized class for handling the response parser, to make it easier to pass in ClientParameters . . . . .	68
<a href="#">Ghoti::Wave::Server</a>	The base <a href="#">Server</a> class . . . . .	77
<a href="#">Ghoti::Wave::ServerSession</a>	Represents a persistent connection with a client . . . . .	87



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

include/wave.hpp	Header file supplied for use by 3rd party code so that they can easily include all necessary headers for the Ghoti.io Wave library . . . . .	95
include/wave/blob.hpp	Header file for declaring the Blob class . . . . .	96
include/wave/client.hpp	Header file for declaring the Client class . . . . .	98
include/wave/clientSession.hpp	Header file for declaring the ClientSession class . . . . .	99
include/wave/hasClientParameters.hpp	Header file for declaring the HasClientParameters class . . . . .	100
include/wave/hasParameters.hpp	Header file for declaring the hasParameters class . . . . .	101
include/wave/hasServerParameters.hpp	Header file for declaring the HasServerParameters class . . . . .	102
include/wave/macros.hpp	Header file for declaring the Client class . . . . .	104
include/wave/message.hpp	Header file for declaring the Message class . . . . .	104
include/wave/parser.hpp	Header file for declaring the Session class . . . . .	106
include/wave/parsing.hpp	Header file for declaring text parsing functions . . . . .	107
include/wave/response.hpp	Header file for declaring the Response class . . . . .	116
include/wave/server.hpp	Header file for declaring the Server class . . . . .	117
include/wave/serverSession.hpp	Header file for declaring the ServerSession class . . . . .	118
src/blob.cpp	Define the <code>Ghoti::Wave::Blob</code> class . . . . .	119
src/client.cpp	Define the <code>Ghoti::Wave::Client</code> class . . . . .	120
src/clientSession.cpp	Define the <code>Ghoti::Wave::ClientSession</code> class . . . . .	121

src/ <a href="#">hasClientParameters.cpp</a>	
Define the <a href="#">Ghoti::Wave::HasClientParameters</a> class . . . . .	121
src/ <a href="#">hasServerParameters.cpp</a>	
Define the <a href="#">Ghoti::Wave::HasServerParameters</a> class . . . . .	122
src/ <a href="#">message.cpp</a>	
Define the <a href="#">Ghoti::Wave::Message</a> class . . . . .	123
src/ <a href="#">parser.cpp</a>	
Define the <a href="#">Ghoti::Wave::Parser</a> class . . . . .	124
src/ <a href="#">parsing.cpp</a>	
Define the text parsing functions . . . . .	127
src/ <a href="#">response.cpp</a>	
Define the <a href="#">Ghoti::Wave::Response</a> class . . . . .	133
src/ <a href="#">server.cpp</a>	
Define the <a href="#">Ghoti::Wave::Server</a> class . . . . .	133
src/ <a href="#">serverSession.cpp</a>	
Define the <a href="#">Ghoti::Wave::ServerSession</a> class . . . . .	134
test/ <a href="#">test-hasParameters.cpp</a>	
Test the general Wave server behavior . . . . .	135
test/ <a href="#">test.cpp</a>	
Test the general Wave server behavior . . . . .	135



## Chapter 4

# Class Documentation

### 4.1 Ghoti::Wave::Blob Class Reference

The [Blob](#) class is a generic container which may reference text (binary or otherwise) either in-memory or on-disk (e.g., in a file).

```
#include <blob.hpp>
```

#### Public Types

- enum class **Type** { **TEXT** , **FILE** }

#### Public Member Functions

- [Blob](#) ()  
*The default constructor.*
- [Blob](#) (const Ghoti::shared\_string\_view &text)  
*Construct a [Blob](#) with a text representation.*
- [Blob](#) (Ghoti::OS::File &&file)  
*Construct a [Blob](#) with a file representation.*
- void [set](#) (Ghoti::shared\_string\_view &text)  
*Set the text contents of the [Blob](#).*
- void [set](#) (Ghoti::OS::File &&file)  
*Set the file contents of the [Blob](#).*
- uint32\_t [size](#) () const noexcept  
*Get the size of the text in the blob.*
- uint32\_t [length](#) () const noexcept  
*Alias for [Blob.size\(\)](#).*
- const Ghoti::shared\_string\_view & [getText](#) () const  
*Get the text in the blob.*
- const Ghoti::OS::File & [getFile](#) () const  
*Get the file in the blob.*
- Ghoti::Wave::Blob::Type [getType](#) () const  
*Get the Ghoti::Wave::Blob::Type of data the blob contains.*
- bool [operator==](#) (const Ghoti::shared\_string\_view &rhs) const

*Compare a file against a string.*

- `std::error_code` [append](#) (const Ghoti::shared\_string\_view &[text](#))

*Append text to the current [Blob](#) object.*

- `std::error_code` [truncate](#) (const Ghoti::shared\_string\_view &[text](#))

*Truncate text in the current [Blob](#) object and replace it with the supplied text.*

- `std::error_code` [convertToFile](#) ()

*Convert the [Blob](#) object to be file-based.*

## Private Attributes

- Ghoti::Wave::Blob::Type [type](#)

*The type of data the blob contains.*

- Ghoti::shared\_string\_view [text](#)

*The text data the blob contains.*

- Ghoti::OS::File [file](#)

*The file data the blob contains.*

### 4.1.1 Detailed Description

The [Blob](#) class is a generic container which may reference text (binary or otherwise) either in-memory or on-disk (e.g., in a file).

The [Blob](#) provides a mechanism to append to the text (whether in-memory or on-disk), and to convert the in-memory text to a file.

The purpose of this container is so that, if text is growing too large (a threshold decision which is left up to the programmer, and is not a part of this object), then it can be converted to disk storage. The disk storage is likewise specifically useful for our approach to HTTP messages. In short, [Blob](#) makes use of the Ghoti::OS::File object, which will put files into the OS temp directory by default, and clean up after itself when the file object goes out of scope (if not handled in other ways).

In short: large text is possibly a file and is moved to the disk so that RAM is not wasted while either waiting on requests to arrive, or when preparing responses. The end result is that it eliminates the need to store the entire file in memory when the disk is available.

Blobs are used for all message types, including chunked and multipart messages (each chunk/part is its own [Blob](#), and may be either in-memory or on-disk).

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 [Blob\(\)](#) [1/2]

```
Blob::Blob (
    const Ghoti::shared_string_view & text )
```

Construct a [Blob](#) with a text representation.

## Parameters

<i>text</i>	The text the blob should contain.
-------------	-----------------------------------

#### 4.1.2.2 Blob() [2/2]

```
Blob::Blob (
    Ghoti::OS::File && file )
```

Construct a [Blob](#) with a file representation.

## Parameters

<i>file</i>	The file the blob should contain.
-------------	-----------------------------------

### 4.1.3 Member Function Documentation

#### 4.1.3.1 append()

```
error_code Blob::append (
    const Ghoti::shared_string_view & text )
```

Append text to the current [Blob](#) object.

The supplied text will be added to the end of any currently existing text.

## Parameters

<i>text</i>	The text to be appended.
-------------	--------------------------

## Returns

The error code resulting from the operation (if any).

#### 4.1.3.2 convertToFile()

```
error_code Blob::convertToFile ( )
```

Convert the [Blob](#) object to be file-based.

If the [Blob](#) is already file-based, no error will be returned.

**Returns**

The error code resulting from the operation (if any).

**4.1.3.3 getFile()**

```
const Ghoti::OS::File & Blob::getFile ( ) const
```

Get the file in the blob.

If the [Blob](#) is a text blob, then the file will be empty.

**Returns**

The file in the blob.

**4.1.3.4 getText()**

```
const Ghoti::shared_string_view & Blob::getText ( ) const
```

Get the text in the blob.

If the [Blob](#) is a file blob, then the text will be empty.

**Returns**

The text in the blob.

**4.1.3.5 getType()**

```
Blob::Type Blob::getType ( ) const
```

Get the `Ghoti::Wave::Blob::Type` of data the blob contains.

**Returns**

The `Ghoti::Wave::Blob::Type` of data the blob contains.

#### 4.1.3.6 length()

```
uint32_t Blob::length ( ) const [noexcept]
```

Alias for [Blob.size\(\)](#).

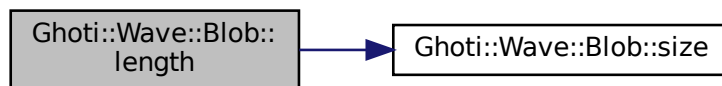
Get the size of the text in the blob.

If the file operation encounters an error, then it will return a size of 0. It is up to the caller to investigate to see if there is a problem with the file.

##### Returns

The size of the text in bytes.

Here is the call graph for this function:



#### 4.1.3.7 operator==()

```
bool Blob::operator== (
    const Ghoti::shared_string_view & rhs ) const
```

Compare a file against a string.

##### Parameters

<i>rhs</i>	The string to compare against.
------------	--------------------------------

##### Returns

True if the values are equivalent, False otherwise.

#### 4.1.3.8 set() [1/2]

```
void Blob::set (
    Ghoti::OS::File && file )
```

Set the file contents of the [Blob](#).

This will replace any current contents, whether file or text.

#### Parameters

<i>file</i>	The file the blob should contain.
-------------	-----------------------------------

#### 4.1.3.9 `set()` [2/2]

```
void Blob::set (
    Ghoti::shared_string_view & text )
```

Set the text contents of the [Blob](#).

This will replace any current contents, whether file or text.

#### Parameters

<i>text</i>	The text the blob should contain.
-------------	-----------------------------------

#### 4.1.3.10 `size()`

```
uint32_t Blob::size ( ) const [noexcept]
```

Get the size of the text in the blob.

If the file operation encounters an error, then it will return a size of 0. It is up to the caller to investigate to see if there is a problem with the file.

#### Returns

The size of the text in bytes.

#### 4.1.3.11 `truncate()`

```
error_code Blob::truncate (
    const Ghoti::shared_string_view & text )
```

Truncate text in the current [Blob](#) object and replace it with the supplied text.

## Parameters

<i>text</i>	The text to be written after the truncation..
-------------	---

## Returns

The error code resulting from the operation (if any).

The documentation for this class was generated from the following files:

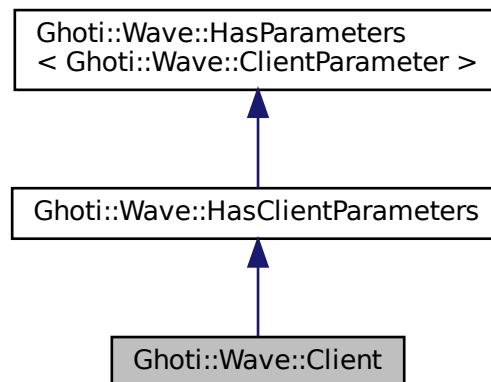
- [include/wave/blob.hpp](#)
- [src/blob.cpp](#)

## 4.2 Ghoti::Wave::Client Class Reference

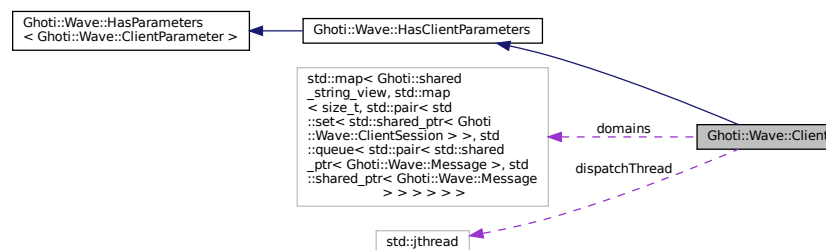
Represents a client and all of its HTTP connections.

```
#include <client.hpp>
```

Inheritance diagram for Ghoti::Wave::Client:



Collaboration diagram for Ghoti::Wave::Client:



## Public Member Functions

- [Client](#) ()  
*The constructor.*
- [~Client](#) ()  
*The destructor.*
- bool [isRunning](#) () const  
*Indicates whether or not the client and its thread pools are currently active.*
- [Client](#) & [start](#) ()  
*Instructs the client to start its thread pool and begin processing the requests in its queue.*
- [Client](#) & [stop](#) ()  
*Instructs the client to gracefully shut down its thread pool.*
- void [dispatchLoop](#) (std::stop\_token token)  
*The dispatch loop used by the thread pool to process sending requests and receiving responses.*
- std::shared\_ptr< [Message](#) > [sendRequest](#) (std::shared\_ptr< [Message](#) > message)  
*Enqueues a message to be sent to a client.*
- virtual [Client](#) & [setParameter](#) (const [ClientParameter](#) &parameter, const std::any &value) override  
*Set a parameter.*
- virtual std::optional< std::any > [getParameterDefault](#) (const [Ghoti::Wave::ClientParameter](#) &parameter) override  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterDefault](#) ([[maybe\_unused]]const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterAny](#) (const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Gets the named parameter if it exists, checking locally first, then checking the global defaults.*
- const std::optional< U > [getParameter](#) (const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Get the parameter as a specified type.*
- const [ParameterMap](#)< [Ghoti::Wave::ClientParameter](#) > & [getAllParameters](#) () const  
*Get the current explicitly-set parameters and their values.*

## Private Attributes

- [Ghoti::Pool::Pool](#) [workers](#)  
*The thread pool worker queue.*
- std::map< [Ghoti::shared\\_string\\_view](#), std::map< size\_t, std::pair< std::set< std::shared\_ptr< [Ghoti::Wave::ClientSession](#) > >, std::queue< std::pair< std::shared\_ptr< [Message](#) >, std::shared\_ptr< [Message](#) > > > > > >  
[domains](#)  
*Stores all connections and their request queues.*
- std::jthread [dispatchThread](#)  
*The thread that runs the read/write processing queues.*
- bool [running](#)  
*Whether or not the client is processing read/write actions from the sockets.*
- [ParameterMap](#)< [Ghoti::Wave::ClientParameter](#) > [parameterValues](#)  
*Store explicitly set parameter key/value pairs.*



### 4.2.1 Detailed Description

Represents a client and all of its HTTP connections.

This class is currently only used for testing (so that the HTTP connection can be controlled explicitly), but that does not mean that it can't be used for more.

The [Client](#) object can establish connections to a server, receive message requests and forward them to the appropriate session, and report on the status of the connections.

This class exists primarily for testing the server, and as such offers fine-grained control of enabling and disabling features.

### 4.2.2 Member Function Documentation

#### 4.2.2.1 dispatchLoop()

```
void Client::dispatchLoop (
    std::stop_token token )
```

The dispatch loop used by the thread pool to process sending requests and receiving responses.

##### Parameters

<i>token</i>	The jthread stop token, used to alert the thread that it should gracefully shut down.
--------------	---

#### 4.2.2.2 getAllParameters()

```
const ParameterMap<Ghoti::Wave::ClientParameter >& Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >::getAllParameters ( ) const [inline], [inherited]
```

Get the current explicitly-set parameters and their values.

##### Returns

The current explicitly-set parameters.

#### 4.2.2.3 getParameter()

```
const std::optional<U> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >::get<U>
Parameter (
    const Ghoti::Wave::ClientParameter & parameter ) [inline], [inherited]
```

Get the parameter as a specified type.

The result is returned as an optional. If there is no parameter value, then the optional value will be false.

**Parameters**

<i>parameter</i>	The parameter value to get.
------------------	-----------------------------

**Returns**

The (optional) parameter value.

**4.2.2.4 getParameterAny()**

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >↔
::getParameterAny (
    const Ghoti::Wave::ClientParameter & parameter ) [inline], [virtual], [inherited]
```

Gets the named parameter if it exists, checking locally first, then checking the global defaults.

**Parameters**

<i>parameter</i>	The parameter to get.
------------------	-----------------------

**Returns**

The parameter value if it exists.

**4.2.2.5 getParameterDefault() [1/2]**

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >↔
::getParameterDefault (
    [[maybe_unused]] const Ghoti::Wave::ClientParameter & parameter ) [inline], [virtual],
[inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

**Parameters**

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

**Returns**

The associated value.

#### 4.2.2.6 getParameterDefault() [2/2]

```
optional< any > HasClientParameters::getParameterDefault (
    const Ghoti::Wave::ClientParameter & parameter ) [override], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

##### Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

##### Returns

The associated value.

#### 4.2.2.7 isRunning()

```
bool Client::isRunning ( ) const
```

Indicates whether or not the client and its thread pools are currently active.

##### Returns

Whether or not the client and its thread pools are currently active.

#### 4.2.2.8 sendRequest()

```
shared_ptr< Message > Client::sendRequest (
    std::shared_ptr< Message > message )
```

Enqueues a message to be sent to a client.

This returns a shared pointer to a [Message](#) which will contain the response when the request is completed.

##### Parameters

<i>message</i>	The request to be sent to a client.
----------------	-------------------------------------

##### Returns

A shared pointer to a [Message](#) the will eventually contain the response when the request is completed.

#### 4.2.2.9 setParameter()

```
Client & Client::setParameter (
    const ClientParameter & parameter,
    const std::any & value ) [override], [virtual]
```

Set a parameter.

Values will be propagated to all client sessions.

##### Parameters

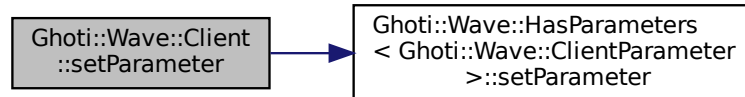
<i>parameter</i>	The parameter key to be set.
<i>value</i>	The parameter value to be set.

##### Returns

The calling object, to allow for chaining.

Reimplemented from [Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >](#).

Here is the call graph for this function:



#### 4.2.2.10 start()

```
Client& Ghoti::Wave::Client::start ( )
```

Instructs the client to start its thread pool and begin processing the requests in its queue.

##### Returns

The [Client](#) object.

#### 4.2.2.11 stop()

```
Client & Client::stop ( )
```

Instructs the client to gracefully shut down its thread pool.

Returns

The [Client](#) object.

### 4.2.3 Member Data Documentation

#### 4.2.3.1 domains

```
std::map<Ghoti::shared_string_view, std::map<size_t, std::pair<std::set<std::shared_ptr<Ghoti::Wave::ClientSession>, std::queue<std::pair<std::shared_ptr<Message>, std::shared_ptr<Message> > > > > > Ghoti::Wave::Client::domains [private]
```

Stores all connections and their request queues.

```
domains[domain][port] = {set{ClientSession}, queue{{request, response}}}
```

The documentation for this class was generated from the following files:

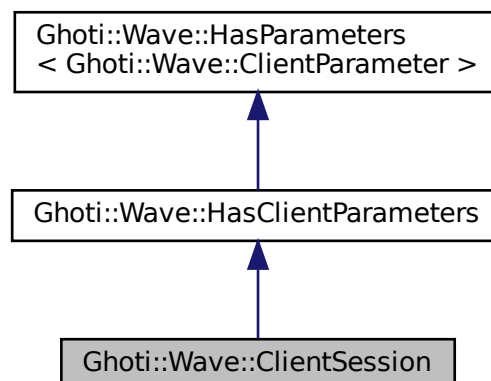
- include/wave/client.hpp
- src/client.cpp

## 4.3 Ghoti::Wave::ClientSession Class Reference

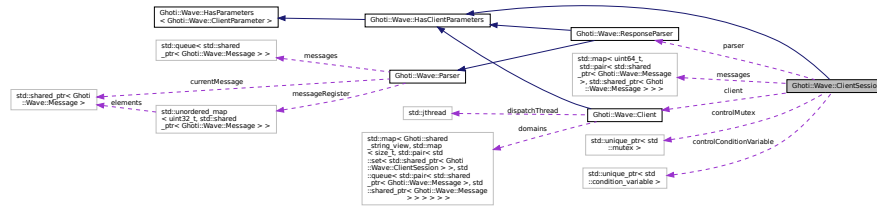
Represents a connection to a particular domain/port pair.

```
#include <clientSession.hpp>
```

Inheritance diagram for Ghoti::Wave::ClientSession:



Collaboration diagram for Ghoti::Wave::ClientSession:



## Public Types

- enum class [Parameter](#) { [MAXBUFFERSIZE](#) }  
*Sessings parameters which influence the behavior of Wave and its components.*

## Public Member Functions

- [ClientSession](#) (int [hServer](#), [Client](#) \*[client](#))  
*The constructor.*
- [~ClientSession](#) ()  
*The destructor.*
- bool [hasReadDataWaiting](#) ()  
*Checks to see whether or not the session has data waiting to be read from the socket.*
- bool [hasWriteDataWaiting](#) ()  
*Checks to see whether or not the session has data waiting to be written to the socket.*
- bool [isFinished](#) ()  
*Indicates whether or not the session has completed all communications and may be terminated.*
- void [read](#) ()  
*Performs a read from the session.*
- void [write](#) ()  
*Performs a write to the session.*
- void [enqueue](#) (std::shared\_ptr< [Message](#) > request, std::shared\_ptr< [Message](#) > response)  
*Add a request/response pair to the session's queue.*
- virtual std::optional< std::any > [getParameterDefault](#) (const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterDefault](#) ([[maybe\_unused]]const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterAny](#) (const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Gets the named parameter if it exists, checking locally first, then checking the global defaults.*
- const std::optional< U > [getParameter](#) (const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Get the parameter as a specified type.*
- virtual [HasParameters](#) & [setParameter](#) (const [Ghoti::Wave::ClientParameter](#) &parameter, const std::any &value)  
*Set a parameter.*
- const [ParameterMap](#)< [Ghoti::Wave::ClientParameter](#) > & [getAllParameters](#) () const  
*Get the current explicitly-set parameters and their values.*

## Public Attributes

- `std::unique_ptr< std::mutex >` [controlMutex](#)  
*Used to synchronize access to the session to make it thread safe.*
- `std::unique_ptr< std::condition_variable >` [controlConditionVariable](#)  
*Used to synchronize access to the session to make it thread safe.*

## Private Attributes

- `int` [hServer](#)  
*The socket handle to the server.*
- `size_t` [requestSequence](#)  
*The index number of the next request to be enqueued.*
- `size_t` [writeSequence](#)  
*The index number of the current request being written.*
- `size_t` [writeOffset](#)  
*A byte offset, used to track how many bytes of a message have been written, so that individual write attempts do not duplicate data.*
- `size_t` [readSequence](#)  
*The index number of the current request being received.*
- `bool` [working](#)  
*Tracks whether or not the session has work queued.*
- `bool` [finished](#)  
*Tracks whether or not the session has completed all pending communications.*
- [ResponseParser](#) [parser](#)  
*The parser object used to parse the raw HTTP stream.*
- `Client *` [client](#)  
*A pointer to the client object.*
- `std::map< uint64_t, std::pair< std::shared_ptr< Message >, std::shared_ptr< Message > > >` [messages](#)  
*Tracks message/response pairs.*
- [ParameterMap](#)< [Ghoti::Wave::ClientParameter](#) > [parameterValues](#)  
*Store explicitly set parameter key/value pairs.*

### 4.3.1 Detailed Description

Represents a connection to a particular domain/port pair.

### 4.3.2 Member Enumeration Documentation

#### 4.3.2.1 Parameter

```
enum Ghoti::Wave::ClientSession::Parameter [strong]
```

Sessings parameters which influence the behavior of Wave and its components.

## Enumerator

MAXBUFFERSIZE	The read/write buffer size used when interacting with sockets.
---------------	--

### 4.3.3 Constructor & Destructor Documentation

#### 4.3.3.1 ClientSession()

```
ClientSession::ClientSession (
    int hServer,
    Client * client )
```

The constructor.

The parent [Client](#) object will do the work of establishing the socket connection. Once the connection is established, then this class takes over the communication.

## Parameters

<i>hServer</i>	The socket handle to the <a href="#">Server</a> to which this session will communicate.
<i>client</i>	A pointer to the parent <a href="#">Client</a> object.

### 4.3.4 Member Function Documentation

#### 4.3.4.1 enqueue()

```
void ClientSession::enqueue (
    std::shared_ptr< Message > request,
    std::shared_ptr< Message > response )
```

Add a request/response pair to the session's queue.

The response object was created by the [Client](#), and we will write our results into it as the request is processed.

## Parameters

<i>request</i>	The HTTP request <a href="#">Message</a> .
<i>response</i>	The HTTP response <a href="#">Message</a> .



#### 4.3.4.2 getAllParameters()

```
const ParameterMap<Ghoti::Wave::ClientParameter> & Ghoti::Wave::HasParameters<Ghoti::Wave::ClientParameter>::getAllParameters ( ) const [inline], [inherited]
```

Get the current explicitly-set parameters and their values.

##### Returns

The current explicitly-set parameters.

#### 4.3.4.3 getParameter()

```
const std::optional<U> Ghoti::Wave::HasParameters<Ghoti::Wave::ClientParameter>::get↔  
Parameter (   
    const T & parameter ) [inline], [inherited]
```

Get the parameter as a specified type.

The result is returned as an optional. If there is no parameter value, then the optional value will be false.

##### Parameters

<i>parameter</i>	The parameter value to get.
------------------	-----------------------------

##### Returns

The (optional) parameter value.

#### 4.3.4.4 getParameterAny()

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters<Ghoti::Wave::ClientParameter>↔  
::getParameterAny (   
    const T & parameter ) [inline], [virtual], [inherited]
```

Gets the named parameter if it exists, checking locally first, then checking the global defaults.

##### Parameters

<i>parameter</i>	The parameter to get.
------------------	-----------------------

##### Returns

The parameter value if it exists.

#### 4.3.4.5 `getParameterDefault()` [1/2]

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >↔
::getParameterDefault (
    [[maybe_unused]] const T & parameter ) [inline], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

##### Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

##### Returns

The associated value.

#### 4.3.4.6 `getParameterDefault()` [2/2]

```
optional< any > HasClientParameters::getParameterDefault (
    const Ghoti::Wave::ClientParameter & parameter ) [override], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

##### Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

##### Returns

The associated value.

#### 4.3.4.7 `hasReadDataWaiting()`

```
bool ClientSession::hasReadDataWaiting ( )
```

Checks to see whether or not the session has data waiting to be read from the socket.

This is non-blocking mutex controlled. If the session is currently working, then this function will return false.

##### Returns

Whether or not the session has data waiting to be read from the socket.

**4.3.4.8 hasWriteDataWaiting()**

```
bool ClientSession::hasWriteDataWaiting ( )
```

Checks to see whether or not the session has data waiting to be written to the socket.

This is non-blocking mutex controlled. If the session is currently working, then this function will return false.

**Returns**

Whether or not the session has data waiting to be written to the socket.

**4.3.4.9 isFinished()**

```
bool ClientSession::isFinished ( )
```

Indicates whether or not the session has completed all communications and may be terminated.

**Returns**

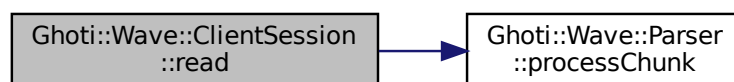
true if all communications have completed, false otherwise.

**4.3.4.10 read()**

```
void ClientSession::read ( )
```

Performs a read from the session.

This function is intended to be called by the client session's dispatch thread. Here is the call graph for this function:

**4.3.4.11 setParameter()**

```
virtual HasParameters& Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >::setParameter
(
    const Ghoti::Wave::ClientParameter & parameter,
    const std::any & value ) [inline], [virtual], [inherited]
```

Set a parameter.

**Parameters**

<i>parameter</i>	The parameter key to be set.
<i>value</i>	The parameter value to be set.

**Returns**

The calling object, to allow for chaining.

Reimplemented in [Ghoti::Wave::Client](#).

**4.3.4.12 write()**

```
void ClientSession::write ( )
```

Performs a write to the session.

This function is intended to be called by the client session's dispatch thread.

**4.3.5 Member Data Documentation****4.3.5.1 messages**

```
std::map<uint64_t, std::pair<std::shared_ptr<Message>, std::shared_ptr<Message> > > Ghoti←  
::Wave::ClientSession::messages [private]
```

Tracks message/response pairs.

messages[request sequence #] = <request, response>

**4.3.5.2 readSequence**

```
size_t Ghoti::Wave::ClientSession::readSequence [private]
```

The index number of the current request being received.

A session may send many requests before a single response is completely received. This variable tracks the response order so that it can be paired to the correct request.

**4.3.5.3 requestSequence**

```
size_t Ghoti::Wave::ClientSession::requestSequence [private]
```

The index number of the next request to be enqueued.

A session may have multiple messages enqueued before the connection has been established. This variable ensures that messages are handled in the order requested.

## 4.3.5.4 writeSequence

```
size_t Ghoti::Wave::ClientSession::writeSequence [private]
```

The index number of the current request being written.

A session must send requests in the order that they were enqueued. This variable tracks which message will be sent next.

The documentation for this class was generated from the following files:

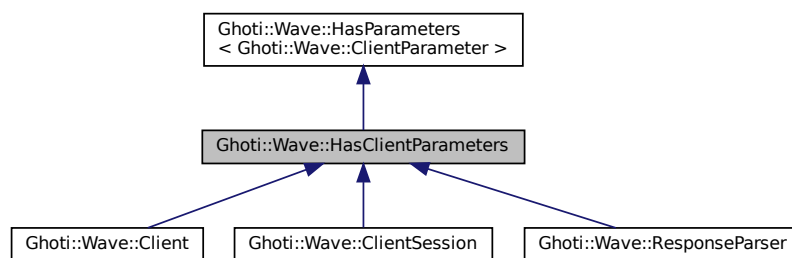
- [include/wave/clientSession.hpp](#)
- [src/clientSession.cpp](#)

## 4.4 Ghoti::Wave::HasClientParameters Class Reference

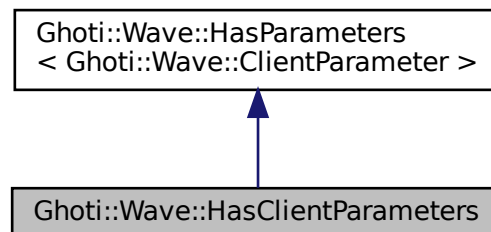
Base class to provide consistent defaults to [Server](#) and [ServerSession](#) classes.

```
#include <hasClientParameters.hpp>
```

Inheritance diagram for Ghoti::Wave::HasClientParameters:



Collaboration diagram for Ghoti::Wave::HasClientParameters:



## Public Member Functions

- virtual `std::optional< std::any > getParameterDefault` (const [Ghoti::Wave::ClientParameter](#) &parameter) override  
*Provide a default value for the provided parameter key.*
- virtual `std::optional< std::any > getParameterDefault` ([[maybe\_unused]]const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Provide a default value for the provided parameter key.*
- virtual `std::optional< std::any > getParameterAny` (const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Gets the named parameter if it exists, checking locally first, then checking the global defaults.*
- const `std::optional< U > getParameter` (const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Get the parameter as a specified type.*
- virtual `HasParameters & setParameter` (const [Ghoti::Wave::ClientParameter](#) &parameter, const `std::any` &value)  
*Set a parameter.*
- const `ParameterMap< Ghoti::Wave::ClientParameter > & getAllParameters` () const  
*Get the current explicitly-set parameters and their values.*

## Private Attributes

- `ParameterMap< Ghoti::Wave::ClientParameter > parameterValues`  
*Store explicitly set parameter key/value pairs.*

### 4.4.1 Detailed Description

Base class to provide consistent defaults to [Server](#) and [ServerSession](#) classes.

### 4.4.2 Member Function Documentation

#### 4.4.2.1 `getAllParameters()`

```
const ParameterMap<Ghoti::Wave::ClientParameter >& Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >::getAllParameters ( ) const [inline], [inherited]
```

Get the current explicitly-set parameters and their values.

#### Returns

The current explicitly-set parameters.

#### 4.4.2.2 `getParameter()`

```
const std::optional<U> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >::get↔  
Parameter (   
    const T & parameter ) [inline], [inherited]
```

Get the parameter as a specified type.

The result is returned as an optional. If there is no parameter value, then the optional value will be false.

## Parameters

<i>parameter</i>	The parameter value to get.
------------------	-----------------------------

## Returns

The (optional) parameter value.

4.4.2.3 `getParameterAny()`

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >↔
::getParameterAny (
    const T & parameter ) [inline], [virtual], [inherited]
```

Gets the named parameter if it exists, checking locally first, then checking the global defaults.

## Parameters

<i>parameter</i>	The parameter to get.
------------------	-----------------------

## Returns

The parameter value if it exists.

4.4.2.4 `getParameterDefault()` [1/2]

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >↔
::getParameterDefault (
    [[maybe_unused]] const T & parameter ) [inline], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

## Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

## Returns

The associated value.

#### 4.4.2.5 `getParameterDefault()` [2/2]

```
optional< any > HasClientParameters::getParameterDefault (
    const Ghoti::Wave::ClientParameter & parameter ) [override], [virtual]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

##### Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

##### Returns

The associated value.

#### 4.4.2.6 `setParameter()`

```
virtual HasParameters& Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >::setParameter
(
    const T & parameter,
    const std::any & value ) [inline], [virtual], [inherited]
```

Set a parameter.

##### Parameters

<i>parameter</i>	The parameter key to be set.
<i>value</i>	The parameter value to be set.

##### Returns

The calling object, to allow for chaining.

Reimplemented in [Ghoti::Wave::Client](#).

The documentation for this class was generated from the following files:

- [include/wave/hasClientParameters.hpp](#)
- [src/hasClientParameters.cpp](#)

## 4.5 `Ghoti::Wave::HasParameters< T >` Class Template Reference

Serves as a base class for any other class to have settings parameters.

```
#include <hasParameters.hpp>
```



## Public Member Functions

- [HasParameters](#) ()  
*The constructor.*
- virtual std::optional< std::any > [getParameterDefault](#) ([[maybe\_unused]]const T &parameter)  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterAny](#) (const T &parameter)  
*Gets the named parameter if it exists, checking locally first, then checking the global defaults.*
- template<class U >  
const std::optional< U > [getParameter](#) (const T &parameter)  
*Get the parameter as a specified type.*
- virtual [HasParameters](#) & [setParameter](#) (const T &parameter, const std::any &value)  
*Set a parameter.*
- const [ParameterMap](#)< T > & [getAllParameters](#) () const  
*Get the current explicitly-set parameters and their values.*

## Private Attributes

- [ParameterMap](#)< T > [parameterValues](#)  
*Store explicitly set parameter key/value pairs.*

### 4.5.1 Detailed Description

```
template<typename T>
class Ghoti::Wave::HasParameters< T >
```

Serves as a base class for any other class to have settings parameters.

[HasParameters](#) is a templated utility class. It's purpose is to associate key/value pairs as settings, in which the keys are of an enum type and the values may be of any type.

In order to use this class, the programmer must supply one or two things.

1. An enum or enum class type (with values defined, of course).
2. (Optional) A function to supply default values.

The default value function provided by the class will only provide an empty object, but the programmer may create a subclass to override this behavior.

A simple example of the usage of this class can be seen below:

```
enum class Foo {
    GIMME_A_INT,
    GIMME_A_STRING,
};
class FooWithDefaults : public Ghoti::Wave::HasParameters<Foo> {
public:
    optional<any> getParameterDefault(const Foo & p) {
        if (p == Foo::GIMME_A_INT) {
            return int{1};
        }
        if (p == Foo::GIMME_A_STRING) {
            return string{"foo"};
        }
        return {};
    }
};
```

Alternate example of FooWithDefaults declaration:

```
class FooWithDefaults : public Ghoti::Wave::HasParameters<Foo> {
public:
    optional<any> getParameterDefault(const Foo & p) {
        unordered_map<Foo, optional<any>> defaults{
            {Foo::GIMME_A_INT, {int{1}}},
            {Foo::GIMME_A_STRING, {string{"foo"}}},
        };
        return defaults.contains(p) ? defaults[p] : {};
    }
};
```

To Use it:

```
int main() {
    FooWithDefaults f{};
    cout << *f.getParameter<uint32_t>(Foo::GIMME_A_INT) << endl;
    return 0;
}
```

Remember that `getParameter()` returns an `optional<any>` and `getParameters<type>()` returns an `optional<type>`. In either case, you should verify that the optional value is set before use.

## 4.5.2 Member Function Documentation

### 4.5.2.1 getAllParameters()

```
template<typename T >
const ParameterMap<T>& Ghoti::Wave::HasParameters< T >::getAllParameters ( ) const [inline]
```

Get the current explicitly-set parameters and their values.

#### Returns

The current explicitly-set parameters.

### 4.5.2.2 getParameter()

```
template<typename T >
template<class U >
const std::optional<U> Ghoti::Wave::HasParameters< T >::getParameter (
    const T & parameter ) [inline]
```

Get the parameter as a specified type.

The result is returned as an optional. If there is no parameter value, then the optional value will be false.

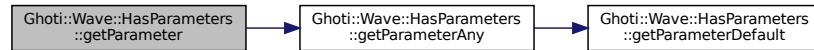
#### Parameters

<i>parameter</i>	The parameter value to get.
------------------	-----------------------------

**Returns**

The (optional) parameter value.

Here is the call graph for this function:

**4.5.2.3 getParameterAny()**

```

template<typename T >
virtual std::optional<std::any> Ghoti::Wave::HasParameters< T >::getParameterAny (
    const T & parameter ) [inline], [virtual]
  
```

Gets the named parameter if it exists, checking locally first, then checking the global defaults.

**Parameters**

<i>parameter</i>	The parameter to get.
------------------	-----------------------

**Returns**

The parameter value if it exists.

Here is the call graph for this function:

**4.5.2.4 getParameterDefault()**

```

template<typename T >
virtual std::optional<std::any> Ghoti::Wave::HasParameters< T >::getParameterDefault (
    [[maybe_unused]] const T & parameter ) [inline], [virtual]
  
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

**Parameters**

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

**Returns**

The associated value.

**4.5.2.5 setParameter()**

```
template<typename T >
virtual HasParameters& Ghoti::Wave::HasParameters< T >::setParameter (
    const T & parameter,
    const std::any & value ) [inline], [virtual]
```

Set a parameter.

**Parameters**

<i>parameter</i>	The parameter key to be set.
<i>value</i>	The parameter value to be set.

**Returns**

The calling object, to allow for chaining.

Reimplemented in [Ghoti::Wave::Server](#), and [Ghoti::Wave::Client](#).

The documentation for this class was generated from the following file:

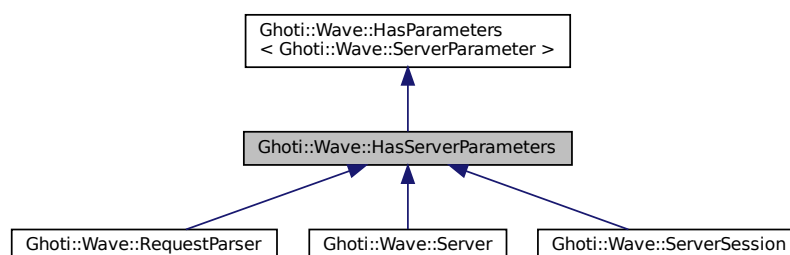
- [include/wave/hasParameters.hpp](#)

**4.6 Ghoti::Wave::HasServerParameters Class Reference**

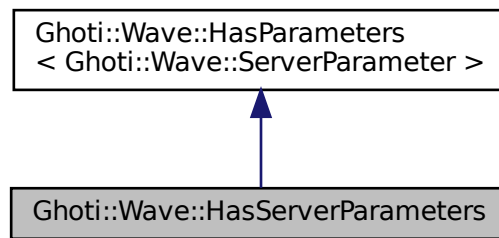
Base class to provide consistent defaults to [Server](#) and [ServerSession](#) classes.

```
#include <hasServerParameters.hpp>
```

Inheritance diagram for Ghoti::Wave::HasServerParameters:



Collaboration diagram for Ghoti::Wave::HasServerParameters:



## Public Member Functions

- virtual std::optional< std::any > [getParameterDefault](#) (const [Ghoti::Wave::ServerParameter](#) &parameter) override  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterDefault](#) ([[maybe\_unused]]const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterAny](#) (const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Gets the named parameter if it exists, checking locally first, then checking the global defaults.*
- const std::optional< U > [getParameter](#) (const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Get the parameter as a specified type.*
- virtual [HasParameters](#) & [setParameter](#) (const [Ghoti::Wave::ServerParameter](#) &parameter, const std::any &value)  
*Set a parameter.*
- const [ParameterMap](#)< [Ghoti::Wave::ServerParameter](#) > & [getAllParameters](#) () const  
*Get the current explicitly-set parameters and their values.*

## Private Attributes

- [ParameterMap](#)< [Ghoti::Wave::ServerParameter](#) > [parameterValues](#)  
*Store explicitly set parameter key/value pairs.*

### 4.6.1 Detailed Description

Base class to provide consistent defaults to [Server](#) and [ServerSession](#) classes.

### 4.6.2 Member Function Documentation

#### 4.6.2.1 getAllParameters()

```
const ParameterMap<Ghoti::Wave::ServerParameter> & Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter>::getAllParameters ( ) const [inline], [inherited]
```

Get the current explicitly-set parameters and their values.

##### Returns

The current explicitly-set parameters.

#### 4.6.2.2 getParameter()

```
const std::optional<U> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter>::get↔  
Parameter (   
    const Ghoti::Wave::ServerParameter & parameter ) [inline], [inherited]
```

Get the parameter as a specified type.

The result is returned as an optional. If there is no parameter value, then the optional value will be false.

##### Parameters

<i>parameter</i>	The parameter value to get.
------------------	-----------------------------

##### Returns

The (optional) parameter value.

#### 4.6.2.3 getParameterAny()

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter>↔  
::getParameterAny (   
    const Ghoti::Wave::ServerParameter & parameter ) [inline], [virtual], [inherited]
```

Gets the named parameter if it exists, checking locally first, then checking the global defaults.

##### Parameters

<i>parameter</i>	The parameter to get.
------------------	-----------------------

##### Returns

The parameter value if it exists.

**4.6.2.4** `getParameterDefault()` [1/2]

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >↔
::getParameterDefault (
    [[maybe_unused]] const Ghoti::Wave::ServerParameter & parameter ) [inline], [virtual],
[inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

**Parameters**

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

**Returns**

The associated value.

**4.6.2.5** `getParameterDefault()` [2/2]

```
optional< any > HasServerParameters::getParameterDefault (
    const Ghoti::Wave::ServerParameter & parameter ) [override], [virtual]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

**Parameters**

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

**Returns**

The associated value.

**4.6.2.6** `setParameter()`

```
virtual HasParameters& Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >::setParameter
(
    const Ghoti::Wave::ServerParameter & parameter,
    const std::any & value ) [inline], [virtual], [inherited]
```

Set a parameter.

**Parameters**

<i>parameter</i>	The parameter key to be set.
<i>value</i>	The parameter value to be set.

**Returns**

The calling object, to allow for chaining.

Reimplemented in [Ghoti::Wave::Server](#).

The documentation for this class was generated from the following files:

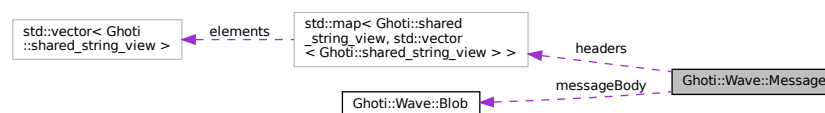
- [include/wave/hasServerParameters.hpp](#)
- [src/hasServerParameters.cpp](#)

## 4.7 Ghoti::Wave::Message Class Reference

Represents a HTTP message.

```
#include <message.hpp>
```

Collaboration diagram for Ghoti::Wave::Message:

**Public Types**

- enum [Type](#) { [REQUEST](#) , [RESPONSE](#) }  
*Indicates whether the message is a request or a response.*
- enum [Transport](#) {  
[UNDECLARED](#) , [FIXED](#) , [MULTIPART](#) , [CHUNKED](#) ,  
[STREAM](#) }  
*Indicate the transport type of the message.*



## Public Member Functions

- [Message](#) ([Type](#) type)  
*The constructor.*
- void [adoptContents](#) ([Message](#) &source)  
*Move the contents of `source` into the `this` object, except for the promise and future attributes.*
- const [Ghoti::shared\\_string\\_view](#) & [getRenderedHeader1](#) ()  
*Get the HTTP/1.1 rendered header as a string.*
- bool [hasError](#) () const  
*Indicates that the message has an error.*
- [Message](#) & [setTransport](#) ([Message::Transport](#) transport)  
*Set the [Message::Transport](#) type of the [Message](#).*
- [Message::Transport](#) [getTransport](#) () const  
*Get the [Message::Transport](#) type of the [Message](#).*
- [Message](#) & [setStatusCode](#) (size\_t statusCode)  
*Set the status code of the message.*
- size\_t [getStatusCode](#) () const  
*Get the status code of the message.*
- [Message](#) & [setErrorMessage](#) (const [Ghoti::shared\\_string\\_view](#) &message)  
*Set an error message description.*
- [Message](#) & [setMessage](#) (const [Ghoti::shared\\_string\\_view](#) &message)  
*Set a status message.*
- const [Ghoti::shared\\_string\\_view](#) & [getMessage](#) () const  
*Get the status message.*
- [Message](#) & [setMethod](#) (const [Ghoti::shared\\_string\\_view](#) &method)  
*Set the HTTP method of the message.*
- const [Ghoti::shared\\_string\\_view](#) & [getMethod](#) () const  
*Get the HTTP method of the message.*
- [Message](#) & [setTarget](#) (const [Ghoti::shared\\_string\\_view](#) &target)  
*Set the URL target of the message.*
- const [Ghoti::shared\\_string\\_view](#) & [getTarget](#) () const  
*Get the URL target of the message.*
- [Message](#) & [setVersion](#) (const [Ghoti::shared\\_string\\_view](#) &version)  
*Set the HTTP version of the message.*
- const [Ghoti::shared\\_string\\_view](#) & [getVersion](#) () const  
*Get the HTTP version of the message.*
- [Message](#) & [addFieldValue](#) (const [Ghoti::shared\\_string\\_view](#) &name, const [Ghoti::shared\\_string\\_view](#) &value)  
*Add a header key/value pair.*
- const std::map< [Ghoti::shared\\_string\\_view](#), std::vector< [Ghoti::shared\\_string\\_view](#) > > & [getFields](#) () const  
*Get the map of all header field key/value pairs.*
- [Type](#) [getType](#) () const  
*Get the [Message::Type](#) of the message.*
- [Message](#) & [setMessageBody](#) ([Ghoti::Wave::Blob](#) &&body)  
*Set the content body of the message.*
- const [Ghoti::Wave::Blob](#) & [getMessageBody](#) () const  
*Get the content body of the message.*
- size\_t [getContentLength](#) () const  
*Get the content length of the message body.*
- [Message](#) & [setPort](#) (size\_t port)  
*Set the port to which the message is targeted.*
- size\_t [getPort](#) () const

- Get the port to which the message is targeted.*

  - `Message & setDomain` (const Ghoti::shared\_string\_view &domain)

*Set the domain to which the message is targeted.*
- const Ghoti::shared\_string\_view & `getDomain` () const

*Get the domain to which the message is targeted.*
- void `setReady` (bool isFinished)

*Notify anyone monitoring the readySemaphore that there is data ready to be processed.*
- bool `isFinished` () const noexcept

*Indicate that the message parsing is completed for this message.*
- std::binary\_semaphore & `getReadySemaphore` ()

*Get the semaphore which will indicate when the message is ready for further processing.*
- `Message & setId` (uint32\_t id)

*Set the ID of the message.*
- uint32\_t `getId` () const

*Get the ID of the message.*

## Private Attributes

- bool `headerIsRendered`

*Used to track whether or not the header has been rendered to a string.*
- bool `errorIsSet`

*Tracks whether or not an error has been set.*
- bool `parsingIsFinished`

*Indicates whether or not the message is "finished" (i.e., there is no more content expected) when the readySemaphore is set.*
- `Type` type

*The `Message::Type` of the message.*
- `Transport` transport

*The `Message::Transport` type of the message.*
- uint32\_t `id`

*The ID number of the message.*
- size\_t `port`

*The port to which the message is targeted.*
- size\_t `statusCode`

*The status code of the message.*
- size\_t `contentLength`

*The contentLength of the message.*
- Ghoti::shared\_string\_view `renderedHeader`

*A cached version of the HTTP/1.1 header.*
- Ghoti::shared\_string\_view `message`

*The status message.*
- Ghoti::shared\_string\_view `method`

*The HTTP method.*
- Ghoti::shared\_string\_view `domain`

*The domain target of the message.*
- Ghoti::shared\_string\_view `target`

*The URL target of the message.*
- Ghoti::shared\_string\_view `version`

*The HTTP version of the message.*
- `Ghoti::Wave::Blob` `messageBody`

*The content body of the message.*

- `std::map< Ghoti::shared_string_view, std::vector< Ghoti::shared_string_view > >` [headers](#)

*A collection of headers and their associated values.*

- `std::binary_semaphore` [readySemaphore](#)

*The semaphore used for asynchronous notification of when the message is ready for processing.*

### 4.7.1 Detailed Description

Represents a HTTP message.

### 4.7.2 Member Enumeration Documentation

#### 4.7.2.1 Transport

enum [Ghoti::Wave::Message::Transport](#)

Indicate the transport type of the message.

Enumerator

UNDECLARED	The transport type has not been declared, and the <a href="#">Message</a> should not be considered to be safe for processing.
FIXED	The <a href="#">Message</a> is a fixed-length and should not be processed until the full length has been received.
MULTIPART	The <a href="#">Message</a> is multipart, each part being separated by a boundary. The message should not be processed until all parts have been received.
CHUNKED	The <a href="#">Message</a> is sent using a chunked encoding. The chunks can be processed as they arrive, asynchronously.
STREAM	The <a href="#">Message</a> did not have a declared (fixed) length. The received bytes may be processed asynchronously.

#### 4.7.2.2 Type

enum [Ghoti::Wave::Message::Type](#)

Indicates whether the message is a request or a response.

Enumerator

REQUEST	A HTTP Request.
RESPONSE	A HTTP Response.

## 4.7.3 Constructor & Destructor Documentation

### 4.7.3.1 Message()

```
Message::Message (
    Type type )
```

The constructor.

Messages must have an associated type.

Parameters

<i>type</i>	The <a href="#">Message::Type</a> of the HTTP message.
-------------	--

## 4.7.4 Member Function Documentation

### 4.7.4.1 addFieldValue()

```
Message & Message::addFieldValue (
    const Ghoti::shared_string_view & name,
    const Ghoti::shared_string_view & value )
```

Add a header key/value pair.

Parameters

<i>name</i>	The field name.
<i>value</i>	The field value.

Returns

The [Message](#) object.

### 4.7.4.2 adoptContents()

```
void Message::adoptContents (
    Message & source )
```

Move the contents of `source` into the `this` object, except for the promise and future attributes.

This method is necessary because the parser may have already started populating a [Message](#) object. A client, however, must supply the [Message](#) object so that the client can know when the promise/future is fulfilled. The only way to accomplish this is to provide a way for the parser to have a provided [Message](#) "adopt" the contents of an existing message, but not bother the associated promise/future of the target.

## Parameters

<i>source</i>	The <a href="#">Message</a> whose contents will be adopted into <code>this</code> .
---------------	---

**4.7.4.3 getContentLength()**

```
size_t Message::getContentLength ( ) const
```

Get the content length of the message body.

## Returns

The content length of the message body.

**4.7.4.4 getDomain()**

```
const shared_string_view & Message::getDomain ( ) const
```

Get the domain to which the message is targeted.

## Returns

The target domain.

**4.7.4.5 getFields()**

```
const map< shared_string_view, vector< shared_string_view > > & Message::getFields ( ) const
```

Get the map of all header field key/value pairs.

`fields[field name] = [field value]`

**4.7.4.6 getId()**

```
uint32_t Message::getId ( ) const
```

Get the ID of the message.

## Returns

The ID number of the message.

#### 4.7.4.7 getMessage()

```
const shared_string_view & Message::getMessage ( ) const
```

Get the status message.

##### Returns

The status message.

#### 4.7.4.8 getMessageBody()

```
const Blob & Message::getMessageBody ( ) const
```

Get the content body of the message.

##### Returns

The content body.

#### 4.7.4.9 getMethod()

```
const shared_string_view & Message::getMethod ( ) const
```

Get the HTTP method of the message.

##### Returns

The HTTP method.

#### 4.7.4.10 getPort()

```
size_t Message::getPort ( ) const
```

Get the port to which the message is targeted.

##### Returns

The target port.

#### 4.7.4.11 getReadySemaphore()

```
binary_semaphore & Message::getReadySemaphore ( )
```

Get the semaphore which will indicate when the message is ready for further processing.

##### Returns

The semaphore used to monitor the status of the message.

#### 4.7.4.12 getRenderedHeader1()

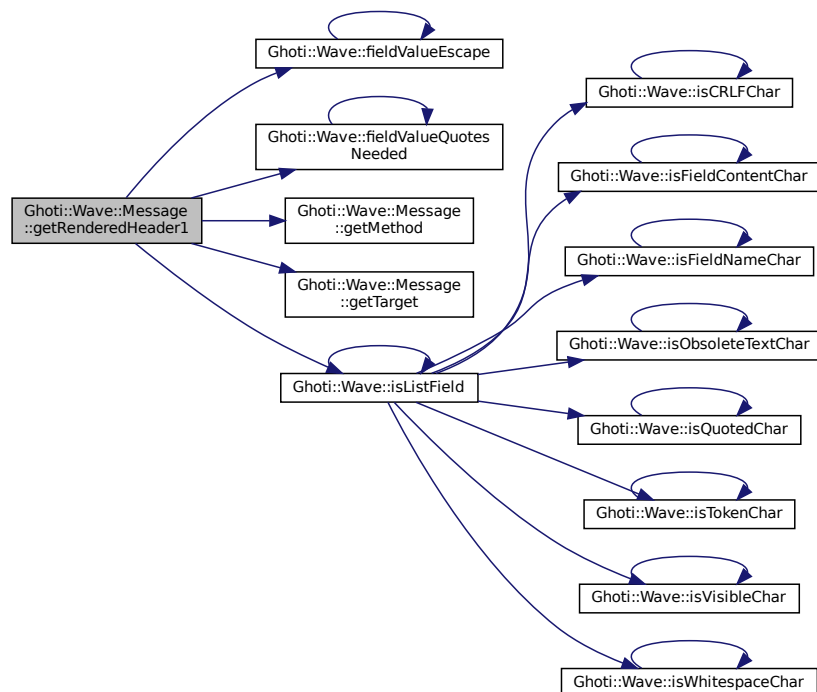
```
const shared_string_view & Message::getRenderedHeader1 ( )
```

Get the HTTP/1.1 rendered header as a string.

##### Returns

A string containing the HTTP/1.1 rendered header.

Here is the call graph for this function:





#### 4.7.4.13 `getStatusCode()`

```
size_t Message::getStatusCode ( ) const
```

Get the status code of the message.

##### Returns

The status code of the message.

#### 4.7.4.14 `getTarget()`

```
const shared_string_view & Message::getTarget ( ) const
```

Get the URL target of the message.

##### Returns

The URL target.

#### 4.7.4.15 `getTransport()`

```
Message::Transport Message::getTransport ( ) const
```

Get the [Message::Transport](#) type of the [Message](#).

##### Returns

The transport type of the [Message](#).

#### 4.7.4.16 `getType()`

```
Message::Type Message::getType ( ) const
```

Get the [Message::Type](#) of the message.

##### Returns

The [Message::Type](#) of the message.

#### 4.7.4.17 getVersion()

```
const shared_string_view & Message::getVersion ( ) const
```

Get the HTTP version of the message.

##### Returns

The HTTP version.

#### 4.7.4.18 hasError()

```
bool Message::hasError ( ) const
```

Indicates that the message has an error.

##### Returns

true if there is an error, false otherwise.

#### 4.7.4.19 isFinished()

```
bool Message::isFinished ( ) const [noexcept]
```

Indicate that the message parsing is completed for this message.

##### Returns

true if the message parsing is complete, false otherwise.

#### 4.7.4.20 setDomain()

```
Message & Message::setDomain (
    const Ghoti::shared_string_view & domain )
```

Set the domain to which the message is targeted.

##### Parameters

<i>domain</i>	The target domain.
---------------	--------------------

**Returns**

The [Message](#) object.

**4.7.4.21 setErrorMessage()**

```
Message & Message::setErrorMessage (
    const Ghoti::shared_string_view & message )
```

Set an error message description.

**Parameters**

<i>message</i>	The error message description.
----------------	--------------------------------

**Returns**

The [Message](#) object.

**4.7.4.22 setId()**

```
Message & Message::setId (
    uint32_t id )
```

Set the ID of the message.

**Parameters**

<i>id</i>	The ID number of the message.
-----------	-------------------------------

**Returns**

The [Message](#) object.

**4.7.4.23 setMessage()**

```
Message & Message::setMessage (
    const Ghoti::shared_string_view & message )
```

Set a status message.

## Parameters

<i>The</i>	status message description.
------------	-----------------------------

## Returns

The [Message](#) object.

**4.7.4.24 setMessageBody()**

```
Message & Message::setMessageBody (
    Ghoti::Wave::Blob && body )
```

Set the content body of the message.

Sets the transport type to Message::Transport::FIXED.

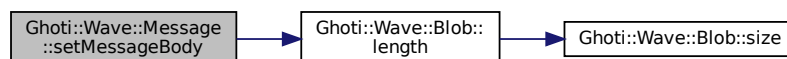
## Parameters

<i>body</i>	The content body.
-------------	-------------------

## Returns

The [Message](#) object.

Here is the call graph for this function:

**4.7.4.25 setMethod()**

```
Message & Message::setMethod (
    const Ghoti::shared_string_view & method )
```

Set the HTTP method of the message.

## Parameters

<i>method</i>	The HTTP method.
---------------	------------------

**Returns**

The [Message](#) object.

**4.7.4.26 setPort()**

```
Message & Message::setPort (
    size_t port )
```

Set the port to which the message is targeted.

**Parameters**

<i>port</i>	The target port.
-------------	------------------

**Returns**

The [Message](#) object.

**4.7.4.27 setReady()**

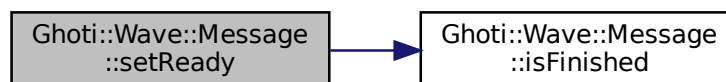
```
void Message::setReady (
    bool isFinished )
```

Notify anyone monitoring the readySemaphore that there is data ready to be processed.

**Parameters**

<i>isFinished</i>	true if the message transmission is completed, otherwise false.
-------------------	---

Here is the call graph for this function:



#### 4.7.4.28 setStatusCode()

```
Message & Message::setStatusCode (
    size_t statusCode )
```

Set the status code of the message.

Per the HTTP spec, this must be a 3-digit number.

##### Parameters

<i>statusCode</i>	The status code of the message.
-------------------	---------------------------------

##### Returns

The [Message](#) object.

#### 4.7.4.29 setTarget()

```
Message & Message::setTarget (
    const Ghoti::shared_string_view & target )
```

Set the URL target of the message.

##### Parameters

<i>target</i>	The URL target.
---------------	-----------------

##### Returns

The [Message](#) object.

#### 4.7.4.30 setTransport()

```
Message & Message::setTransport (
    Message::Transport transport )
```

Set the [Message::Transport](#) type of the [Message](#).

##### Parameters

<i>type</i>	The transport type of the <a href="#">Message</a> .
-------------	---

**Returns**

The [Message](#) object.

**4.7.4.31 setVersion()**

```
Message & Message::setVersion (
    const Ghoti::shared_string_view & version )
```

Set the HTTP version of the message.

**Parameters**

<i>version</i>	The HTTP version.
----------------	-------------------

**Returns**

The [Message](#) object.

**4.7.5 Member Data Documentation****4.7.5.1 headers**

```
std::map<Ghoti::shared_string_view, std::vector<Ghoti::shared_string_view> > Ghoti::Wave::↵
Message::headers [private]
```

A collection of headers and their associated values.

```
headers[field name] = [field value]
```

**4.7.5.2 parsingIsFinished**

```
bool Ghoti::Wave::Message::parsingIsFinished [private]
```

Indicates whether or not the message is "finished" (i.e., there is no more content expected) when the ready↵ Semaphore is set.

Streaming, multipart, and chunked messages will use the readySemaphore to indicate that some part of the message is newly available so that processing can be done in a streaming format.

The documentation for this class was generated from the following files:

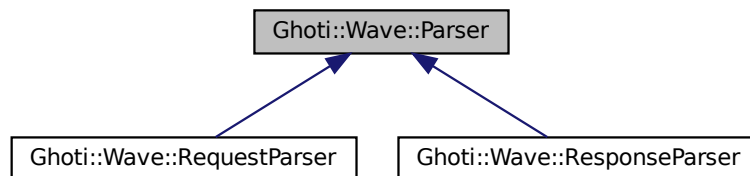
- [include/wave/message.hpp](#)
- [src/message.cpp](#)

## 4.8 Ghoti::Wave::Parser Class Reference

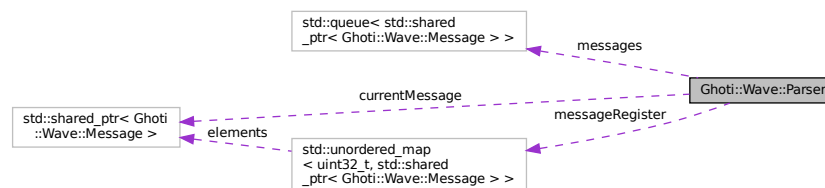
Parses a HTTP/1.1 data stream into discrete messages.

```
#include <parser.hpp>
```

Inheritance diagram for Ghoti::Wave::Parser:



Collaboration diagram for Ghoti::Wave::Parser:



### Public Types

- enum `Type` { `REQUEST` , `RESPONSE` }
- Represents the type of parsing being performed.*

### Public Member Functions

- `Parser` (`Type` type)  
*The constructor.*
- void `processChunk` (const char \*buffer, size\_t len)  
*Process a chunk of data.*
- void `parseMessageTarget` (const Ghoti::shared\_string\_view &target)
- void `registerMessage` (std::shared\_ptr< `Message` > message)  
*Use the provided `Message` as the recipient of parsing for the `Message`'s id.*

### Public Attributes

- std::queue< std::shared\_ptr< `Message` > > `messages`  
*A queue of messages that have been parsed so far.*



## Private Types

- enum [ReadStateMajor](#) { [NEW\\_HEADER](#) , [FIELD\\_LINE](#) , [MESSAGE\\_BODY](#) }  
*Primary state tracking values.*
- enum [ReadStateMinor](#) {  
[BEGINNING\\_OF\\_REQUEST\\_LINE](#) , [BEGINNING\\_OF\\_STATUS\\_LINE](#) , [BEGINNING\\_OF\\_FIELD\\_LINE](#) ,  
[CRLF](#) ,  
[AFTER\\_CRLF](#) , [BEGINNING\\_OF\\_REQUEST](#) , [BEGINNING\\_OF\\_STATUS](#) , [METHOD](#) ,  
[AFTER\\_METHOD](#) , [REQUEST\\_TARGET](#) , [AFTER\\_REQUEST\\_TARGET](#) , [HTTP\\_VERSION](#) ,  
[AFTER\\_HTTP\\_VERSION](#) , [RESPONSE\\_CODE](#) , [REASON\\_PHRASE](#) , [FIELD\\_NAME](#) ,  
[AFTER\\_FIELD\\_NAME](#) , [BEFORE\\_FIELD\\_VALUE](#) , [FIELD\\_VALUE](#) , [SINGLETON\\_FIELD\\_VALUE](#) ,  
[LIST\\_FIELD\\_VALUE](#) , [UNQUOTED\\_FIELD\\_VALUE](#) , [QUOTED\\_FIELD\\_VALUE\\_OPEN](#) , [QUOTED\\_FIELD\\_VALUE\\_PROCESS](#)  
,  
[QUOTED\\_FIELD\\_VALUE\\_ESCAPE](#) , [QUOTED\\_FIELD\\_VALUE\\_CLOSE](#) , [AFTER\\_FIELD\\_VALUE](#) ,  
[FIELD\\_VALUE\\_COMMA](#) ,  
[AFTER\\_FIELD\\_VALUE\\_COMMA](#) , [AFTER\\_HEADER\\_FIELDS](#) , [MESSAGE\\_START](#) , [MESSAGE\\_READ](#) }  
*Secondary state tracking values.*

## Private Member Functions

- virtual uint32\_t [getMEMCHUNKSIZELIMIT](#) ()=0  
*Return the parameter value for MEMCHUNKSIZELIMIT.*
- std::shared\_ptr< [Message](#) > [createNewMessage](#) () const  
*Create a new message whose [Message::Type](#) matches the [Parser::Type](#) of this parser.*

## Private Attributes

- [Type](#) type  
*The [Parser::Type](#) of HTTP/1.1 stream that will be processed.*
- size\_t [cursor](#)  
*An internal counter that indicates the character currently being processed.*
- [ReadStateMajor](#) [readStateMajor](#)  
*Tracks the primary state for the parsing state machine.*
- [ReadStateMinor](#) [readStateMinor](#)  
*Tracks the secondary state for the parsing state machine.*
- size\_t [majorStart](#)  
*Indicates the cursor position at which the major state was last updated.*
- size\_t [minorStart](#)  
*Indicates the cursor position at which the minor state was last updated.*
- Ghoti::shared\_string\_view [input](#)  
*The input string, stored internally so that the stream will be processed correctly, even if it is split across multiple buffered reads.*
- Ghoti::shared\_string\_view [errorMessage](#)  
*An error message to communicate a parsing issue.*
- Ghoti::shared\_string\_view [tempFieldName](#)  
*The field name currently being processed.*
- Ghoti::shared\_string\_view [tempFieldValue](#)  
*The field value currently being processed.*
- std::unordered\_map< uint32\_t, std::shared\_ptr< [Message](#) > > [messageRegister](#)  
*A map to store a [Message](#) associated with a sequence.*
- std::shared\_ptr< [Message](#) > [currentMessage](#)  
*The current message being parsed.*
- size\_t [contentLength](#)  
*The content length that was encountered when parsing the header.*

### 4.8.1 Detailed Description

Parses a HTTP/1.1 data stream into discrete messages.

### 4.8.2 Member Enumeration Documentation

#### 4.8.2.1 ReadStateMajor

```
enum Ghoti::Wave::Parser::ReadStateMajor [private]
```

Primary state tracking values.

These values are used to indicate which major stage the parser is in while parsing the message stream.

The parser uses two stages, to make the parser switch cases easier to follow and to reuse common stages in different contexts (e.g., CRLF).

Enumerator

NEW_HEADER	Expect a new message header.
FIELD_LINE	Expect a new header field.
MESSAGE_BODY	Expect the message body.

#### 4.8.2.2 ReadStateMinor

```
enum Ghoti::Wave::Parser::ReadStateMinor [private]
```

Secondary state tracking values.

These values are used to indicate which "part" of the primary state is being tracked.

Enumerator

BEGINNING_OF_REQUEST_LINE	A request line is starting.
BEGINNING_OF_STATUS_LINE	A status line is starting.
BEGINNING_OF_FIELD_LINE	A header field line is starting.
CRLF	Expect a CRLF.
AFTER_CRLF	A CRLF has been identified.
BEGINNING_OF_REQUEST	Optional whitespace parsed, request line is now starting.
BEGINNING_OF_STATUS	Optional whitespace parsed, status line is now starting.
METHOD	Method expected.
AFTER_METHOD	Method successfully parsed.
REQUEST_TARGET	Expect request target.
AFTER_REQUEST_TARGET	Request target successfully parsed.

## Enumerator

HTTP_VERSION	HTTP version expected.
AFTER_HTTP_VERSION	HTTP version successfully parsed.
RESPONSE_CODE	Response Code Expected.
REASON_PHRASE	Reason Phrase Expected.
FIELD_NAME	Header field name expected.
AFTER_FIELD_NAME	Header field name successfully parsed.
BEFORE_FIELD_VALUE	Header field value about to be processed.
FIELD_VALUE	Header field value expected.
SINGLETON_FIELD_VALUE	Singleton header field value expected.
LIST_FIELD_VALUE	List of header fields expected.
UNQUOTED_FIELD_VALUE	Unquoted field value expected.
QUOTED_FIELD_VALUE_OPEN	Quoted field value begin.
QUOTED_FIELD_VALUE_PROCESS	Quoted field value is being processed.
QUOTED_FIELD_VALUE_ESCAPE	Quoted field value char is being escaped.
QUOTED_FIELD_VALUE_CLOSE	Quoted field value is being closed.
AFTER_FIELD_VALUE	Field value processed.
FIELD_VALUE_COMMA	Field value comma expected.
AFTER_FIELD_VALUE_COMMA	Field value comma processed.
AFTER_HEADER_FIELDS	Header fields processed.
MESSAGE_START	<a href="#">Message</a> started.
MESSAGE_READ	<a href="#">Message</a> being read.

## 4.8.2.3 Type

```
enum Ghoti::Wave::Parser::Type
```

Represents the type of parsing being performed.

## Enumerator

REQUEST	This is a Request stream.
RESPONSE	This is a Response stream.

## 4.8.3 Constructor &amp; Destructor Documentation

## 4.8.3.1 Parser()

```
Parser::Parser (
    Type type )
```

The constructor.

HTTP/1.1 streams do not have an interchangeable syntax, so the stream type must be declared.

The stream will accept an array of bytes, and it will remember its previous parsing position.

#### Parameters

<i>type</i>	The <a href="#">Parser::Type</a> of the message stream.
-------------	---

## 4.8.4 Member Function Documentation

### 4.8.4.1 createNewMessage()

```
shared_ptr< Message > Parser::createNewMessage ( ) const [private]
```

Create a new message whose [Message::Type](#) matches the [Parser::Type](#) of this parser.

This function should really only be used by [Parser::Type::Request](#) parsing, since all [Parser::Type::Response](#) streams should have already registered a [Message](#) object to receive the parsed message.

#### Returns

A properly typed message.

### 4.8.4.2 getMEMCHUNKSIZELIMIT()

```
virtual uint32_t Ghoti::Wave::Parser::getMEMCHUNKSIZELIMIT ( ) [private], [pure virtual]
```

Return the parameter value for MEMCHUNKSIZELIMIT.

#### Returns

The parameter value.

Implemented in [Ghoti::Wave::ResponseParser](#), and [Ghoti::Wave::RequestParser](#).

### 4.8.4.3 processChunk()

```
void Parser::processChunk (
    const char * buffer,
    size_t len )
```

Process a chunk of data.

## Parameters

<i>buffer</i>	The buffer to be processed.
<i>len</i>	The length of the buffer in bytes.

**4.8.4.4 registerMessage()**

```
void Parser::registerMessage (
    std::shared_ptr< Message > message )
```

Use the provided [Message](#) as the recipient of parsing for the [Message](#)'s id.

If a [Message](#) with the target ID already exists, then the provided message will adopt the contents of the existing data.

## Parameters

<i>message</i>	The object that should receive the desired messages.
----------------	--

**4.8.5 Member Data Documentation****4.8.5.1 messageRegister**

```
std::unordered_map<uint32_t, std::shared_ptr<Message> > Ghoti::Wave::Parser::messageRegister
[private]
```

A map to store a [Message](#) associated with a sequence.

This approach is used so that the parser can be informed of the existence of an expected message. This way, the supplied [Message](#) object can act as the recipient of the message as it is parsed.

The registered message should be the same message that was provided to the caller of the [Client::sendRequest\(\)](#) function.

```
messageRegister[ID] = message
```

**4.8.5.2 messages**

```
std::queue<std::shared_ptr<Message> > Ghoti::Wave::Parser::messages
```

A queue of messages that have been parsed so far.

The calling session manager may pop messages from the queue as needed.

The documentation for this class was generated from the following files:

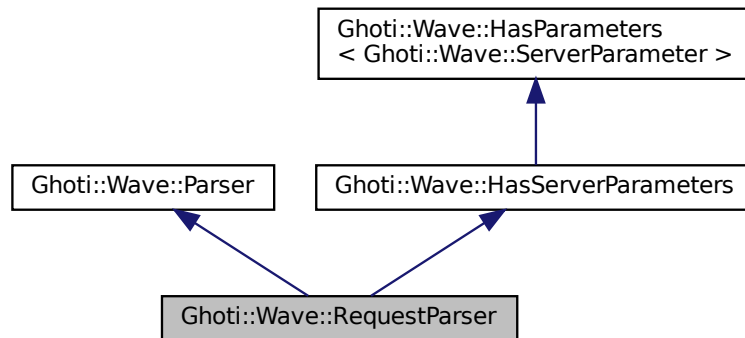
- [include/wave/parser.hpp](#)
- [src/parser.cpp](#)

## 4.9 Ghoti::Wave::RequestParser Class Reference

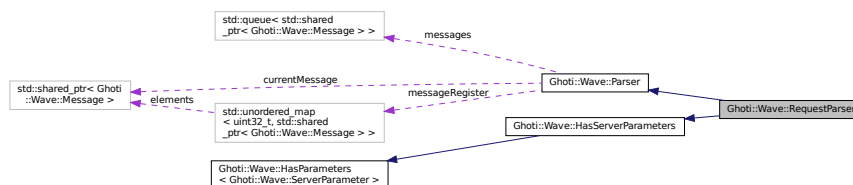
Specialized class for handling the request parser, to make it easier to pass in ServerParameters.

```
#include <parser.hpp>
```

Inheritance diagram for Ghoti::Wave::RequestParser:



Collaboration diagram for Ghoti::Wave::RequestParser:



### Public Types

- enum `Type` { `REQUEST` , `RESPONSE` }  
Represents the type of parsing being performed.

### Public Member Functions

- `RequestParser ()`  
Default constructor.
- void `processChunk` (const char \*buffer, size\_t len)  
Process a chunk of data.
- void `parseMessageTarget` (const Ghoti::shared\_string\_view &target)
- void `registerMessage` (std::shared\_ptr< `Message` > message)  
Use the provided `Message` as the recipient of parsing for the `Message`'s id.

- virtual std::optional< std::any > [getParameterDefault](#) (const [Ghoti::Wave::ServerParameter](#) &parameter) override  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterDefault](#) ([[maybe\_unused]]const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterAny](#) (const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Gets the named parameter if it exists, checking locally first, then checking the global defaults.*
- const std::optional< U > [getParameter](#) (const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Get the parameter as a specified type.*
- virtual [HasParameters](#) & [setParameter](#) (const [Ghoti::Wave::ServerParameter](#) &parameter, const std::any &value)  
*Set a parameter.*
- const [ParameterMap](#)< [Ghoti::Wave::ServerParameter](#) > & [getAllParameters](#) () const  
*Get the current explicitly-set parameters and their values.*

## Public Attributes

- std::queue< std::shared\_ptr< [Message](#) > > [messages](#)  
*A queue of messages that have been parsed so far.*

## Private Types

- enum [ReadStateMajor](#) { [NEW\\_HEADER](#) , [FIELD\\_LINE](#) , [MESSAGE\\_BODY](#) }  
*Primary state tracking values.*
- enum [ReadStateMinor](#) { [BEGINNING\\_OF\\_REQUEST\\_LINE](#) , [BEGINNING\\_OF\\_STATUS\\_LINE](#) , [BEGINNING\\_OF\\_FIELD\\_LINE](#) , [CRLF](#) , [AFTER\\_CRLF](#) , [BEGINNING\\_OF\\_REQUEST](#) , [BEGINNING\\_OF\\_STATUS](#) , [METHOD](#) , [AFTER\\_METHOD](#) , [REQUEST\\_TARGET](#) , [AFTER\\_REQUEST\\_TARGET](#) , [HTTP\\_VERSION](#) , [AFTER\\_HTTP\\_VERSION](#) , [RESPONSE\\_CODE](#) , [REASON\\_PHRASE](#) , [FIELD\\_NAME](#) , [AFTER\\_FIELD\\_NAME](#) , [BEFORE\\_FIELD\\_VALUE](#) , [FIELD\\_VALUE](#) , [SINGLETON\\_FIELD\\_VALUE](#) , [LIST\\_FIELD\\_VALUE](#) , [UNQUOTED\\_FIELD\\_VALUE](#) , [QUOTED\\_FIELD\\_VALUE\\_OPEN](#) , [QUOTED\\_FIELD\\_VALUE\\_PROCESS](#) , [QUOTED\\_FIELD\\_VALUE\\_ESCAPE](#) , [QUOTED\\_FIELD\\_VALUE\\_CLOSE](#) , [AFTER\\_FIELD\\_VALUE](#) , [FIELD\\_VALUE\\_COMMA](#) , [AFTER\\_FIELD\\_VALUE\\_COMMA](#) , [AFTER\\_HEADER\\_FIELDS](#) , [MESSAGE\\_START](#) , [MESSAGE\\_READ](#) }  
*Secondary state tracking values.*

## Private Member Functions

- virtual uint32\_t [getMEMCHUNKSIZELIMIT](#) () override  
*Return the parameter value for MEMCHUNKSIZELIMIT.*
- std::shared\_ptr< [Message](#) > [createNewMessage](#) () const  
*Create a new message whose [Message::Type](#) matches the [Parser::Type](#) of this parser.*

## Private Attributes

- [Type type](#)  
The [Parser::Type](#) of HTTP/1.1 stream that will be processed.
- [size\\_t cursor](#)  
An internal counter that indicates the character currently being processed.
- [ReadStateMajor readStateMajor](#)  
Tracks the primary state for the parsing state machine.
- [ReadStateMinor readStateMinor](#)  
Tracks the secondary state for the parsing state machine.
- [size\\_t majorStart](#)  
Indicates the cursor position at which the major state was last updated.
- [size\\_t minorStart](#)  
Indicates the cursor position at which the minor state was last updated.
- [Ghoti::shared\\_string\\_view input](#)  
The input string, stored internally so that the stream will be processed correctly, even if it is split across multiple buffered reads.
- [Ghoti::shared\\_string\\_view errorMessage](#)  
An error message to communicate a parsing issue.
- [Ghoti::shared\\_string\\_view tempFieldName](#)  
The field name currently being processed.
- [Ghoti::shared\\_string\\_view tempFieldValue](#)  
The field value currently being processed.
- [std::unordered\\_map< uint32\\_t, std::shared\\_ptr< Message > > messageRegister](#)  
A map to store a [Message](#) associated with a sequence.
- [std::shared\\_ptr< Message > currentMessage](#)  
The current message being parsed.
- [size\\_t contentLength](#)  
The content length that was encountered when parsing the header.
- [ParameterMap< Ghoti::Wave::ServerParameter > parameterValues](#)  
Store explicitly set parameter key/value pairs.

### 4.9.1 Detailed Description

Specialized class for handling the request parser, to make it easier to pass in ServerParameters.

### 4.9.2 Member Enumeration Documentation

#### 4.9.2.1 ReadStateMajor

```
enum Ghoti::Wave::Parser::ReadStateMajor [private], [inherited]
```

Primary state tracking values.

These values are used to indicate which major stage the parser is in while parsing the message stream.

The parser uses two stages, to make the parser switch cases easier to follow and to reuse common stages in different contexts (e.g., CRLF).



## Enumerator

NEW_HEADER	Expect a new message header.
FIELD_LINE	Expect a new header field.
MESSAGE_BODY	Expect the message body.

## 4.9.2.2 ReadStateMinor

```
enum Ghoti::Wave::Parser::ReadStateMinor [private], [inherited]
```

Secondary state tracking values.

These values are used to indicate which "part" of the primary state is being tracked.

## Enumerator

BEGINNING_OF_REQUEST_LINE	A request line is starting.
BEGINNING_OF_STATUS_LINE	A status line is starting.
BEGINNING_OF_FIELD_LINE	A header field line is starting.
CRLF	Expect a CRLF.
AFTER_CRLF	A CRLF has been identified.
BEGINNING_OF_REQUEST	Optional whitespace parsed, request line is now starting.
BEGINNING_OF_STATUS	Optional whitespace parsed, status line is now starting.
METHOD	Method expected.
AFTER_METHOD	Method successfully parsed.
REQUEST_TARGET	Expect request target.
AFTER_REQUEST_TARGET	Request target successfully parsed.
HTTP_VERSION	HTTP version expected.
AFTER_HTTP_VERSION	HTTP version successfully parsed.
RESPONSE_CODE	Response Code Expected.
REASON_PHRASE	Reason Phrase Expected.
FIELD_NAME	Header field name expected.
AFTER_FIELD_NAME	Header field name successfully parsed.
BEFORE_FIELD_VALUE	Header field value about to be processed.
FIELD_VALUE	Header field value expected.
SINGLETON_FIELD_VALUE	Singleton header field value expected.
LIST_FIELD_VALUE	List of header fields expected.
UNQUOTED_FIELD_VALUE	Unquoted field value expected.
QUOTED_FIELD_VALUE_OPEN	Quoted field value begin.
QUOTED_FIELD_VALUE_PROCESS	Quoted field value is being processed.
QUOTED_FIELD_VALUE_ESCAPE	Quoted field value char is being escaped.
QUOTED_FIELD_VALUE_CLOSE	Quoted field value is being closed.
AFTER_FIELD_VALUE	Field value processed.
FIELD_VALUE_COMMA	Field value comma expected.
AFTER_FIELD_VALUE_COMMA	Field value comma processed.
AFTER_HEADER_FIELDS	Header fields processed.
MESSAGE_START	<a href="#">Message</a> started.
MESSAGE_READ	<a href="#">Message</a> being read.

### 4.9.2.3 Type

```
enum Ghoti::Wave::Parser::Type [inherited]
```

Represents the type of parsing being performed.

Enumerator

REQUEST	This is a Request stream.
RESPONSE	This is a Response stream.

## 4.9.3 Member Function Documentation

### 4.9.3.1 createNewMessage()

```
shared_ptr< Message > Parser::createNewMessage ( ) const [private], [inherited]
```

Create a new message whose [Message::Type](#) matches the [Parser::Type](#) of this parser.

This function should really only be used by `Parser::Type::Request` parsing, since all `Parser::Type::Response` streams should have already registered a [Message](#) object to receive the parsed message.

Returns

A properly typed message.

### 4.9.3.2 getAllParameters()

```
const ParameterMap<Ghoti::Wave::ServerParameter >& Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >::getAllParameters ( ) const [inline], [inherited]
```

Get the current explicitly-set parameters and their values.

Returns

The current explicitly-set parameters.

#### 4.9.3.3 getMEMCHUNKSIZELIMIT()

```
uint32_t RequestParser::getMEMCHUNKSIZELIMIT ( ) [override], [private], [virtual]
```

Return the parameter value for MEMCHUNKSIZELIMIT.

##### Returns

The parameter value.

Implements [Ghoti::Wave::Parser](#).

#### 4.9.3.4 getParameter()

```
const std::optional<U> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >::get↔
Parameter (
    const T & parameter ) [inline], [inherited]
```

Get the parameter as a specified type.

The result is returned as an optional. If there is no parameter value, then the optional value will be false.

##### Parameters

<i>parameter</i>	The parameter value to get.
------------------	-----------------------------

##### Returns

The (optional) parameter value.

#### 4.9.3.5 getParameterAny()

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >↔
::getParameterAny (
    const T & parameter ) [inline], [virtual], [inherited]
```

Gets the named parameter if it exists, checking locally first, then checking the global defaults.

##### Parameters

<i>parameter</i>	The parameter to get.
------------------	-----------------------

##### Returns

The parameter value if it exists.

#### 4.9.3.6 `getParameterDefault()` [1/2]

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >↔
::getParameterDefault (
    [[maybe_unused] ]const T & parameter ) [inline], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

##### Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

##### Returns

The associated value.

#### 4.9.3.7 `getParameterDefault()` [2/2]

```
optional< any > HasServerParameters::getParameterDefault (
    const Ghoti::Wave::ServerParameter & parameter ) [override], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

##### Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

##### Returns

The associated value.

#### 4.9.3.8 `processChunk()`

```
void Parser::processChunk (
    const char * buffer,
    size_t len ) [inherited]
```

Process a chunk of data.

## Parameters

<i>buffer</i>	The buffer to be processed.
<i>len</i>	The length of the buffer in bytes.

**4.9.3.9 registerMessage()**

```
void Parser::registerMessage (
    std::shared_ptr< Message > message ) [inherited]
```

Use the provided [Message](#) as the recipient of parsing for the [Message](#)'s id.

If a [Message](#) with the target ID already exists, then the provided message will adopt the contents of the existing data.

## Parameters

<i>message</i>	The object that should receive the desired messages.
----------------	--

**4.9.3.10 setParameter()**

```
virtual HasParameters& Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >::setParameter
(
    const T & parameter,
    const std::any & value ) [inline], [virtual], [inherited]
```

Set a parameter.

## Parameters

<i>parameter</i>	The parameter key to be set.
<i>value</i>	The parameter value to be set.

## Returns

The calling object, to allow for chaining.

Reimplemented in [Ghoti::Wave::Server](#).

**4.9.4 Member Data Documentation**

#### 4.9.4.1 messageRegister

```
std::unordered_map<uint32_t, std::shared_ptr<Message> > Ghoti::Wave::Parser::messageRegister
[private], [inherited]
```

A map to store a [Message](#) associated with a sequence.

This approach is used so that the parser can be informed of the existence of an expected message. This way, the supplied [Message](#) object can act as the recipient of the message as it is parsed.

The registered message should be the same message that was provided to the caller of the [Client::sendRequest\(\)](#) function.

```
messageRegister[ID] = message
```

#### 4.9.4.2 messages

```
std::queue<std::shared_ptr<Message> > Ghoti::Wave::Parser::messages [inherited]
```

A queue of messages that have been parsed so far.

The calling session manager may pop messages from the queue as needed.

The documentation for this class was generated from the following files:

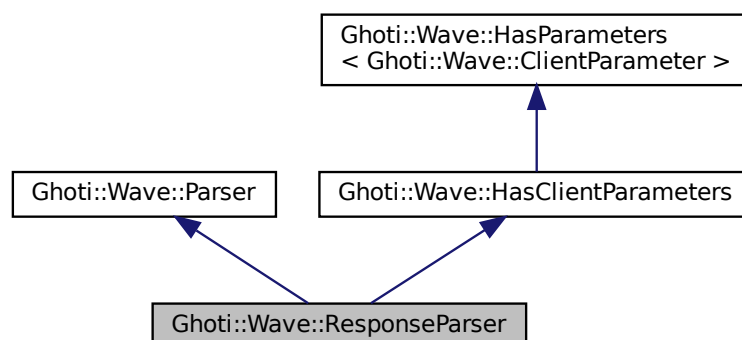
- [include/wave/parser.hpp](#)
- [src/parser.cpp](#)

## 4.10 Ghoti::Wave::ResponseParser Class Reference

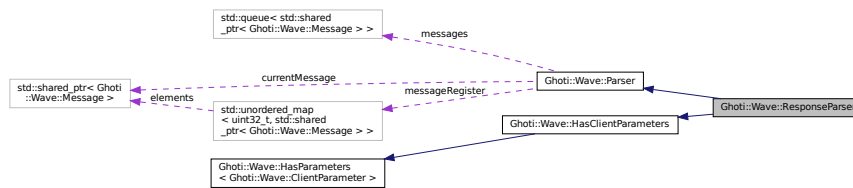
Specialized class for handling the response parser, to make it easier to pass in ClientParameters.

```
#include <parser.hpp>
```

Inheritance diagram for Ghoti::Wave::ResponseParser:



Collaboration diagram for Ghoti::Wave::ResponseParser:



## Public Types

- enum [Type](#) { [REQUEST](#) , [RESPONSE](#) }  
*Represents the type of parsing being performed.*

## Public Member Functions

- [ResponseParser](#) ()  
*Default constructor.*
- void [processChunk](#) (const char \*buffer, size\_t len)  
*Process a chunk of data.*
- void [parseMessageTarget](#) (const Ghoti::shared\_string\_view &target)
- void [registerMessage](#) (std::shared\_ptr< [Message](#) > message)  
*Use the provided [Message](#) as the recipient of parsing for the [Message](#)'s id.*
- virtual std::optional< std::any > [getParameterDefault](#) (const [Ghoti::Wave::ClientParameter](#) &parameter) override  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterDefault](#) ([[maybe\_unused]]const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterAny](#) (const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Gets the named parameter if it exists, checking locally first, then checking the global defaults.*
- const std::optional< U > [getParameter](#) (const [Ghoti::Wave::ClientParameter](#) &parameter)  
*Get the parameter as a specified type.*
- virtual [HasParameters](#) & [setParameter](#) (const [Ghoti::Wave::ClientParameter](#) &parameter, const std::any &value)  
*Set a parameter.*
- const [ParameterMap](#)< [Ghoti::Wave::ClientParameter](#) > & [getAllParameters](#) () const  
*Get the current explicitly-set parameters and their values.*

## Public Attributes

- std::queue< std::shared\_ptr< [Message](#) > > [messages](#)  
*A queue of messages that have been parsed so far.*

## Private Types

- enum [ReadStateMajor](#) { [NEW\\_HEADER](#) , [FIELD\\_LINE](#) , [MESSAGE\\_BODY](#) }  
*Primary state tracking values.*
- enum [ReadStateMinor](#) {  
[BEGINNING\\_OF\\_REQUEST\\_LINE](#) , [BEGINNING\\_OF\\_STATUS\\_LINE](#) , [BEGINNING\\_OF\\_FIELD\\_LINE](#) ,  
[CRLF](#) ,  
[AFTER\\_CRLF](#) , [BEGINNING\\_OF\\_REQUEST](#) , [BEGINNING\\_OF\\_STATUS](#) , [METHOD](#) ,  
[AFTER\\_METHOD](#) , [REQUEST\\_TARGET](#) , [AFTER\\_REQUEST\\_TARGET](#) , [HTTP\\_VERSION](#) ,  
[AFTER\\_HTTP\\_VERSION](#) , [RESPONSE\\_CODE](#) , [REASON\\_PHRASE](#) , [FIELD\\_NAME](#) ,  
[AFTER\\_FIELD\\_NAME](#) , [BEFORE\\_FIELD\\_VALUE](#) , [FIELD\\_VALUE](#) , [SINGLETON\\_FIELD\\_VALUE](#) ,  
[LIST\\_FIELD\\_VALUE](#) , [UNQUOTED\\_FIELD\\_VALUE](#) , [QUOTED\\_FIELD\\_VALUE\\_OPEN](#) , [QUOTED\\_FIELD\\_VALUE\\_PROCESS](#)  
,  
[QUOTED\\_FIELD\\_VALUE\\_ESCAPE](#) , [QUOTED\\_FIELD\\_VALUE\\_CLOSE](#) , [AFTER\\_FIELD\\_VALUE](#) ,  
[FIELD\\_VALUE\\_COMMA](#) ,  
[AFTER\\_FIELD\\_VALUE\\_COMMA](#) , [AFTER\\_HEADER\\_FIELDS](#) , [MESSAGE\\_START](#) , [MESSAGE\\_READ](#) }  
*Secondary state tracking values.*

## Private Member Functions

- virtual uint32\_t [getMEMCHUNKSIZELIMIT](#) () override  
*Return the parameter value for MEMCHUNKSIZELIMIT.*
- std::shared\_ptr< [Message](#) > [createNewMessage](#) () const  
*Create a new message whose [Message::Type](#) matches the [Parser::Type](#) of this parser.*

## Private Attributes

- [Type](#) type  
*The [Parser::Type](#) of HTTP/1.1 stream that will be processed.*
- size\_t [cursor](#)  
*An internal counter that indicates the character currently being processed.*
- [ReadStateMajor](#) [readStateMajor](#)  
*Tracks the primary state for the parsing state machine.*
- [ReadStateMinor](#) [readStateMinor](#)  
*Tracks the secondary state for the parsing state machine.*
- size\_t [majorStart](#)  
*Indicates the cursor position at which the major state was last updated.*
- size\_t [minorStart](#)  
*Indicates the cursor position at which the minor state was last updated.*
- Ghоти::shared\_string\_view [input](#)  
*The input string, stored internally so that the stream will be processed correctly, even if it is split across multiple buffered reads.*
- Ghоти::shared\_string\_view [errorMessage](#)  
*An error message to communicate a parsing issue.*
- Ghоти::shared\_string\_view [tempFieldName](#)  
*The field name currently being processed.*
- Ghоти::shared\_string\_view [tempFieldValue](#)  
*The field value currently being processed.*
- std::unordered\_map< uint32\_t, std::shared\_ptr< [Message](#) > > [messageRegister](#)  
*A map to store a [Message](#) associated with a sequence.*
- std::shared\_ptr< [Message](#) > [currentMessage](#)  
*The current message being parsed.*
- size\_t [contentLength](#)  
*The content length that was encountered when parsing the header.*
- [ParameterMap](#)< Ghоти::Wave::ClientParameter > [parameterValues](#)  
*Store explicitly set parameter key/value pairs.*



### 4.10.1 Detailed Description

Specialized class for handling the response parser, to make it easier to pass in ClientParameters.

### 4.10.2 Member Enumeration Documentation

#### 4.10.2.1 ReadStateMajor

```
enum Ghoti::Wave::Parser::ReadStateMajor [private], [inherited]
```

Primary state tracking values.

These values are used to indicate which major stage the parser is in while parsing the message stream.

The parser uses two stages, to make the parser switch cases easier to follow and to reuse common stages in different contexts (e.g., CRLF).

Enumerator

NEW_HEADER	Expect a new message header.
FIELD_LINE	Expect a new header field.
MESSAGE_BODY	Expect the message body.

#### 4.10.2.2 ReadStateMinor

```
enum Ghoti::Wave::Parser::ReadStateMinor [private], [inherited]
```

Secondary state tracking values.

These values are used to indicate which "part" of the primary state is being tracked.

Enumerator

BEGINNING_OF_REQUEST_LINE	A request line is starting.
BEGINNING_OF_STATUS_LINE	A status line is starting.
BEGINNING_OF_FIELD_LINE	A header field line is starting.
CRLF	Expect a CRLF.
AFTER_CRLF	A CRLF has been identified.
BEGINNING_OF_REQUEST	Optional whitespace parsed, request line is now starting.
BEGINNING_OF_STATUS	Optional whitespace parsed, status line is now starting.
METHOD	Method expected.
AFTER_METHOD	Method successfully parsed.
REQUEST_TARGET	Expect request target.
AFTER_REQUEST_TARGET	Request target successfully parsed.

## Enumerator

HTTP_VERSION	HTTP version expected.
AFTER_HTTP_VERSION	HTTP version successfully parsed.
RESPONSE_CODE	Response Code Expected.
REASON_PHRASE	Reason Phrase Expected.
FIELD_NAME	Header field name expected.
AFTER_FIELD_NAME	Header field name successfully parsed.
BEFORE_FIELD_VALUE	Header field value about to be processed.
FIELD_VALUE	Header field value expected.
SINGLETON_FIELD_VALUE	Singleton header field value expected.
LIST_FIELD_VALUE	List of header fields expected.
UNQUOTED_FIELD_VALUE	Unquoted field value expected.
QUOTED_FIELD_VALUE_OPEN	Quoted field value begin.
QUOTED_FIELD_VALUE_PROCESS	Quoted field value is being processed.
QUOTED_FIELD_VALUE_ESCAPE	Quoted field value char is being escaped.
QUOTED_FIELD_VALUE_CLOSE	Quoted field value is being closed.
AFTER_FIELD_VALUE	Field value processed.
FIELD_VALUE_COMMA	Field value comma expected.
AFTER_FIELD_VALUE_COMMA	Field value comma processed.
AFTER_HEADER_FIELDS	Header fields processed.
MESSAGE_START	<a href="#">Message</a> started.
MESSAGE_READ	<a href="#">Message</a> being read.

## 4.10.2.3 Type

```
enum Ghoti::Wave::Parser::Type [inherited]
```

Represents the type of parsing being performed.

## Enumerator

REQUEST	This is a Request stream.
RESPONSE	This is a Response stream.

## 4.10.3 Member Function Documentation

## 4.10.3.1 createNewMessage()

```
shared_ptr< Message > Parser::createNewMessage ( ) const [private], [inherited]
```

Create a new message whose [Message::Type](#) matches the [Parser::Type](#) of this parser.

This function should really only be used by [Parser::Type::Request](#) parsing, since all [Parser::Type::Response](#) streams should have already registered a [Message](#) object to receive the parsed message.

**Returns**

A properly typed message.

**4.10.3.2 getAllParameters()**

```
const ParameterMap<Ghoti::Wave::ClientParameter> & Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter>::getAllParameters ( ) const [inline], [inherited]
```

Get the current explicitly-set parameters and their values.

**Returns**

The current explicitly-set parameters.

**4.10.3.3 getMEMCHUNKSIZELIMIT()**

```
uint32_t ResponseParser::getMEMCHUNKSIZELIMIT ( ) [override], [private], [virtual]
```

Return the parameter value for MEMCHUNKSIZELIMIT.

**Returns**

The parameter value.

Implements [Ghoti::Wave::Parser](#).

**4.10.3.4 getParameter()**

```
const std::optional<U> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter>::get↵
Parameter (
    const T & parameter ) [inline], [inherited]
```

Get the parameter as a specified type.

The result is returned as an optional. If there is no parameter value, then the optional value will be false.

**Parameters**

<i>parameter</i>	The parameter value to get.
------------------	-----------------------------

**Returns**

The (optional) parameter value.

**4.10.3.5 getParameterAny()**

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >↔
::getParameterAny (
    const T & parameter ) [inline], [virtual], [inherited]
```

Gets the named parameter if it exists, checking locally first, then checking the global defaults.

**Parameters**

<i>parameter</i>	The parameter to get.
------------------	-----------------------

**Returns**

The parameter value if it exists.

**4.10.3.6 getParameterDefault() [1/2]**

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >↔
::getParameterDefault (
    [[maybe_unused]] const T & parameter ) [inline], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

**Parameters**

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

**Returns**

The associated value.

**4.10.3.7 getParameterDefault() [2/2]**

```
optional< any > HasClientParameters::getParameterDefault (
    const Ghoti::Wave::ClientParameter & parameter ) [override], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

#### Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

#### Returns

The associated value.

#### 4.10.3.8 processChunk()

```
void Parser::processChunk (
    const char * buffer,
    size_t len ) [inherited]
```

Process a chunk of data.

#### Parameters

<i>buffer</i>	The buffer to be processed.
<i>len</i>	The length of the buffer in bytes.

#### 4.10.3.9 registerMessage()

```
void Parser::registerMessage (
    std::shared_ptr< Message > message ) [inherited]
```

Use the provided [Message](#) as the recipient of parsing for the [Message](#)'s id.

If a [Message](#) with the target ID already exists, then the provided message will adopt the contents of the existing data.

#### Parameters

<i>message</i>	The object that should receive the desired messages.
----------------	--

#### 4.10.3.10 setParameter()

```
virtual HasParameters& Ghoti::Wave::HasParameters< Ghoti::Wave::ClientParameter >::setParameter
```

```
(
    const T & parameter,
    const std::any & value ) [inline], [virtual], [inherited]
```

Set a parameter.

#### Parameters

<i>parameter</i>	The parameter key to be set.
<i>value</i>	The parameter value to be set.

#### Returns

The calling object, to allow for chaining.

Reimplemented in [Ghoti::Wave::Client](#).

## 4.10.4 Member Data Documentation

### 4.10.4.1 messageRegister

```
std::unordered_map<uint32_t, std::shared_ptr<Message> > Ghoti::Wave::Parser::messageRegister
[private], [inherited]
```

A map to store a [Message](#) associated with a sequence.

This approach is used so that the parser can be informed of the existence of an expected message. This way, the supplied [Message](#) object can act as the recipient of the message as it is parsed.

The registered message should be the same message that was provided to the caller of the [Client::sendRequest\(\)](#) function.

```
messageRegister[ID] = message
```

### 4.10.4.2 messages

```
std::queue<std::shared_ptr<Message> > Ghoti::Wave::Parser::messages [inherited]
```

A queue of messages that have been parsed so far.

The calling session manager may pop messages from the queue as needed.

The documentation for this class was generated from the following files:

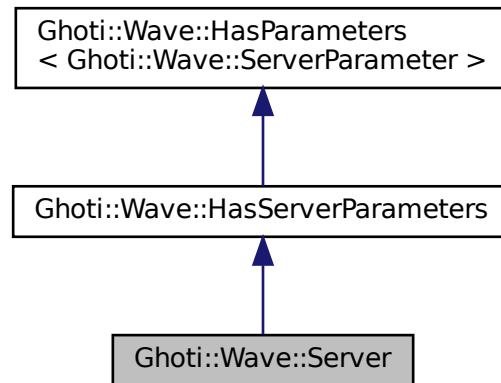
- [include/wave/parser.hpp](#)
- [src/parser.cpp](#)

## 4.11 Ghoti::Wave::Server Class Reference

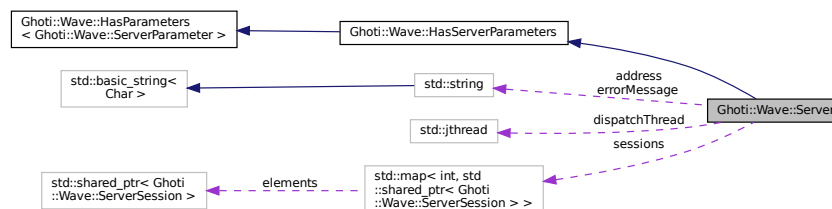
The base [Server](#) class.

```
#include <server.hpp>
```

Inheritance diagram for Ghoti::Wave::Server:



Collaboration diagram for Ghoti::Wave::Server:



### Public Types

- enum [ErrorCode](#) { [NO\\_ERROR](#) , [SERVER\\_ALREADY\\_RUNNING](#) , [START\\_FAILED](#) }

*These are the error codes that the [Server](#) may generate when control functions fail.*

### Public Member Functions

- [Server](#) ()  
*The constructor.*
- [~Server](#) ()  
*The destructor.*

- [Server](#) & [clearError](#) ()  
*Clears any error code and error message that may be set.*
- [ErrorCode](#) [getErrorCode](#) () const  
*Returns the [Server::ErrorCode](#) error that was most recently generated.*
- const std::string & [getErrorMessage](#) () const  
*Returns an error message string that was most recently generated.*
- bool [isRunning](#) () const  
*Returns whether or not the server is running.*
- [Server](#) & [setPort](#) (uint16\_t port)  
*Set the port that the server is listening on.*
- uint16\_t [getPort](#) () const  
*Return the server's current port setting.*
- [Server](#) & [setAddress](#) (const char \*ip)  
*Set the ip address that the server is listening on.*
- const std::string & [getAddress](#) () const  
*Return the server's current ip address setting.*
- int [getSocketHandle](#) () const  
*Returns the socket handle of the server (if set).*
- [Server](#) & [start](#) ()  
*Start the server listening on the designated ip address and port.*
- [Server](#) & [stop](#) ()  
*Signal the server to stop listening and terminate its thread pool.*
- void [dispatchLoop](#) (std::stop\_token token)  
*The Dispatch loop used by the thread pool to handle asynchronous reading and writing of the server ports.*
- virtual [Server](#) & [setParameter](#) (const [ServerParameter](#) &parameter, const std::any &value) override  
*Set a parameter.*
- virtual std::optional< std::any > [getParameterDefault](#) (const [Ghoti::Wave::ServerParameter](#) &parameter) override  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterDefault](#) ([[maybe\_unused]]const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterAny](#) (const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Gets the named parameter if it exists, checking locally first, then checking the global defaults.*
- const std::optional< U > [getParameter](#) (const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Get the parameter as a specified type.*
- const [ParameterMap](#)< [Ghoti::Wave::ServerParameter](#) > & [getAllParameters](#) () const  
*Get the current explicitly-set parameters and their values.*

## Private Attributes

- [Ghoti::Pool::Pool](#) [workers](#)  
*The thread pool worker queue.*
- std::map< int, std::shared\_ptr< [Ghoti::Wave::ServerSession](#) > > [sessions](#)  
*Stores active sessions.*
- std::jthread [dispatchThread](#)  
*The dispatch thread used to monitor for new connections and to dispatch read/write tasks as needed by the sessions.*
- [ErrorCode](#) [errorCode](#)  
*The most recently generated error code.*
- std::string [errorMessage](#)



- *The most recently generated error message.*
- bool [running](#)  
*Stores whether or not the server is set to be running.*
- int [hSocket](#)  
*The socket handle to which the running server is attached.*
- std::string [address](#)  
*The ip address that the server is configured to use.*
- uint16\_t [port](#)  
*The port that the server is configured to use.*
- [ParameterMap](#)< [Ghoti::Wave::ServerParameter](#) > [parameterValues](#)  
*Store explicitly set parameter key/value pairs.*

### 4.11.1 Detailed Description

The base [Server](#) class.

This class is the foundation of the Ghoti.io HTTP server. It serves as the interface to control and expand the server programmatically.

### 4.11.2 Member Enumeration Documentation

#### 4.11.2.1 ErrorCode

```
enum Ghoti::Wave::Server::ErrorCode
```

These are the error codes that the [Server](#) may generate when control functions fail.

Enumerator

<a href="#">NO_ERROR</a>	No error.
<a href="#">SERVER_ALREADY_RUNNING</a>	The change could not be applied because the server is already running.
<a href="#">START_FAILED</a>	The server could not be started.

### 4.11.3 Constructor & Destructor Documentation

#### 4.11.3.1 Server()

```
Server::Server ( )
```

The constructor.

The constructor only creates the server object. It does not begin listening for connections. In order to begin listening for connections, the [Server.start\(\)](#) function must be called.

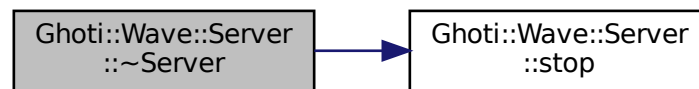
By default, the server will bind to "127.0.0.1" and a port number assigned by the operating system. This default functionality can be changed by using [Server.setAddress\(\)](#) and [Server.setPort\(\)](#), respectively.

#### 4.11.3.2 ~Server()

```
Server::~~Server ( )
```

The destructor.

The destructor will call [Server.stop\(\)](#). Here is the call graph for this function:



### 4.11.4 Member Function Documentation

#### 4.11.4.1 clearError()

```
Server & Server::clearError ( )
```

Clears any error code and error message that may be set.

Error messages are not cleared automatically. This function must be called explicitly.

##### Returns

The server object.

#### 4.11.4.2 dispatchLoop()

```
void Server::dispatchLoop (
    std::stop_token token )
```

The Dispatch loop used by the thread pool to handle asynchronous reading and writing of the server ports.

## Parameters

<code>stop_token</code>	The stop token provided by the jthread to indicate that the thread should be safely shut down.
-------------------------	--

#### 4.11.4.3 getAddress()

```
const string & Server::getAddress ( ) const
```

Return the server's current ip address setting.

This setting does not imply that the server is active.

## Returns

The current ip address.

#### 4.11.4.4 getAllParameters()

```
const ParameterMap<Ghoti::Wave::ServerParameter >& Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >::getAllParameters ( ) const [inline], [inherited]
```

Get the current explicitly-set parameters and their values.

## Returns

The current explicitly-set parameters.

#### 4.11.4.5 getErrorCode()

```
Server::ErrorCode Server::getErrorCode ( ) const
```

Returns the [Server::ErrorCode](#) error that was most recently generated.

Calling the function does not clear the error. The error must be cleared explicitly by calling [Server::clearError\(\)](#).

## Returns

The [Server::ErrorCode](#) error that was most recently generated.

#### 4.11.4.6 `getErrorMessage()`

```
const std::string & Server::getErrorMessage ( ) const
```

Returns an error message string that was most recently generated.

Calling the function does not clear the error. The error must be cleared explicitly by calling [Server::clearError\(\)](#).

##### Returns

The error message string that was most recently generated.

#### 4.11.4.7 `getParameter()`

```
const std::optional<U> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >::get↔
Parameter (
    const T & parameter ) [inline], [inherited]
```

Get the parameter as a specified type.

The result is returned as an optional. If there is no parameter value, then the optional value will be false.

##### Parameters

<i>parameter</i>	The parameter value to get.
------------------	-----------------------------

##### Returns

The (optional) parameter value.

#### 4.11.4.8 `getParameterAny()`

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >↔
::getParameterAny (
    const T & parameter ) [inline], [virtual], [inherited]
```

Gets the named parameter if it exists, checking locally first, then checking the global defaults.

##### Parameters

<i>parameter</i>	The parameter to get.
------------------	-----------------------

##### Returns

The parameter value if it exists.

**4.11.4.9** `getParameterDefault()` [1/2]

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >↵
::getParameterDefault (
    [[maybe_unused]] const T & parameter ) [inline], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

**Parameters**

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

**Returns**

The associated value.

**4.11.4.10** `getParameterDefault()` [2/2]

```
optional< any > HasServerParameters::getParameterDefault (
    const Ghoti::Wave::ServerParameter & parameter ) [override], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

**Parameters**

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

**Returns**

The associated value.

**4.11.4.11** `getPort()`

```
uint16_t Server::getPort ( ) const
```

Return the server's current port setting.

This setting does not imply that the server is active.

**Returns**

The current port number.

**4.11.4.12 getSocketHandle()**

```
int Server::getSocketHandle ( ) const
```

Returns the socket handle of the server (if set).

**Returns**

The socket handle of the server.

**4.11.4.13 isRunning()**

```
bool Server::isRunning ( ) const
```

Returns whether or not the server is running.

**Returns**

True/False whether or not the server is running.

**4.11.4.14 setAddress()**

```
Server & Server::setAddress (
    const char * ip )
```

Set the ip address that the server is listening on.

This setting cannot be changed if the server is running. If the server is running, then an error will be set.

**Parameters**

<i>ip</i>	The ip address that the server should listen on.
-----------	--

**Returns**

The server object.

#### 4.11.4.15 setParameter()

```
Server & Server::setParameter (
    const ServerParameter & parameter,
    const std::any & value ) [override], [virtual]
```

Set a parameter.

Values will be propagated to all [Server](#) sessions.

##### Parameters

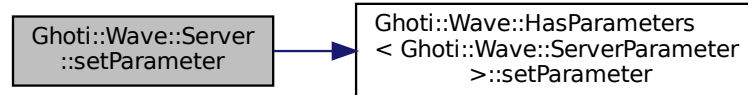
<i>parameter</i>	The parameter key to be set.
<i>value</i>	The parameter value to be set.

##### Returns

The calling object, to allow for chaining.

Reimplemented from [Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >](#).

Here is the call graph for this function:



#### 4.11.4.16 setPort()

```
Server & Server::setPort (
    uint16_t port )
```

Set the port that the server is listening on.

This setting cannot be changed if the server is running. If the server is running, then an error will be set.

##### Parameters

<i>port</i>	The port number that the server should listen on.
-------------	---

**Returns**

The server object.

**4.11.4.17 start()**

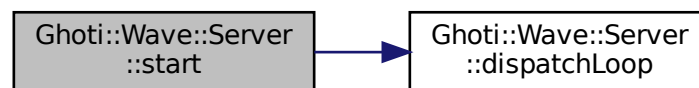
```
Server & Server::start ( )
```

Start the server listening on the designated ip address and port.

**Returns**

The server object.

Here is the call graph for this function:

**4.11.4.18 stop()**

```
Server & Server::stop ( )
```

Signal the server to stop listening and terminate its thread pool.

**Returns**

The server object.

**4.11.5 Member Data Documentation**



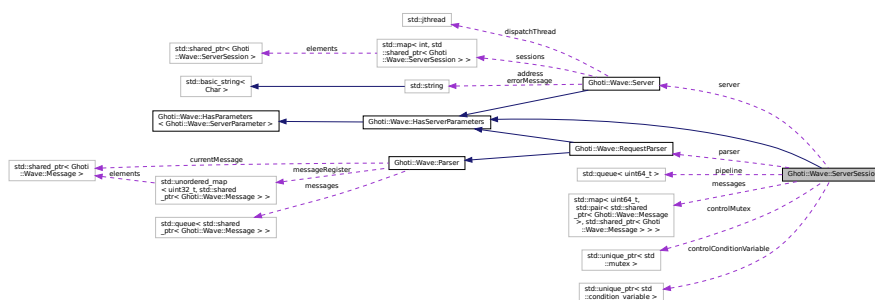
```
std::map<int, std::shared_ptr<Ghoti::Wave::ServerSession> > Ghoti::Wave::Server::sessions
[private]
```

The sessions are keyed by the socket handle to which the session is associated.

- include/wave/server.hpp
- src/server.cpp

Represents a persistent connection with a client.

Inheritance diagram for Ghoti::Wave::ServerSession:



## Public Member Functions

- [ServerSession](#) (int [hClient](#), [Server](#) \*[server](#))  
*The constructor.*
- [~ServerSession](#) ()  
*The destructor.*
- bool [hasReadDataWaiting](#) ()  
*Checks to see whether or not the session has data waiting to be read from the socket.*
- bool [hasWriteDataWaiting](#) ()  
*Checks to see whether or not the session has data waiting to be written to the socket.*
- bool [isFinished](#) ()  
*Indicates whether or not the session has completed all communications and may be terminated.*
- void [read](#) ()  
*Perform a read from the session.*
- void [write](#) ()  
*Perform a write to the session.*
- virtual std::optional< std::any > [getParameterDefault](#) (const [Ghoti::Wave::ServerParameter](#) &parameter)  
override  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterDefault](#) ([[maybe\_unused]]const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Provide a default value for the provided parameter key.*
- virtual std::optional< std::any > [getParameterAny](#) (const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Gets the named parameter if it exists, checking locally first, then checking the global defaults.*
- const std::optional< U > [getParameter](#) (const [Ghoti::Wave::ServerParameter](#) &parameter)  
*Get the parameter as a specified type.*
- virtual [HasParameters](#) & [setParameter](#) (const [Ghoti::Wave::ServerParameter](#) &parameter, const std::any &value)  
*Set a parameter.*
- const [ParameterMap](#)< [Ghoti::Wave::ServerParameter](#) > & [getAllParameters](#) () const  
*Get the current explicitly-set parameters and their values.*

## Public Attributes

- std::unique\_ptr< std::mutex > [controlMutex](#)  
*Used to synchronize access to the session to make it thread safe.*
- std::unique\_ptr< std::condition\_variable > [controlConditionVariable](#)  
*Used to synchronize access to the session to make it thread safe.*

## Private Attributes

- int [hClient](#)  
*The socket handle to the client.*
- size\_t [requestSequence](#)  
*A monotonically increasing counter to track request/response pairs.*
- size\_t [writeOffset](#)  
*A byte offset used to track how many bytes of a message have been written, so that individual write attempts do not duplicate data.*
- bool [working](#)  
*Tracks whether or not the session has work queued.*

- bool [finished](#)  
*Tracks whether or not the session has completed all pending communications.*
- [RequestParser](#) [parser](#)  
*The parser object used to parse the raw HTTP stream.*
- [Server](#) \* [server](#)  
*A pointer to the server object.*
- std::map< uint64\_t, std::pair< std::shared\_ptr< [Message](#) >, std::shared\_ptr< [Message](#) > > > [messages](#)  
*Tracks request/response pairs.*
- std::queue< uint64\_t > [pipeline](#)  
*Simple queue to track which request sequence # should be parsed next.*
- [ParameterMap](#)< [Ghoti::Wave::ServerParameter](#) > [parameterValues](#)  
*Store explicitly set parameter key/value pairs.*

### 4.12.1 Detailed Description

Represents a persistent connection with a client.

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 ServerSession()

```
ServerSession::ServerSession (
    int hClient,
    Server * server )
```

The constructor.

#### Parameters

<i>hClient</i>	The socket handle to the client connection.
<i>server</i>	A pointer to the parent <a href="#">Server</a> object.

### 4.12.3 Member Function Documentation

#### 4.12.3.1 getAllParameters()

```
const ParameterMap<Ghoti::Wave::ServerParameter >& Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >::getAllParameters ( ) const [inline], [inherited]
```

Get the current explicitly-set parameters and their values.

**Returns**

The current explicitly-set parameters.

**4.12.3.2 getParameter()**

```
const std::optional<U> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >::get↔
Parameter (
    const T & parameter ) [inline], [inherited]
```

Get the parameter as a specified type.

The result is returned as an optional. If there is no parameter value, then the optional value will be false.

**Parameters**

<i>parameter</i>	The parameter value to get.
------------------	-----------------------------

**Returns**

The (optional) parameter value.

**4.12.3.3 getParameterAny()**

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >↔
::getParameterAny (
    const T & parameter ) [inline], [virtual], [inherited]
```

Gets the named parameter if it exists, checking locally first, then checking the global defaults.

**Parameters**

<i>parameter</i>	The parameter to get.
------------------	-----------------------

**Returns**

The parameter value if it exists.

**4.12.3.4 getParameterDefault() [1/2]**

```
virtual std::optional<std::any> Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >↔
::getParameterDefault (
    [[maybe_unused]] const T & parameter ) [inline], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

#### Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

#### Returns

The associated value.

#### 4.12.3.5 getParameterDefault() [2/2]

```
optional< any > HasServerParameters::getParameterDefault (
    const Ghoti::Wave::ServerParameter & parameter ) [override], [virtual], [inherited]
```

Provide a default value for the provided parameter key.

The default behavior of this function is to only return an empty optional value. The intent is for this to be overridden by subclasses.

#### Parameters

<i>parameter</i>	The parameter key to fetch.
------------------	-----------------------------

#### Returns

The associated value.

#### 4.12.3.6 hasReadDataWaiting()

```
bool ServerSession::hasReadDataWaiting ( )
```

Checks to see whether or not the session has data waiting to be read from the socket.

This is non-blocking mutex controlled. If the session is currently working, then this function will return false.

#### Returns

Whether or not the session has data waiting to be read from the socket.

#### 4.12.3.7 hasWriteDataWaiting()

```
bool ServerSession::hasWriteDataWaiting ( )
```

Checks to see whether or not the session has data waiting to be written to the socket.

This is non-blocking mutex controlled. If the session is currently working, then this function will return false.

##### Returns

Whether or not the session has data waiting to be written to the socket.

#### 4.12.3.8 isFinished()

```
bool ServerSession::isFinished ( )
```

Indicates whether or not the session has completed all communications and may be terminated.

##### Returns

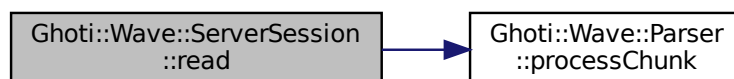
true if all communications have completed, false otherwise.

#### 4.12.3.9 read()

```
void ServerSession::read ( )
```

Perform a read from the session.

This function is intended to be called by the server's thread pool worker queue, probably in a lambda expression. Here is the call graph for this function:



#### 4.12.3.10 setParameter()

```
virtual HasParameters& Ghoti::Wave::HasParameters< Ghoti::Wave::ServerParameter >::setParameter
(
    const T & parameter,
    const std::any & value ) [inline], [virtual], [inherited]
```

Set a parameter.

#### Parameters

<i>parameter</i>	The parameter key to be set.
<i>value</i>	The parameter value to be set.

#### Returns

The calling object, to allow for chaining.

Reimplemented in [Ghoti::Wave::Server](#).

#### 4.12.3.11 write()

```
void ServerSession::write ( )
```

Perform a write to the session.

This function is intended to be called by the server's thread pool worker queue, probably in a lambda expression.

### 4.12.4 Member Data Documentation

#### 4.12.4.1 messages

```
std::map<uint64_t, std::pair<std::shared_ptr<Message>, std::shared_ptr<Message> > > Ghoti←  
::Wave::ServerSession::messages [private]
```

Tracks request/response pairs.

```
messages[request sequence #] = <request, response>
```

The documentation for this class was generated from the following files:

- [include/wave/serverSession.hpp](#)
- [src/serverSession.cpp](#)





## Chapter 5

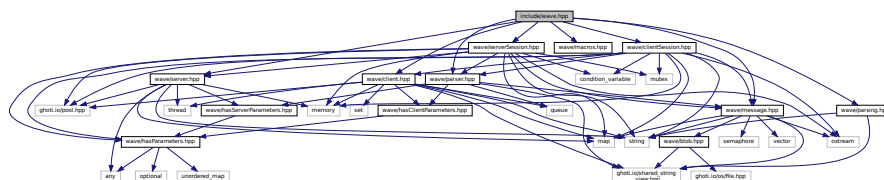
# File Documentation

### 5.1 include/wave.hpp File Reference

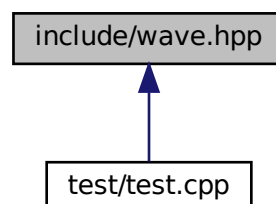
Header file supplied for use by 3rd party code so that they can easily include all necessary headers for the Ghoti.io Wave library.

```
#include "wave/client.hpp"
#include "wave/clientSession.hpp"
#include "wave/macros.hpp"
#include "wave/message.hpp"
#include "wave/parser.hpp"
#include "wave/parsing.hpp"
#include "wave/server.hpp"
#include "wave/serverSession.hpp"
```

Include dependency graph for wave.hpp:



This graph shows which files directly or indirectly include this file:



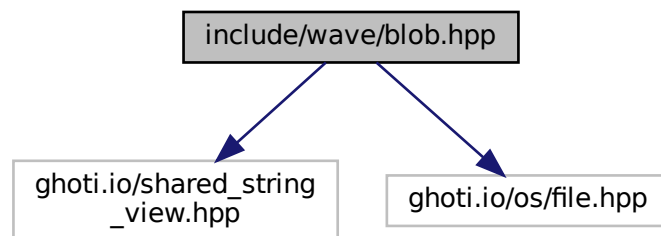
### 5.1.1 Detailed Description

Header file supplied for use by 3rd party code so that they can easily include all necessary headers for the Ghoti.io Wave library.

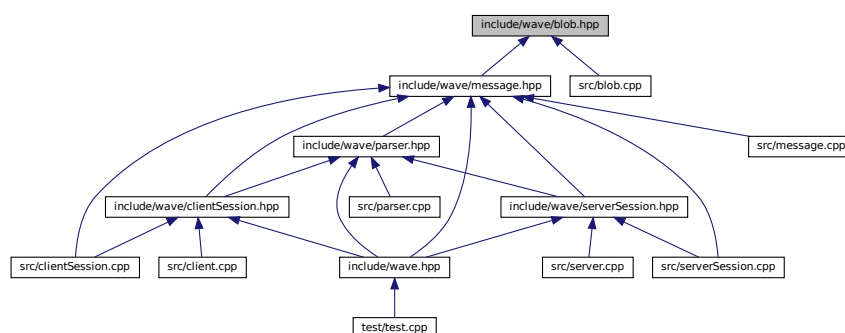
## 5.2 include/wave/blob.hpp File Reference

Header file for declaring the Blob class.

```
#include <ghoti.io/shared_string_view.hpp>
#include <ghoti.io/os/file.hpp>
Include dependency graph for blob.hpp:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [Ghoti::Wave::Blob](#)

The *Blob* class is a generic container which may reference text (binary or otherwise) either in-memory or on-disk (e.g., in a file).

## Functions

- `std::ostream & Ghoti::Wave::operator<< (std::ostream &out, const Blob &blob)`  
*Helper function to output a [Blob](#) to a stream.*

### 5.2.1 Detailed Description

Header file for declaring the Blob class.

### 5.2.2 Function Documentation

#### 5.2.2.1 `operator<<()`

```
ostream & Ghoti::Wave::operator<< (
    std::ostream & out,
    const Blob & blob )
```

Helper function to output a Blob to a stream.

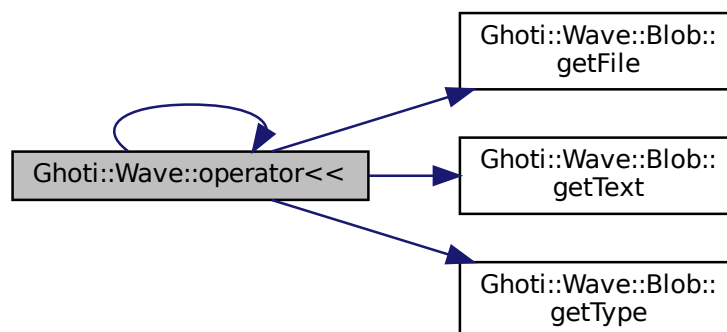
##### Parameters

<i>out</i>	The output stream.
<i>blob</i>	The Blob to be inserted into the stream.

##### Returns

The output stream.

Here is the call graph for this function:

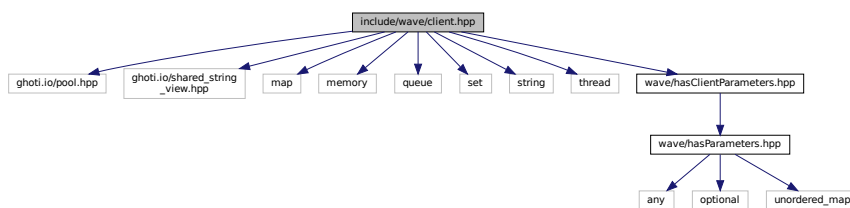


### 5.3 include/wave/client.hpp File Reference

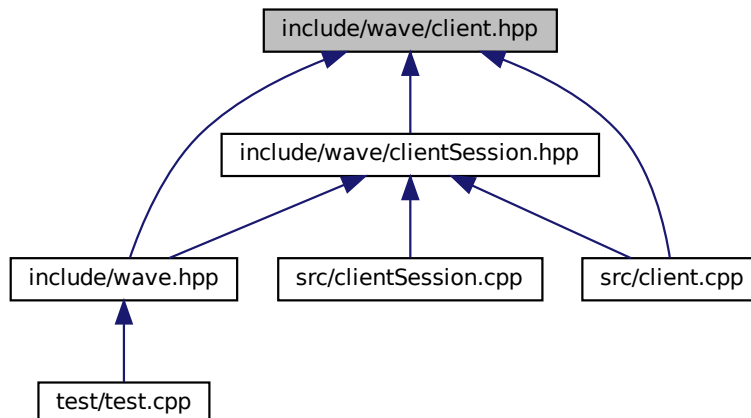
Header file for declaring the Client class.

```
#include <ghoti.io/pool.hpp>
#include <ghoti.io/shared_string_view.hpp>
#include <map>
#include <memory>
#include <queue>
#include <set>
#include <string>
#include <thread>
#include "wave/hasClientParameters.hpp"
```

Include dependency graph for client.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Ghoti::Wave::Client](#)

*Represents a client and all of its HTTP connections.*

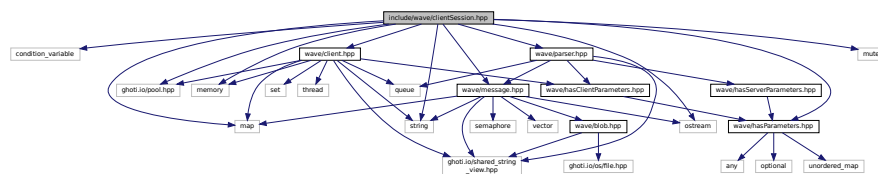
### 5.3.1 Detailed Description

Header file for declaring the Client class.

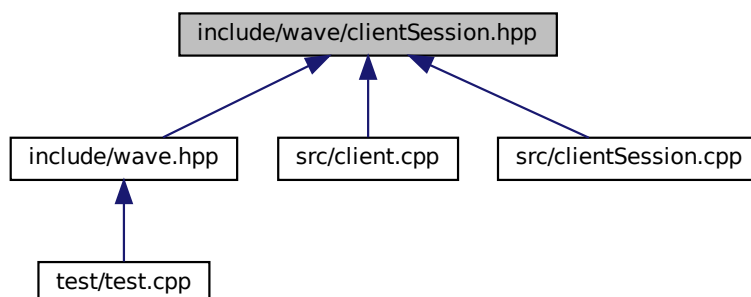
## 5.4 include/wave/clientSession.hpp File Reference

Header file for declaring the ClientSession class.

```
#include <condition_variable>
#include <ghoti.io/pool.hpp>
#include <memory>
#include <mutex>
#include <ostream>
#include <map>
#include <string>
#include "wave/client.hpp"
#include "wave/hasParameters.hpp"
#include "wave/message.hpp"
#include "wave/parser.hpp"
Include dependency graph for clientSession.hpp:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [Ghoti::Wave::ClientSession](#)

*Represents a connection to a particular domain/port pair.*

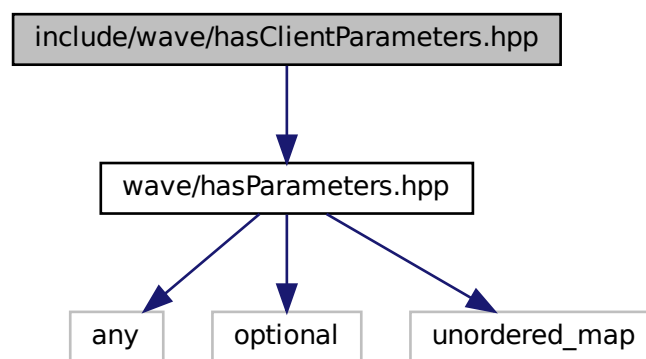
### 5.4.1 Detailed Description

Header file for declaring the ClientSession class.

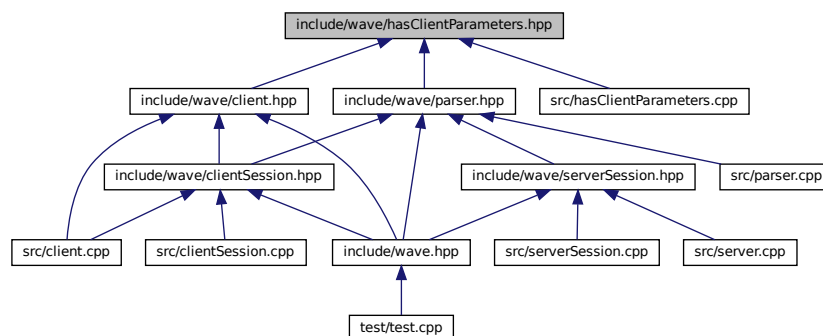
## 5.5 include/wave/hasClientParameters.hpp File Reference

Header file for declaring the HasClientParameters class.

```
#include "wave/hasParameters.hpp"
Include dependency graph for hasClientParameters.hpp:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [Ghoti::Wave::HasClientParameters](#)

*Base class to provide consistent defaults to [Server](#) and [ServerSession](#) classes.*

## Enumerations

- enum class [Ghoti::Wave::ClientParameter](#) { [MAXBUFFERSIZE](#) , [MEMCHUNKSIZELIMIT](#) }  
*Sessings parameters which influence the behavior of Wave and its components.*

### 5.5.1 Detailed Description

Header file for declaring the HasClientParameters class.

### 5.5.2 Enumeration Type Documentation

#### 5.5.2.1 ClientParameter

```
enum Ghoti::Wave::ClientParameter [strong]
```

Sessings parameters which influence the behavior of Wave and its components.

##### Enumerator

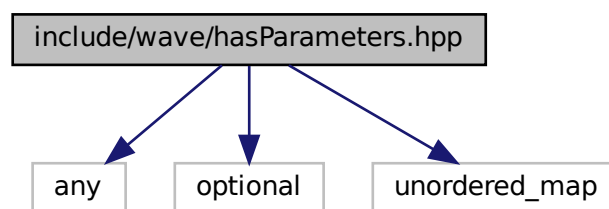
<a href="#">MAXBUFFERSIZE</a>	The read/write buffer size used when interacting with sockets.
<a href="#">MEMCHUNKSIZELIMIT</a>	The maximum size in bytes allowed for a chunk before converting the chunk to a file.

## 5.6 include/wave/hasParameters.hpp File Reference

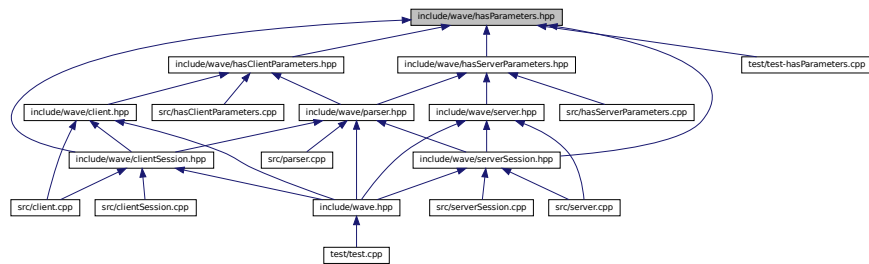
Header file for declaring the hasParameters class.

```
#include <any>
#include <optional>
#include <unordered_map>
```

Include dependency graph for hasParameters.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Ghoti::Wave::HasParameters< T >](#)  
Serves as a base class for any other class to have settings parameters.

## Typedefs

- template<typename T >  
using [Ghoti::Wave::ParameterMap](#) = std::unordered\_map< T, std::any >  
A type alias for the structure that stores the settings map.

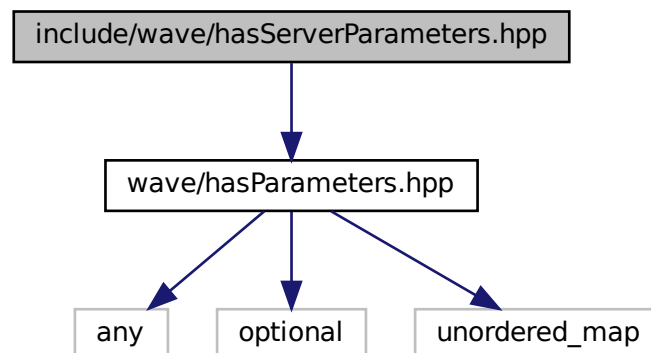
### 5.6.1 Detailed Description

Header file for declaring the hasParameters class.

## 5.7 include/wave/hasServerParameters.hpp File Reference

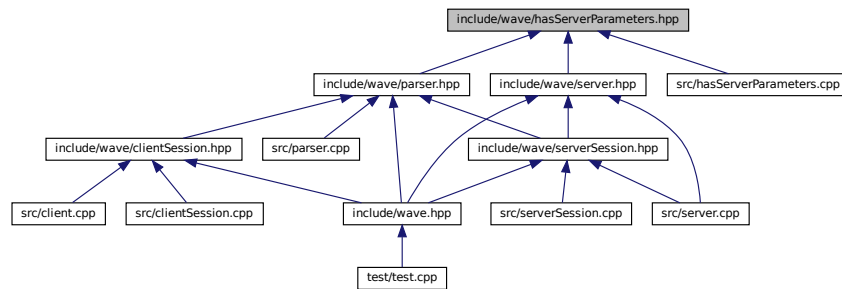
Header file for declaring the HasServerParameters class.

```
#include "wave/hasParameters.hpp"
Include dependency graph for hasServerParameters.hpp:
```





This graph shows which files directly or indirectly include this file:



## Classes

- class [Ghoti::Wave::HasServerParameters](#)

Base class to provide consistent defaults to [Server](#) and [ServerSession](#) classes.

## Enumerations

- enum class [Ghoti::Wave::ServerParameter](#) { [MAXBUFFERSIZE](#) , [MEMCHUNKSIZELIMIT](#) }

Sessings parameters which influence the behavior of a Server.

### 5.7.1 Detailed Description

Header file for declaring the HasServerParameters class.

### 5.7.2 Enumeration Type Documentation

#### 5.7.2.1 ServerParameter

```
enum Ghoti::Wave::ServerParameter [strong]
```

Sessings parameters which influence the behavior of a Server.

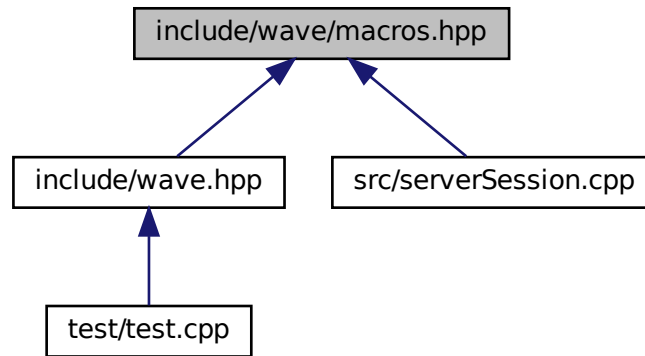
#### Enumerator

<a href="#">MAXBUFFERSIZE</a>	The read/write buffer size used when interacting with sockets.
<a href="#">MEMCHUNKSIZELIMIT</a>	The maximum size in bytes allowed for a chunk before converting the chunk to a file.

## 5.8 include/wave/macros.hpp File Reference

Header file for declaring the Client class.

This graph shows which files directly or indirectly include this file:



### 5.8.1 Detailed Description

Header file for declaring the Client class.

## 5.9 include/wave/message.hpp File Reference

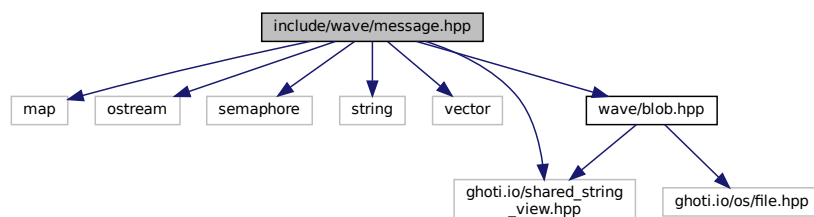
Header file for declaring the Message class.

```

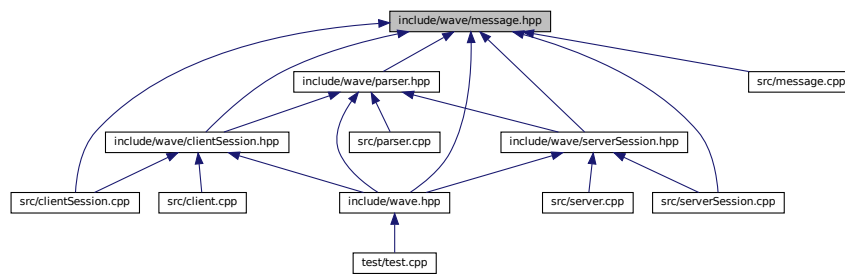
#include <map>
#include <ostream>
#include <semaphore>
#include <string>
#include <vector>
#include <ghoti.io/shared_string_view.hpp>
#include "wave/blob.hpp"

```

Include dependency graph for message.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Ghoti::Wave::Message](#)  
*Represents a HTTP message.*

## Functions

- `std::ostream & Ghoti::Wave::operator<< (std::ostream &out, Message &message)`  
*Helper function to output a [Message](#) to a stream.*

### 5.9.1 Detailed Description

Header file for declaring the Message class.

### 5.9.2 Function Documentation

#### 5.9.2.1 `operator<<()`

```
ostream & Ghoti::Wave::operator<< (
    std::ostream & out,
    Message & message )
```

Helper function to output a Message to a stream.

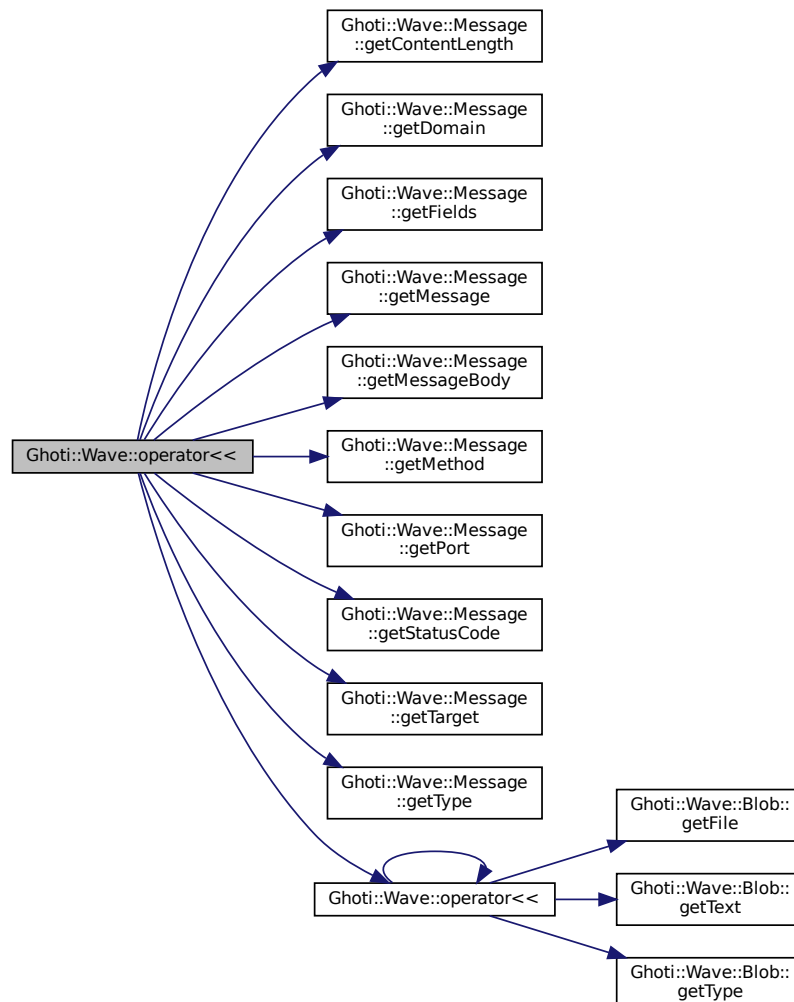
#### Parameters

<i>out</i>	The output stream.
<i>message</i>	The Message to be inserted into the stream.

**Returns**

The output stream.

Here is the call graph for this function:



## 5.10 include/wave/parser.hpp File Reference

Header file for declaring the Session class.

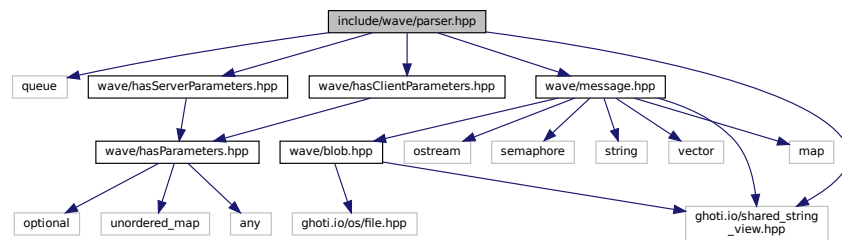
```

#include <queue>
#include <ghoti.io/shared_string_view.hpp>
#include "wave/hasClientParameters.hpp"
#include "wave/hasServerParameters.hpp"

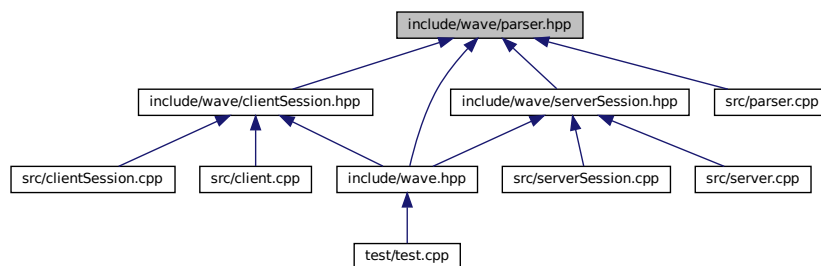
```

```
#include "wave/message.hpp"
```

Include dependency graph for parser.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Ghoti::Wave::Parser](#)  
*Parses a HTTP/1.1 data stream into discrete messages.*
- class [Ghoti::Wave::RequestParser](#)  
*Specialized class for handling the request parser, to make it easier to pass in ServerParameters.*
- class [Ghoti::Wave::ResponseParser](#)  
*Specialized class for handling the response parser, to make it easier to pass in ClientParameters.*

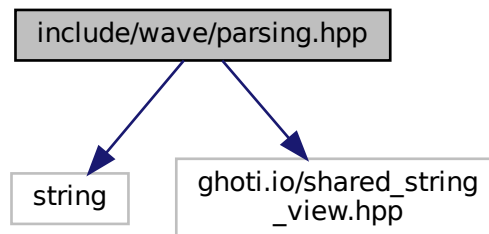
### 5.10.1 Detailed Description

Header file for declaring the Session class.

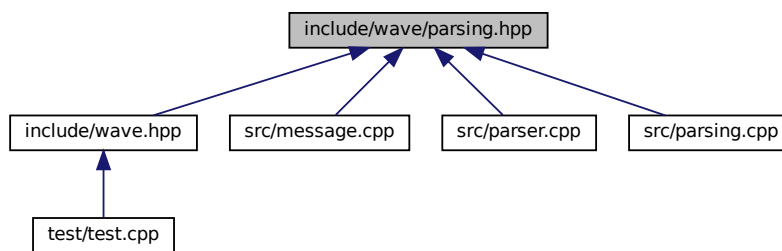
## 5.11 include/wave/parsing.hpp File Reference

Header file for declaring text parsing functions.

```
#include <string>
#include <ghoti.io/shared_string_view.hpp>
Include dependency graph for parsing.hpp:
```



This graph shows which files directly or indirectly include this file:



## Functions

- bool [Ghoti::Wave::isListField](#) (const Ghoti::shared\_string\_view &name)  
*Identify a field name as accepting a list-based set of values.*
- bool [Ghoti::Wave::isTokenChar](#) (uint8\_t c)  
*Identify valid Token characters.*
- bool [Ghoti::Wave::isWhitespaceChar](#) (uint8\_t c)  
*Identify valid whitespace characters.*
- bool [Ghoti::Wave::isVisibleChar](#) (uint8\_t c)  
*Identify valid Visible (printing) characters.*
- bool [Ghoti::Wave::isObsoleteTextChar](#) (uint8\_t c)  
*Identify valid obs-text characters.*
- bool [Ghoti::Wave::isFieldNameChar](#) (uint8\_t c)  
*Identify valid field-name characters.*
- bool [Ghoti::Wave::isQuotedChar](#) (uint8\_t c)  
*Identify valid quoted characters.*
- bool [Ghoti::Wave::isFieldContentChar](#) (uint8\_t c)

*Identify valid field-content characters.*

- bool [Ghoti::Wave::isCRLFChar](#) (uint8\_t c)

*Identify CRLF characters.*

- bool [Ghoti::Wave::fieldValueQuotesNeeded](#) (const Ghoti::shared\_string\_view &str)

*Indicate whether or not the string contains a character which makes it necessary to wrap the string in double quotes.*

- std::string [Ghoti::Wave::fieldValueEscape](#) (const Ghoti::shared\_string\_view &str)

*Escape a field value.*

### 5.11.1 Detailed Description

Header file for declaring text parsing functions.

### 5.11.2 Function Documentation

#### 5.11.2.1 fieldValueEscape()

```
std::string Ghoti::Wave::fieldValueEscape (  
    const Ghoti::shared_string_view & str )
```

Escape a field value.

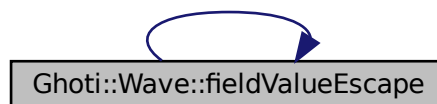
##### Parameters

<i>str</i>	The field value to be escaped.
------------	--------------------------------

##### Returns

The escaped field value.

Here is the call graph for this function:



### 5.11.2.2 fieldValueQuotesNeeded()

```
bool Ghoti::Wave::fieldValueQuotesNeeded (
    const Ghoti::shared_string_view & str )
```

Indicate whether or not the string contains a character which makes it necessary to wrap the string in double quotes.

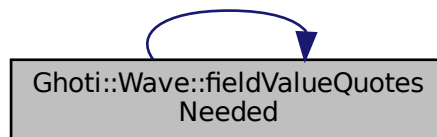
#### Parameters

<i>str</i>	The string in question.
------------	-------------------------

#### Returns

Whether or not the string needs to be wrapped in double quotes.

Here is the call graph for this function:



### 5.11.2.3 isCRLFChar()

```
bool Ghoti::Wave::isCRLFChar (
    uint8_t c )
```

Identify CRLF characters.

#### Parameters

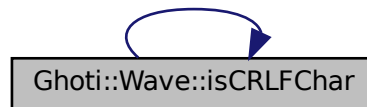
<i>c</i>	The character to test.
----------	------------------------



**Returns**

Whether or not the character is a valid CRLF character.

Here is the call graph for this function:

**5.11.2.4 isFieldContentChar()**

```
bool Ghoti::Wave::isFieldContentChar (
    uint8_t c )
```

Identify valid field-content characters.

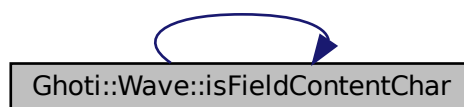
**Parameters**

<code>c</code>	The character to test.
----------------	------------------------

**Returns**

Whether or not the character is a valid field-content character.

Here is the call graph for this function:



### 5.11.2.5 isFieldNameChar()

```
bool Ghoti::Wave::isFieldNameChar (
    uint8_t c )
```

Identify valid field-name characters.

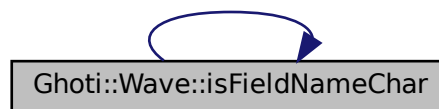
#### Parameters

<i>c</i>	The character to test.
----------	------------------------

#### Returns

Whether or not the character is a valid field-name character.

Here is the call graph for this function:



### 5.11.2.6 isListField()

```
bool Ghoti::Wave::isListField (
    const Ghoti::shared_string_view & name )
```

Identify a field name as accepting a list-based set of values.

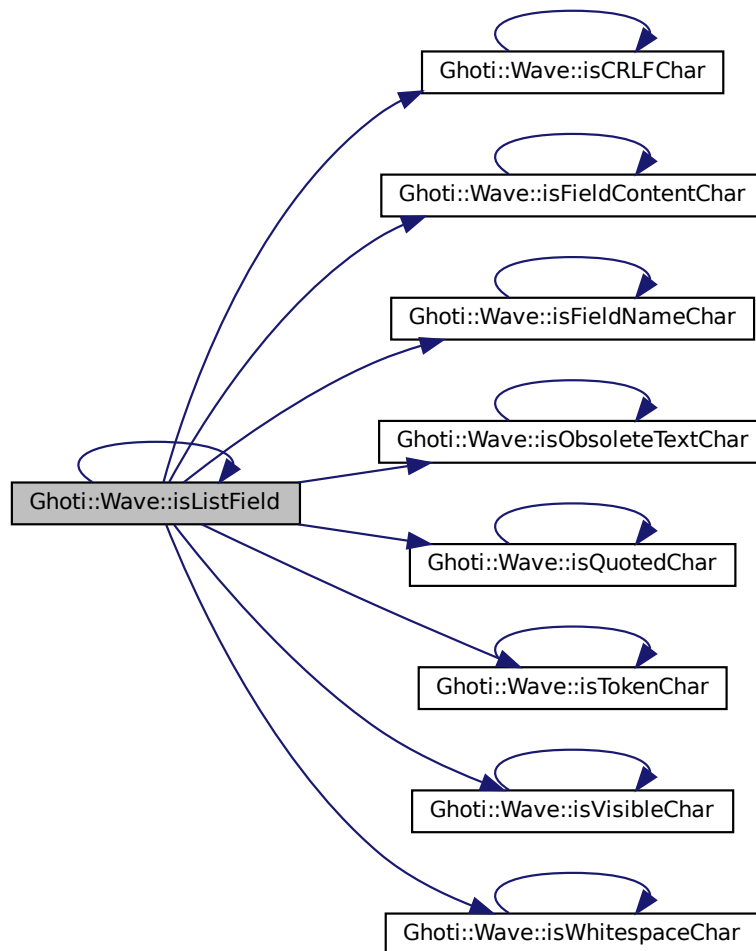
#### Parameters

<i>name</i>	The field name. The field name must be uppercase.
-------------	---

**Returns**

Whether or not the field name is recognized as a list-based field.

Here is the call graph for this function:

**5.11.2.7 isObsoleteTextChar()**

```
bool Ghoti::Wave::isObsoleteTextChar (
    uint8_t c )
```

Identify valid obs-text characters.

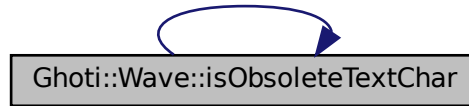
**Parameters**

<code>c</code>	The character to test.
----------------	------------------------

**Returns**

Whether or not the character is a valid obs-text character.

Here is the call graph for this function:

**5.11.2.8 isQuotedChar()**

```
bool Ghoti::Wave::isQuotedChar (
    uint8_t c )
```

Identify valid quoted characters.

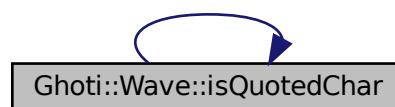
**Parameters**

<code>c</code>	The character to test.
----------------	------------------------

**Returns**

Whether or not the character is a valid quoted character.

Here is the call graph for this function:



### 5.11.2.9 isTokenChar()

```
bool Ghoti::Wave::isTokenChar (
    uint8_t c )
```

Identify valid Token characters.

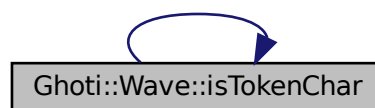
#### Parameters

<i>c</i>	The character to test.
----------	------------------------

#### Returns

Whether or not the character is a valid token character.

Here is the call graph for this function:



### 5.11.2.10 isVisibleChar()

```
bool Ghoti::Wave::isVisibleChar (
    uint8_t c )
```

Identify valid Visible (printing) characters.

#### Parameters

<i>c</i>	The character to test.
----------	------------------------

**Returns**

Whether or not the character is a valid visible character.

Here is the call graph for this function:

**5.11.2.11 isWhitespaceChar()**

```
bool Ghoti::Wave::isWhitespaceChar (
    uint8_t c )
```

Identify valid whitespace characters.

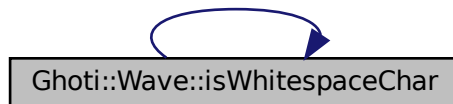
**Parameters**

<code>c</code>	The character to test.
----------------	------------------------

**Returns**

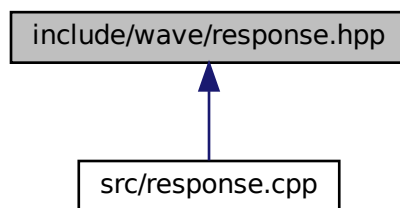
Whether or not the character is a valid visible character.

Here is the call graph for this function:

**5.12 include/wave/response.hpp File Reference**

Header file for declaring the Response class.

This graph shows which files directly or indirectly include this file:



### 5.12.1 Detailed Description

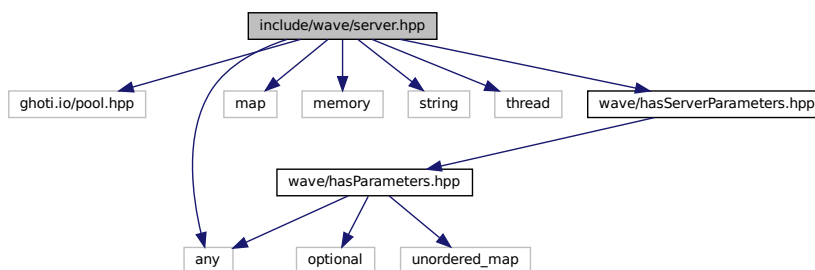
Header file for declaring the Response class.

## 5.13 include/wave/server.hpp File Reference

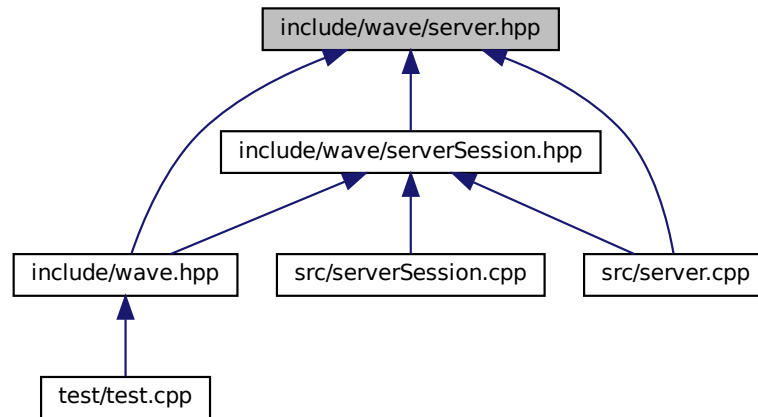
Header file for declaring the Server class.

```
#include <ghoti.io/pool.hpp>
#include <any>
#include <map>
#include <memory>
#include <string>
#include <thread>
#include "wave/hasServerParameters.hpp"
```

Include dependency graph for server.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Ghoti::Wave::Server](#)  
The base [Server](#) class.

### 5.13.1 Detailed Description

Header file for declaring the Server class.

## 5.14 include/wave/serverSession.hpp File Reference

Header file for declaring the ServerSession class.

```

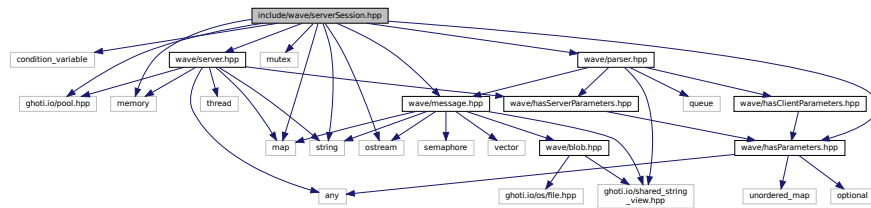
#include <condition_variable>
#include <ghoti.io/pool.hpp>
#include <memory>
#include <map>
#include <mutex>
#include <ostream>
#include <string>
#include "wave/hasParameters.hpp"
#include "wave/message.hpp"
#include "wave/parser.hpp"

```

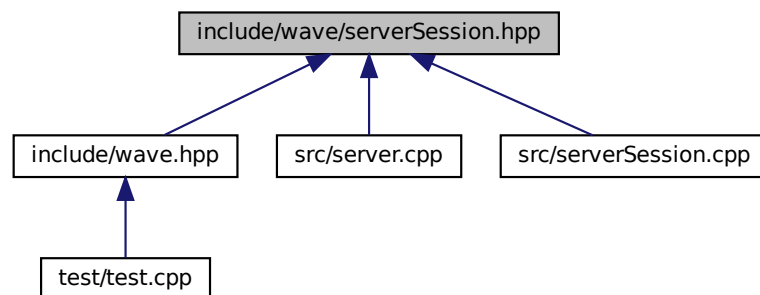


```
#include "wave/server.hpp"
```

Include dependency graph for serverSession.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Ghoti::Wave::ServerSession](#)  
*Represents a persistent connection with a client.*

### 5.14.1 Detailed Description

Header file for declaring the ServerSession class.

## 5.15 src/blob.cpp File Reference

Define the [Ghoti::Wave::Blob](#) class.



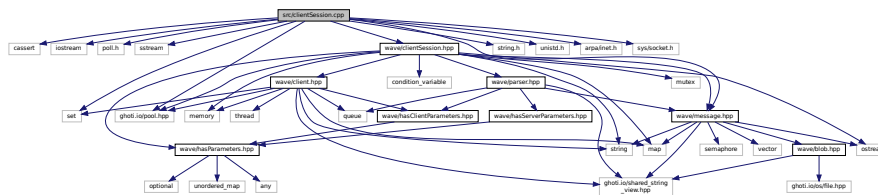
### 5.16.1 Detailed Description

Define the [Ghoti::Wave::Client](#) class.

## 5.17 src/clientSession.cpp File Reference

Define the [Ghoti::Wave::ClientSession](#) class.

```
#include <cassert>
#include <iostream>
#include <poll.h>
#include <sstream>
#include <set>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <ghoti.io/pool.hpp>
#include <sys/socket.h>
#include "wave/clientSession.hpp"
#include "wave/message.hpp"
Include dependency graph for clientSession.cpp:
```



### 5.17.1 Detailed Description

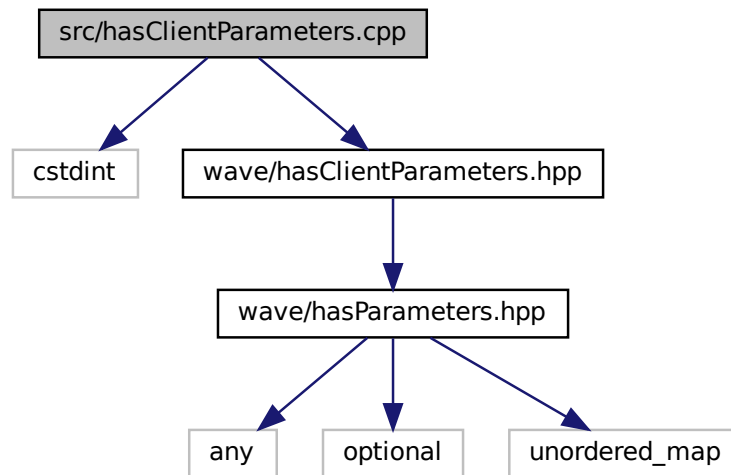
Define the [Ghoti::Wave::ClientSession](#) class.

## 5.18 src/hasClientParameters.cpp File Reference

Define the [Ghoti::Wave::HasClientParameters](#) class.

```
#include <cstdint>
#include "wave/hasClientParameters.hpp"
```

Include dependency graph for `hasClientParameters.cpp`:



### 5.18.1 Detailed Description

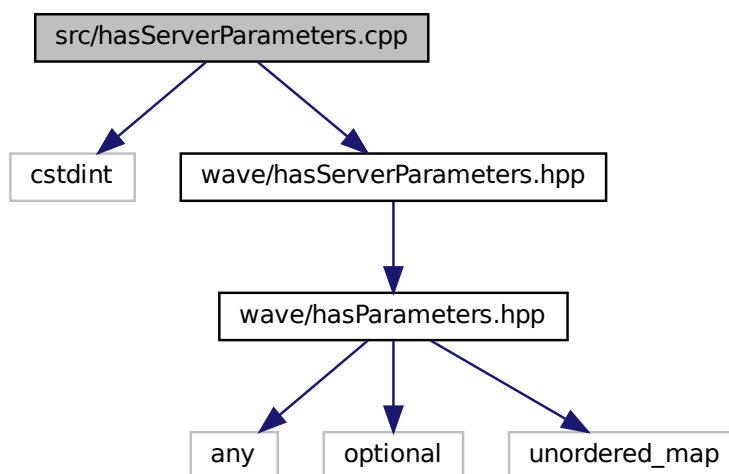
Define the [Ghoti::Wave::HasClientParameters](#) class.

## 5.19 `src/hasServerParameters.cpp` File Reference

Define the [Ghoti::Wave::HasServerParameters](#) class.

```
#include <stdint>
#include "wave/hasServerParameters.hpp"
```

Include dependency graph for hasServerParameters.cpp:



### 5.19.1 Detailed Description

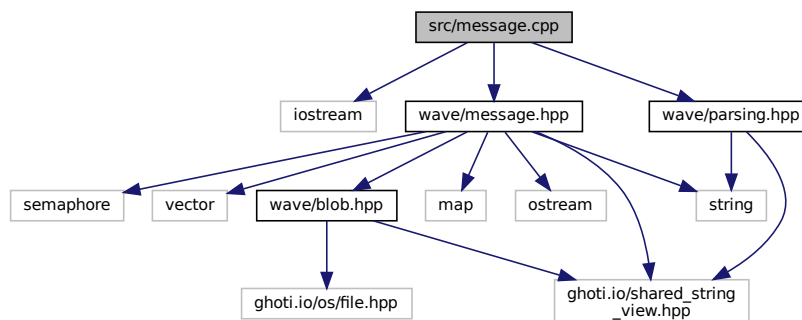
Define the [Ghoti::Wave::HasServerParameters](#) class.

## 5.20 src/message.cpp File Reference

Define the [Ghoti::Wave::Message](#) class.

```
#include <iostream>
#include "wave/message.hpp"
#include "wave/parsing.hpp"
```

Include dependency graph for message.cpp:



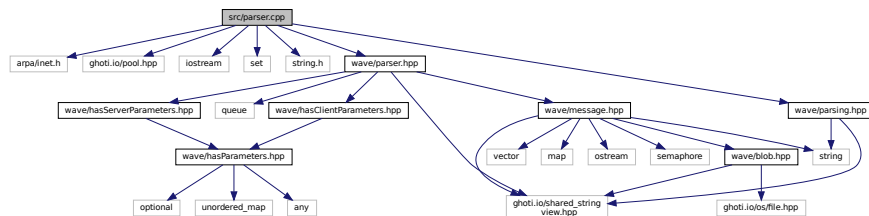
### 5.20.1 Detailed Description

Define the [Ghoti::Wave::Message](#) class.

## 5.21 src/parser.cpp File Reference

Define the [Ghoti::Wave::Parser](#) class.

```
#include <arpa/inet.h>
#include <ghoti.io/pool.hpp>
#include <iostream>
#include <set>
#include <string.h>
#include "wave/parser.hpp"
#include "wave/parsing.hpp"
Include dependency graph for parser.cpp:
```



## Macros

- `#define START_NEW_INPUT`
- `#define SET_NEW_HEADER`
- `#define SET_MINOR_STATE(nextState)`
- `#define SET_MAJOR_STATE(nextMajorState, nextMinorState)`
- `#define READ_WHITESPACE_OPTIONAL(nextState)`
- `#define READ_WHITESPACE_REQUIRED(nextState, statusCode, errorMessage)`
- `#define READ_CRLF_OPTIONAL(nextState)`
- `#define READ_CRLF_REQUIRED(nextState, statusCode, errorMessage)`
- `#define REQUEST_STATUS_ERROR (this->type == REQUEST ? "Error reading request line." : "Error reading status line.")`

### 5.21.1 Detailed Description

Define the [Ghoti::Wave::Parser](#) class.

### 5.21.2 Macro Definition Documentation

## 5.21.2.1 READ\_CRLF\_OPTIONAL

```
#define READ_CRLF_OPTIONAL(  
    nextState )
```

**Value:**

```
while (this->cursor < input_length) { \
    if ((this->input[this->cursor] != '\r') && (this->input[this->cursor] != '\n')) { \
        SET_MINOR_STATE(nextState); \
        break; \
    } \
    ++this->cursor; \
}
```

## 5.21.2.2 READ\_CRLF\_REQUIRED

```
#define READ_CRLF_REQUIRED(  
    nextState,  
    statusCode,  
    errorMessage )
```

**Value:**

```
size_t len = this->cursor - this->minorStart; \
while ((this->cursor < input_length) && (len < 2)) { \
    if (((len == 0) && !((this->input[this->cursor] == '\r') || (this->input[this->cursor] == '\n'))) \
        || ((len == 1) && (this->input[this->cursor] != '\n'))) { \
        this->currentMessage->setStatusCode(statusCode).setErrorMessage(errorMessage); \
    } \
    if (!this->currentMessage->hasError() && (this->input[this->cursor] == '\n')) { \
        SET_MINOR_STATE(nextState); \
        ++this->cursor; \
        break; \
    } \
    ++this->cursor; \
    ++len; \
}
```

## 5.21.2.3 READ\_WHITESPACE\_OPTIONAL

```
#define READ_WHITESPACE_OPTIONAL(  
    nextState )
```

**Value:**

```
while ((this->cursor < input_length) && ( \
    isspace(this->input[this->cursor]) \
    && (this->input[this->cursor] != '\n') \
    && (this->input[this->cursor] != '\r'))) { \
    ++this->cursor; \
} \
if ((this->cursor < input_length) && ( \
    !isspace(this->input[this->cursor]) \
    || (this->input[this->cursor] == '\n') \
    || (this->input[this->cursor] == '\r'))) { \
    SET_MINOR_STATE(nextState); \
}
```

### 5.21.2.4 READ\_WHITESPACE\_REQUIRED

```
#define READ_WHITESPACE_REQUIRED(
    nextState,
    statusCode,
    errorMessage )
```

**Value:**

```
while ((this->cursor < input_length) && ( \
    isspace(this->input[this->cursor]) \
    && (this->input[this->cursor] != '\n') \
    && (this->input[this->cursor] != '\r'))) { \
    ++this->cursor; \
} \
if (this->cursor < input_length) { \
    if (this->cursor > this->minorStart) { \
        SET_MINOR_STATE(nextState); \
    } \
    else { \
        this->currentMessage->setStatusCode(statusCode).setErrorMessage(errorMessage); \
    } \
}
```

### 5.21.2.5 SET\_MAJOR\_STATE

```
#define SET_MAJOR_STATE(
    nextMajorState,
    nextMinorState )
```

**Value:**

```
this->readStateMajor = nextMajorState; \
this->majorStart = this->cursor; \
SET_MINOR_STATE(nextMinorState);
```

### 5.21.2.6 SET\_MINOR\_STATE

```
#define SET_MINOR_STATE(
    nextState )
```

**Value:**

```
this->readStateMinor = nextState; \
this->minorStart = this->cursor;
```

### 5.21.2.7 SET\_NEW\_HEADER

```
#define SET_NEW_HEADER
```

**Value:**

```
this->readStateMajor = NEW_HEADER; \
this->readStateMinor = this->type == REQUEST \
    ? BEGINNING_OF_REQUEST_LINE \
    : BEGINNING_OF_STATUS_LINE; \
this->majorStart = this->cursor; \
this->minorStart = this->cursor; \
this->contentLength = 0;
```



## 5.21.2.8 START\_NEW\_INPUT

```
#define START_NEW_INPUT
```

**Value:**

```
this->input = string{this->input.substr(this->cursor, this->input.length())}; \
this->cursor = 0; \
input_length = this->input.length();
```

## 5.22 src/parsing.cpp File Reference

Define the text parsing functions.

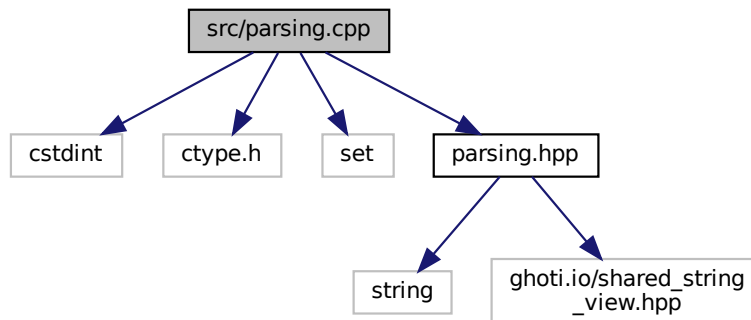
```
#include <cstdint>
```

```
#include <ctype.h>
```

```
#include <set>
```

```
#include "parsing.hpp"
```

Include dependency graph for parsing.cpp:

**Functions**

- bool **Ghoti::Wave::isListField** (const shared\_string\_view &name)
- bool **Ghoti::Wave::isTokenChar** (uint8\_t c)  
*Identify valid Token characters.*
- bool **Ghoti::Wave::isWhitespaceChar** (uint8\_t c)  
*Identify valid whitespace characters.*
- bool **Ghoti::Wave::isVisibleChar** (uint8\_t c)  
*Identify valid Visible (printing) characters.*
- bool **Ghoti::Wave::isObsoleteTextChar** (uint8\_t c)  
*Identify valid obs-text characters.*
- bool **Ghoti::Wave::isFieldNameChar** (uint8\_t c)  
*Identify valid field-name characters.*
- bool **Ghoti::Wave::isQuotedChar** (uint8\_t c)  
*Identify valid quoted characters.*
- bool **Ghoti::Wave::isFieldContentChar** (uint8\_t c)  
*Identify valid field-content characters.*
- bool **Ghoti::Wave::isCRLFChar** (uint8\_t c)  
*Identify CRLF characters.*
- bool **Ghoti::Wave::fieldValueQuotesNeeded** (const shared\_string\_view &str)
- string **Ghoti::Wave::fieldValueEscape** (const shared\_string\_view &str)

### 5.22.1 Detailed Description

Define the text parsing functions.

### 5.22.2 Function Documentation

#### 5.22.2.1 isCRLFChar()

```
bool Ghoti::Wave::isCRLFChar (
    uint8_t c )
```

Identify CRLF characters.

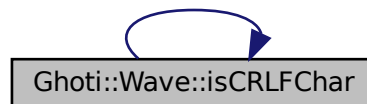
##### Parameters

<code>c</code>	The character to test.
----------------	------------------------

##### Returns

Whether or not the character is a valid CRLF character.

Here is the call graph for this function:



#### 5.22.2.2 isFieldContentChar()

```
bool Ghoti::Wave::isFieldContentChar (
    uint8_t c )
```

Identify valid field-content characters.

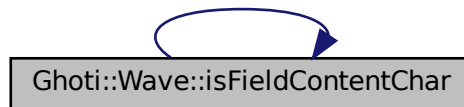
##### Parameters

<code>c</code>	The character to test.
----------------	------------------------

**Returns**

Whether or not the character is a valid field-content character.

Here is the call graph for this function:

**5.22.2.3 isFieldNameChar()**

```
bool Ghoti::Wave::isFieldNameChar (
    uint8_t c )
```

Identify valid field-name characters.

**Parameters**

<code>c</code>	The character to test.
----------------	------------------------

**Returns**

Whether or not the character is a valid field-name character.

Here is the call graph for this function:



#### 5.22.2.4 isObsoleteTextChar()

```
bool Ghoti::Wave::isObsoleteTextChar (
    uint8_t c )
```

Identify valid obs-text characters.

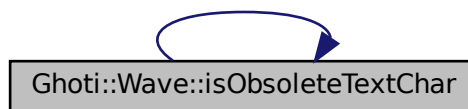
##### Parameters

<i>c</i>	The character to test.
----------	------------------------

##### Returns

Whether or not the character is a valid obs-text character.

Here is the call graph for this function:



#### 5.22.2.5 isQuotedChar()

```
bool Ghoti::Wave::isQuotedChar (
    uint8_t c )
```

Identify valid quoted characters.

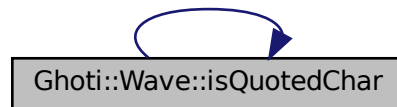
##### Parameters

<i>c</i>	The character to test.
----------	------------------------

**Returns**

Whether or not the character is a valid quoted character.

Here is the call graph for this function:

**5.22.2.6 isTokenChar()**

```
bool Ghoti::Wave::isTokenChar (
    uint8_t c )
```

Identify valid Token characters.

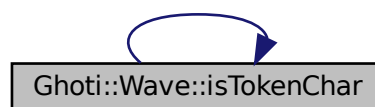
**Parameters**

<code>c</code>	The character to test.
----------------	------------------------

**Returns**

Whether or not the character is a valid token character.

Here is the call graph for this function:



### 5.22.2.7 isVisibleChar()

```
bool Ghoti::Wave::isVisibleChar (
    uint8_t c )
```

Identify valid Visible (printing) characters.

#### Parameters

<i>c</i>	The character to test.
----------	------------------------

#### Returns

Whether or not the character is a valid visible character.

Here is the call graph for this function:



### 5.22.2.8 isWhitespaceChar()

```
bool Ghoti::Wave::isWhitespaceChar (
    uint8_t c )
```

Identify valid whitespace characters.

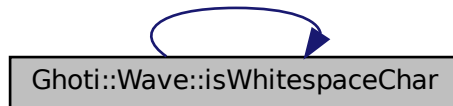
#### Parameters

<i>c</i>	The character to test.
----------	------------------------

### Returns

Whether or not the character is a valid visible character.

Here is the call graph for this function:

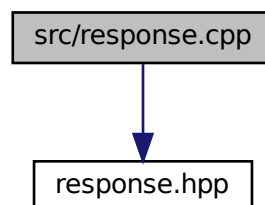


## 5.23 src/response.cpp File Reference

Define the `Ghoti::Wave::Response` class.

```
#include "response.hpp"
```

Include dependency graph for `response.cpp`:



### 5.23.1 Detailed Description

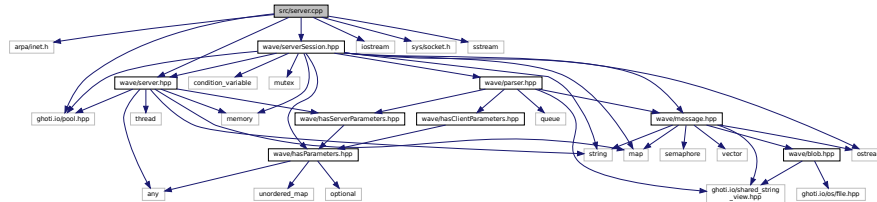
Define the `Ghoti::Wave::Response` class.

## 5.24 src/server.cpp File Reference

Define the [Ghoti::Wave::Server](#) class.

```
#include <arpa/inet.h>
#include <ghoti.io/pool.hpp>
#include <iostream>
```

```
#include <sys/socket.h>
#include <sstream>
#include "wave/server.hpp"
#include "wave/serverSession.hpp"
Include dependency graph for server.cpp:
```



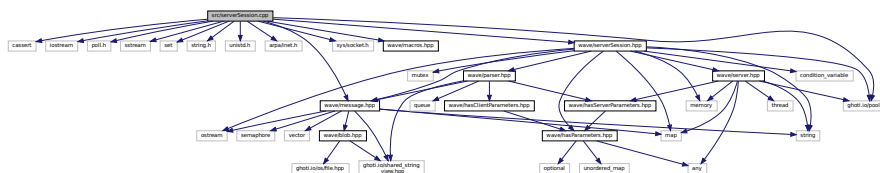
### 5.24.1 Detailed Description

Define the [Ghoti::Wave::Server](#) class.

## 5.25 src/serverSession.cpp File Reference

Define the [Ghoti::Wave::ServerSession](#) class.

```
#include <cassert>
#include <iostream>
#include <poll.h>
#include <sstream>
#include <set>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <ghoti.io/pool.hpp>
#include <sys/socket.h>
#include "wave/macros.hpp"
#include "wave/message.hpp"
#include "wave/serverSession.hpp"
Include dependency graph for serverSession.cpp:
```



### 5.25.1 Detailed Description

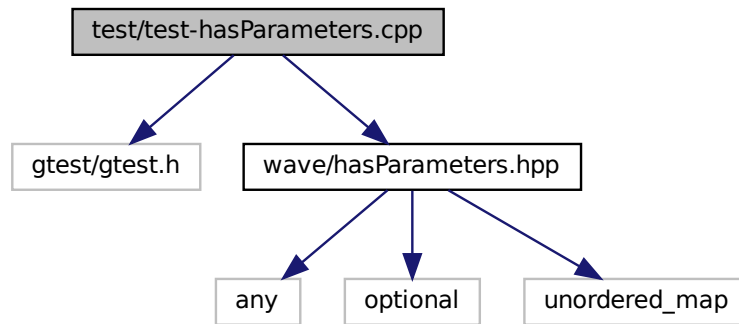
Define the [Ghoti::Wave::ServerSession](#) class.



## 5.26 test/test-hasParameters.cpp File Reference

Test the general Wave server behavior.

```
#include <gtest/gtest.h>
#include "wave/hasParameters.hpp"
Include dependency graph for test-hasParameters.cpp:
```



### Functions

- **TEST** ([HasParameters](#), Default)
- **TEST** (HasParam, Set)
- `int main` (int argc, char \*\*argv)

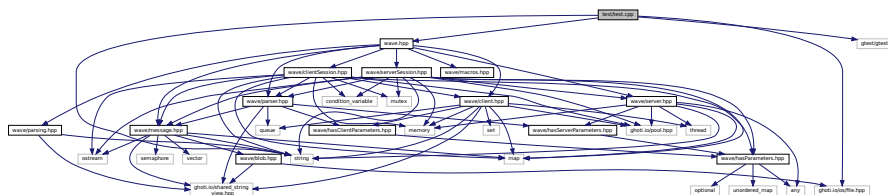
#### 5.26.1 Detailed Description

Test the general Wave server behavior.

## 5.27 test/test.cpp File Reference

Test the general Wave server behavior.

```
#include <string>
#include <gtest/gtest.h>
#include <ghoti.io/os/file.hpp>
#include "wave.hpp"
Include dependency graph for test.cpp:
```



## Functions

- **TEST** ([Blob](#), General)
- **TEST** ([Server](#), Startup)
- **TEST** ([Message](#), Defaults)
- **TEST** ([Message](#), Fields)
- **TEST** (Integration, Simple)
- **TEST** ([Client](#), BufferSize)
- **int main** (int argc, char \*\*argv)

## Variables

- `constexpr auto quantum {10ms}`

### 5.27.1 Detailed Description

Test the general Wave server behavior.

# Index

- ~Server
  - Ghoti::Wave::Server, [80](#)
- addFieldValue
  - Ghoti::Wave::Message, [42](#)
- adoptContents
  - Ghoti::Wave::Message, [42](#)
- AFTER\_CRLF
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- AFTER\_FIELD\_NAME
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- AFTER\_FIELD\_VALUE
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- AFTER\_FIELD\_VALUE\_COMMA
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- AFTER\_HEADER\_FIELDS
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- AFTER\_HTTP\_VERSION
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- AFTER\_METHOD
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- AFTER\_REQUEST\_TARGET
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- append
  - Ghoti::Wave::Blob, [9](#)
- BEFORE\_FIELD\_VALUE
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- BEGINNING\_OF\_FIELD\_LINE
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- BEGINNING\_OF\_REQUEST
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- BEGINNING\_OF\_REQUEST\_LINE
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- BEGINNING\_OF\_STATUS
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- BEGINNING\_OF\_STATUS\_LINE
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- Blob
  - Ghoti::Wave::Blob, [8](#), [9](#)
- blob.hpp
  - operator<<, [97](#)
- CHUNKED
  - Ghoti::Wave::Message, [41](#)
- clearError
  - Ghoti::Wave::Server, [80](#)
- ClientParameter
  - hasClientParameters.hpp, [101](#)
- ClientSession
  - Ghoti::Wave::ClientSession, [22](#)
- convertToFile
  - Ghoti::Wave::Blob, [9](#)
- createNewMessage
  - Ghoti::Wave::Parser, [58](#)
  - Ghoti::Wave::RequestParser, [64](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- CRLF
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- dispatchLoop
  - Ghoti::Wave::Client, [15](#)
  - Ghoti::Wave::Server, [80](#)
- domains
  - Ghoti::Wave::Client, [19](#)
- enqueue
  - Ghoti::Wave::ClientSession, [22](#)
- ErrorCode
  - Ghoti::Wave::Server, [79](#)

- FIELD\_LINE
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- FIELD\_NAME
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- FIELD\_VALUE
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- FIELD\_VALUE\_COMMA
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- fieldValueEscape
  - parsing.hpp, [109](#)
- fieldValueQuotesNeeded
  - parsing.hpp, [109](#)
- FIXED
  - Ghoti::Wave::Message, [41](#)
- getAddress
  - Ghoti::Wave::Server, [81](#)
- getAllParameters
  - Ghoti::Wave::Client, [15](#)
  - Ghoti::Wave::ClientSession, [22](#)
  - Ghoti::Wave::HasClientParameters, [28](#)
  - Ghoti::Wave::HasParameters< T >, [32](#)
  - Ghoti::Wave::HasServerParameters, [35](#)
  - Ghoti::Wave::RequestParser, [64](#)
  - Ghoti::Wave::ResponseParser, [73](#)
  - Ghoti::Wave::Server, [81](#)
  - Ghoti::Wave::ServerSession, [89](#)
- getContentLength
  - Ghoti::Wave::Message, [44](#)
- getDomain
  - Ghoti::Wave::Message, [44](#)
- getErrorCode
  - Ghoti::Wave::Server, [81](#)
- getErrorMessage
  - Ghoti::Wave::Server, [81](#)
- getFields
  - Ghoti::Wave::Message, [44](#)
- getFile
  - Ghoti::Wave::Blob, [10](#)
- getId
  - Ghoti::Wave::Message, [44](#)
- getMEMCHUNKSIZELIMIT
  - Ghoti::Wave::Parser, [58](#)
  - Ghoti::Wave::RequestParser, [64](#)
  - Ghoti::Wave::ResponseParser, [73](#)
- getMessage
  - Ghoti::Wave::Message, [44](#)
- getMessageBody
  - Ghoti::Wave::Message, [45](#)
- getMethod
  - Ghoti::Wave::Message, [45](#)
- getParameter
  - Ghoti::Wave::Client, [15](#)
  - Ghoti::Wave::ClientSession, [23](#)
  - Ghoti::Wave::HasClientParameters, [28](#)
  - Ghoti::Wave::HasParameters< T >, [32](#)
  - Ghoti::Wave::HasServerParameters, [36](#)
  - Ghoti::Wave::RequestParser, [65](#)
  - Ghoti::Wave::ResponseParser, [73](#)
  - Ghoti::Wave::Server, [82](#)
  - Ghoti::Wave::ServerSession, [90](#)
- getParameterAny
  - Ghoti::Wave::Client, [16](#)
  - Ghoti::Wave::ClientSession, [23](#)
  - Ghoti::Wave::HasClientParameters, [29](#)
  - Ghoti::Wave::HasParameters< T >, [33](#)
  - Ghoti::Wave::HasServerParameters, [36](#)
  - Ghoti::Wave::RequestParser, [65](#)
  - Ghoti::Wave::ResponseParser, [74](#)
  - Ghoti::Wave::Server, [82](#)
  - Ghoti::Wave::ServerSession, [90](#)
- getParameterDefault
  - Ghoti::Wave::Client, [16](#)
  - Ghoti::Wave::ClientSession, [23, 24](#)
  - Ghoti::Wave::HasClientParameters, [29](#)
  - Ghoti::Wave::HasParameters< T >, [33](#)
  - Ghoti::Wave::HasServerParameters, [36, 37](#)
  - Ghoti::Wave::RequestParser, [66](#)
  - Ghoti::Wave::ResponseParser, [74](#)
  - Ghoti::Wave::Server, [83](#)
  - Ghoti::Wave::ServerSession, [90, 91](#)
- getPort
  - Ghoti::Wave::Message, [45](#)
  - Ghoti::Wave::Server, [83](#)
- getReadySemaphore
  - Ghoti::Wave::Message, [45](#)
- getRenderedHeader1
  - Ghoti::Wave::Message, [46](#)
- getSocketHandle
  - Ghoti::Wave::Server, [84](#)
- getStatusCode
  - Ghoti::Wave::Message, [46](#)
- getTarget
  - Ghoti::Wave::Message, [47](#)
- getText
  - Ghoti::Wave::Blob, [10](#)
- getTransport
  - Ghoti::Wave::Message, [47](#)
- getType
  - Ghoti::Wave::Blob, [10](#)
  - Ghoti::Wave::Message, [47](#)
- getVersion
  - Ghoti::Wave::Message, [47](#)
- Ghoti::Wave::Blob, [7](#)
  - append, [9](#)
  - Blob, [8, 9](#)
  - convertToFile, [9](#)
  - getFile, [10](#)
  - getText, [10](#)

- getType, 10
- length, 10
- operator==, 11
- set, 11, 12
- size, 12
- truncate, 12
- Ghoti::Wave::Client, 13
  - dispatchLoop, 15
  - domains, 19
  - getAllParameters, 15
  - getParameter, 15
  - getParameterAny, 16
  - getParameterDefault, 16
  - isRunning, 17
  - sendRequest, 17
  - setParameter, 17
  - start, 18
  - stop, 18
- Ghoti::Wave::ClientSession, 19
  - ClientSession, 22
  - enqueue, 22
  - getAllParameters, 22
  - getParameter, 23
  - getParameterAny, 23
  - getParameterDefault, 23, 24
  - hasReadDataWaiting, 24
  - hasWriteDataWaiting, 24
  - isFinished, 25
  - MAXBUFFERSIZE, 22
  - messages, 26
  - Parameter, 21
  - read, 25
  - readSequence, 26
  - requestSequence, 26
  - setParameter, 25
  - write, 26
  - writeSequence, 26
- Ghoti::Wave::HasClientParameters, 27
  - getAllParameters, 28
  - getParameter, 28
  - getParameterAny, 29
  - getParameterDefault, 29
  - setParameter, 30
- Ghoti::Wave::HasParameters< T >, 30
  - getAllParameters, 32
  - getParameter, 32
  - getParameterAny, 33
  - getParameterDefault, 33
  - setParameter, 34
- Ghoti::Wave::HasServerParameters, 34
  - getAllParameters, 35
  - getParameter, 36
  - getParameterAny, 36
  - getParameterDefault, 36, 37
  - setParameter, 37
- Ghoti::Wave::Message, 38
  - addFieldValue, 42
  - adoptContents, 42
- CHUNKED, 41
- FIXED, 41
- getContentLength, 44
- getDomain, 44
- getFields, 44
- getId, 44
- getMessage, 44
- getMessageBody, 45
- getMethod, 45
- getPort, 45
- getReadySemaphore, 45
- getRenderedHeader1, 46
- getStatusCode, 46
- getTarget, 47
- getTransport, 47
- getType, 47
- getVersion, 47
- hasError, 48
- headers, 53
- isFinished, 48
- Message, 42
- MULTIPART, 41
- parseIsFinished, 53
- REQUEST, 41
- RESPONSE, 41
- setDomain, 48
- setErrorMessage, 49
- setId, 49
- setMessage, 49
- setMessageBody, 50
- setMethod, 50
- setPort, 51
- setReady, 51
- setStatusCode, 51
- setTarget, 52
- setTransport, 52
- setVersion, 53
- STREAM, 41
- Transport, 41
- Type, 41
- UNDECLARED, 41
- Ghoti::Wave::Parser, 54
  - AFTER\_CRLF, 56
  - AFTER\_FIELD\_NAME, 57
  - AFTER\_FIELD\_VALUE, 57
  - AFTER\_FIELD\_VALUE\_COMMA, 57
  - AFTER\_HEADER\_FIELDS, 57
  - AFTER\_HTTP\_VERSION, 57
  - AFTER\_METHOD, 56
  - AFTER\_REQUEST\_TARGET, 56
  - BEFORE\_FIELD\_VALUE, 57
  - BEGINNING\_OF\_FIELD\_LINE, 56
  - BEGINNING\_OF\_REQUEST, 56
  - BEGINNING\_OF\_REQUEST\_LINE, 56
  - BEGINNING\_OF\_STATUS, 56
  - BEGINNING\_OF\_STATUS\_LINE, 56
  - createNewMessage, 58
  - CRLF, 56

FIELD\_LINE, 56  
FIELD\_NAME, 57  
FIELD\_VALUE, 57  
FIELD\_VALUE\_COMMA, 57  
getMEMCHUNKSIZELIMIT, 58  
HTTP\_VERSION, 57  
LIST\_FIELD\_VALUE, 57  
MESSAGE\_BODY, 56  
MESSAGE\_READ, 57  
MESSAGE\_START, 57  
messageRegister, 59  
messages, 59  
METHOD, 56  
NEW\_HEADER, 56  
Parser, 57  
processChunk, 58  
QUOTED\_FIELD\_VALUE\_CLOSE, 57  
QUOTED\_FIELD\_VALUE\_ESCAPE, 57  
QUOTED\_FIELD\_VALUE\_OPEN, 57  
QUOTED\_FIELD\_VALUE\_PROCESS, 57  
ReadStateMajor, 56  
ReadStateMinor, 56  
REASON\_PHRASE, 57  
registerMessage, 59  
REQUEST, 57  
REQUEST\_TARGET, 56  
RESPONSE, 57  
RESPONSE\_CODE, 57  
SINGLETON\_FIELD\_VALUE, 57  
Type, 57  
UNQUOTED\_FIELD\_VALUE, 57  
Ghoti::Wave::RequestParser, 60  
AFTER\_CRLF, 63  
AFTER\_FIELD\_NAME, 63  
AFTER\_FIELD\_VALUE, 63  
AFTER\_FIELD\_VALUE\_COMMA, 63  
AFTER\_HEADER\_FIELDS, 63  
AFTER\_HTTP\_VERSION, 63  
AFTER\_METHOD, 63  
AFTER\_REQUEST\_TARGET, 63  
BEFORE\_FIELD\_VALUE, 63  
BEGINNING\_OF\_FIELD\_LINE, 63  
BEGINNING\_OF\_REQUEST, 63  
BEGINNING\_OF\_REQUEST\_LINE, 63  
BEGINNING\_OF\_STATUS, 63  
BEGINNING\_OF\_STATUS\_LINE, 63  
createNewMessage, 64  
CRLF, 63  
FIELD\_LINE, 63  
FIELD\_NAME, 63  
FIELD\_VALUE, 63  
FIELD\_VALUE\_COMMA, 63  
getAllParameters, 64  
getMEMCHUNKSIZELIMIT, 64  
getParameter, 65  
getParameterAny, 65  
getParameterDefault, 66  
HTTP\_VERSION, 63  
LIST\_FIELD\_VALUE, 63  
MESSAGE\_BODY, 63  
MESSAGE\_READ, 63  
MESSAGE\_START, 63  
messageRegister, 67  
messages, 68  
METHOD, 63  
NEW\_HEADER, 63  
processChunk, 66  
QUOTED\_FIELD\_VALUE\_CLOSE, 63  
QUOTED\_FIELD\_VALUE\_ESCAPE, 63  
QUOTED\_FIELD\_VALUE\_OPEN, 63  
QUOTED\_FIELD\_VALUE\_PROCESS, 63  
ReadStateMajor, 62  
ReadStateMinor, 63  
REASON\_PHRASE, 63  
registerMessage, 67  
REQUEST, 64  
REQUEST\_TARGET, 63  
RESPONSE, 64  
RESPONSE\_CODE, 63  
setParameter, 67  
SINGLETON\_FIELD\_VALUE, 63  
Type, 64  
UNQUOTED\_FIELD\_VALUE, 63  
Ghoti::Wave::ResponseParser, 68  
AFTER\_CRLF, 71  
AFTER\_FIELD\_NAME, 72  
AFTER\_FIELD\_VALUE, 72  
AFTER\_FIELD\_VALUE\_COMMA, 72  
AFTER\_HEADER\_FIELDS, 72  
AFTER\_HTTP\_VERSION, 72  
AFTER\_METHOD, 71  
AFTER\_REQUEST\_TARGET, 71  
BEFORE\_FIELD\_VALUE, 72  
BEGINNING\_OF\_FIELD\_LINE, 71  
BEGINNING\_OF\_REQUEST, 71  
BEGINNING\_OF\_REQUEST\_LINE, 71  
BEGINNING\_OF\_STATUS, 71  
BEGINNING\_OF\_STATUS\_LINE, 71  
createNewMessage, 72  
CRLF, 71  
FIELD\_LINE, 71  
FIELD\_NAME, 72  
FIELD\_VALUE, 72  
FIELD\_VALUE\_COMMA, 72  
getAllParameters, 73  
getMEMCHUNKSIZELIMIT, 73  
getParameter, 73  
getParameterAny, 74  
getParameterDefault, 74  
HTTP\_VERSION, 72  
LIST\_FIELD\_VALUE, 72  
MESSAGE\_BODY, 71  
MESSAGE\_READ, 72  
MESSAGE\_START, 72  
messageRegister, 76  
messages, 76

- METHOD, 71
- NEW\_HEADER, 71
- processChunk, 75
- QUOTED\_FIELD\_VALUE\_CLOSE, 72
- QUOTED\_FIELD\_VALUE\_ESCAPE, 72
- QUOTED\_FIELD\_VALUE\_OPEN, 72
- QUOTED\_FIELD\_VALUE\_PROCESS, 72
- ReadStateMajor, 71
- ReadStateMinor, 71
- REASON\_PHRASE, 72
- registerMessage, 75
- REQUEST, 72
- REQUEST\_TARGET, 71
- RESPONSE, 72
- RESPONSE\_CODE, 72
- setParameter, 75
- SINGLETON\_FIELD\_VALUE, 72
- Type, 72
- UNQUOTED\_FIELD\_VALUE, 72
- Ghoti::Wave::Server, 77
  - ~Server, 80
  - clearError, 80
  - dispatchLoop, 80
  - ErrorCode, 79
  - getAddress, 81
  - getAllParameters, 81
  - getErrorCode, 81
  - getErrorMessage, 81
  - getParameter, 82
  - getParameterAny, 82
  - getParameterDefault, 83
  - getPort, 83
  - getSocketHandle, 84
  - isRunning, 84
  - NO\_ERROR, 79
  - Server, 79
  - SERVER\_ALREADY\_RUNNING, 79
  - sessions, 86
  - setAddress, 84
  - setParameter, 84
  - setPort, 85
  - start, 86
  - START\_FAILED, 79
  - stop, 86
- Ghoti::Wave::ServerSession, 87
  - getAllParameters, 89
  - getParameter, 90
  - getParameterAny, 90
  - getParameterDefault, 90, 91
  - hasReadDataWaiting, 91
  - hasWriteDataWaiting, 91
  - isFinished, 92
  - messages, 93
  - read, 92
  - ServerSession, 89
  - setParameter, 92
  - write, 93
- hasClientParameters.hpp
  - ClientParameter, 101
  - MAXBUFFERSIZE, 101
  - MEMCHUNKSIZELIMIT, 101
- hasError
  - Ghoti::Wave::Message, 48
- hasReadDataWaiting
  - Ghoti::Wave::ClientSession, 24
  - Ghoti::Wave::ServerSession, 91
- hasServerParameters.hpp
  - MAXBUFFERSIZE, 103
  - MEMCHUNKSIZELIMIT, 103
  - ServerParameter, 103
- hasWriteDataWaiting
  - Ghoti::Wave::ClientSession, 24
  - Ghoti::Wave::ServerSession, 91
- headers
  - Ghoti::Wave::Message, 53
- HTTP\_VERSION
  - Ghoti::Wave::Parser, 57
  - Ghoti::Wave::RequestParser, 63
  - Ghoti::Wave::ResponseParser, 72
- include/wave.hpp, 95
- include/wave/blob.hpp, 96
- include/wave/client.hpp, 98
- include/wave/clientSession.hpp, 99
- include/wave/hasClientParameters.hpp, 100
- include/wave/hasParameters.hpp, 101
- include/wave/hasServerParameters.hpp, 102
- include/wave/macros.hpp, 104
- include/wave/message.hpp, 104
- include/wave/parser.hpp, 106
- include/wave/parsing.hpp, 107
- include/wave/response.hpp, 116
- include/wave/server.hpp, 117
- include/wave/serverSession.hpp, 118
- isCRLFChar
  - parsing.cpp, 128
  - parsing.hpp, 110
- isFieldContentChar
  - parsing.cpp, 128
  - parsing.hpp, 111
- isFieldNameChar
  - parsing.cpp, 129
  - parsing.hpp, 111
- isFinished
  - Ghoti::Wave::ClientSession, 25
  - Ghoti::Wave::Message, 48
  - Ghoti::Wave::ServerSession, 92
- isListField
  - parsing.hpp, 112
- isObsoleteTextChar
  - parsing.cpp, 129
  - parsing.hpp, 113
- isQuotedChar
  - parsing.cpp, 130
  - parsing.hpp, 114
- isRunning
  - Ghoti::Wave::Client, 17

- Ghoti::Wave::Server, [84](#)
- isTokenChar
  - parsing.cpp, [131](#)
  - parsing.hpp, [114](#)
- isVisibleChar
  - parsing.cpp, [131](#)
  - parsing.hpp, [115](#)
- isWhitespaceChar
  - parsing.cpp, [132](#)
  - parsing.hpp, [116](#)
- length
  - Ghoti::Wave::Blob, [10](#)
- LIST\_FIELD\_VALUE
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- MAXBUFFERSIZE
  - Ghoti::Wave::ClientSession, [22](#)
  - hasClientParameters.hpp, [101](#)
  - hasServerParameters.hpp, [103](#)
- MEMCHUNKSIZELIMIT
  - hasClientParameters.hpp, [101](#)
  - hasServerParameters.hpp, [103](#)
- Message
  - Ghoti::Wave::Message, [42](#)
- message.hpp
  - operator<<, [105](#)
- MESSAGE\_BODY
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- MESSAGE\_READ
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- MESSAGE\_START
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- messageRegister
  - Ghoti::Wave::Parser, [59](#)
  - Ghoti::Wave::RequestParser, [67](#)
  - Ghoti::Wave::ResponseParser, [76](#)
- messages
  - Ghoti::Wave::ClientSession, [26](#)
  - Ghoti::Wave::Parser, [59](#)
  - Ghoti::Wave::RequestParser, [68](#)
  - Ghoti::Wave::ResponseParser, [76](#)
  - Ghoti::Wave::ServerSession, [93](#)
- METHOD
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- MULTIPART
  - Ghoti::Wave::Message, [41](#)
- NEW\_HEADER
  - Ghoti::Wave::Parser, [56](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [71](#)
- NO\_ERROR
  - Ghoti::Wave::Server, [79](#)
- operator<<
  - blob.hpp, [97](#)
  - message.hpp, [105](#)
- operator==
  - Ghoti::Wave::Blob, [11](#)
- Parameter
  - Ghoti::Wave::ClientSession, [21](#)
- Parser
  - Ghoti::Wave::Parser, [57](#)
- parser.cpp
  - READ\_CRLF\_OPTIONAL, [124](#)
  - READ\_CRLF\_REQUIRED, [125](#)
  - READ\_WHITESPACE\_OPTIONAL, [125](#)
  - READ\_WHITESPACE\_REQUIRED, [125](#)
  - SET\_MAJOR\_STATE, [126](#)
  - SET\_MINOR\_STATE, [126](#)
  - SET\_NEW\_HEADER, [126](#)
  - START\_NEW\_INPUT, [126](#)
- parsing.cpp
  - isCRLFChar, [128](#)
  - isFieldContentChar, [128](#)
  - isFieldNameChar, [129](#)
  - isObsoleteTextChar, [129](#)
  - isQuotedChar, [130](#)
  - isTokenChar, [131](#)
  - isVisibleChar, [131](#)
  - isWhitespaceChar, [132](#)
- parsing.hpp
  - fieldValueEscape, [109](#)
  - fieldValueQuotesNeeded, [109](#)
  - isCRLFChar, [110](#)
  - isFieldContentChar, [111](#)
  - isFieldNameChar, [111](#)
  - isListField, [112](#)
  - isObsoleteTextChar, [113](#)
  - isQuotedChar, [114](#)
  - isTokenChar, [114](#)
  - isVisibleChar, [115](#)
  - isWhitespaceChar, [116](#)
- parsingIsFinished
  - Ghoti::Wave::Message, [53](#)
- processChunk
  - Ghoti::Wave::Parser, [58](#)
  - Ghoti::Wave::RequestParser, [66](#)
  - Ghoti::Wave::ResponseParser, [75](#)
- QUOTED\_FIELD\_VALUE\_CLOSE
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- QUOTED\_FIELD\_VALUE\_ESCAPE
  - Ghoti::Wave::Parser, [57](#)



- Ghoti::Wave::RequestParser, 63
- Ghoti::Wave::ResponseParser, 72
- QUOTED\_FIELD\_VALUE\_OPEN
  - Ghoti::Wave::Parser, 57
  - Ghoti::Wave::RequestParser, 63
  - Ghoti::Wave::ResponseParser, 72
- QUOTED\_FIELD\_VALUE\_PROCESS
  - Ghoti::Wave::Parser, 57
  - Ghoti::Wave::RequestParser, 63
  - Ghoti::Wave::ResponseParser, 72
- read
  - Ghoti::Wave::ClientSession, 25
  - Ghoti::Wave::ServerSession, 92
- READ\_CRLF\_OPTIONAL
  - parser.cpp, 124
- READ\_CRLF\_REQUIRED
  - parser.cpp, 125
- READ\_WHITESPACE\_OPTIONAL
  - parser.cpp, 125
- READ\_WHITESPACE\_REQUIRED
  - parser.cpp, 125
- readSequence
  - Ghoti::Wave::ClientSession, 26
- ReadStateMajor
  - Ghoti::Wave::Parser, 56
  - Ghoti::Wave::RequestParser, 62
  - Ghoti::Wave::ResponseParser, 71
- ReadStateMinor
  - Ghoti::Wave::Parser, 56
  - Ghoti::Wave::RequestParser, 63
  - Ghoti::Wave::ResponseParser, 71
- REASON\_PHRASE
  - Ghoti::Wave::Parser, 57
  - Ghoti::Wave::RequestParser, 63
  - Ghoti::Wave::ResponseParser, 72
- registerMessage
  - Ghoti::Wave::Parser, 59
  - Ghoti::Wave::RequestParser, 67
  - Ghoti::Wave::ResponseParser, 75
- REQUEST
  - Ghoti::Wave::Message, 41
  - Ghoti::Wave::Parser, 57
  - Ghoti::Wave::RequestParser, 64
  - Ghoti::Wave::ResponseParser, 72
- REQUEST\_TARGET
  - Ghoti::Wave::Parser, 56
  - Ghoti::Wave::RequestParser, 63
  - Ghoti::Wave::ResponseParser, 71
- requestSequence
  - Ghoti::Wave::ClientSession, 26
- RESPONSE
  - Ghoti::Wave::Message, 41
  - Ghoti::Wave::Parser, 57
  - Ghoti::Wave::RequestParser, 64
  - Ghoti::Wave::ResponseParser, 72
- RESPONSE\_CODE
  - Ghoti::Wave::Parser, 57
  - Ghoti::Wave::RequestParser, 63
  - Ghoti::Wave::ResponseParser, 72
- Ghoti::Wave::ResponseParser, 72
- sendRequest
  - Ghoti::Wave::Client, 17
- Server
  - Ghoti::Wave::Server, 79
- SERVER\_ALREADY\_RUNNING
  - Ghoti::Wave::Server, 79
- ServerParameter
  - hasServerParameters.hpp, 103
- ServerSession
  - Ghoti::Wave::ServerSession, 89
- sessions
  - Ghoti::Wave::Server, 86
- set
  - Ghoti::Wave::Blob, 11, 12
- SET\_MAJOR\_STATE
  - parser.cpp, 126
- SET\_MINOR\_STATE
  - parser.cpp, 126
- SET\_NEW\_HEADER
  - parser.cpp, 126
- setAddress
  - Ghoti::Wave::Server, 84
- setDomain
  - Ghoti::Wave::Message, 48
- setErrorMessage
  - Ghoti::Wave::Message, 49
- setId
  - Ghoti::Wave::Message, 49
- setMessage
  - Ghoti::Wave::Message, 49
- setMessageBody
  - Ghoti::Wave::Message, 50
- setMethod
  - Ghoti::Wave::Message, 50
- setParameter
  - Ghoti::Wave::Client, 17
  - Ghoti::Wave::ClientSession, 25
  - Ghoti::Wave::HasClientParameters, 30
  - Ghoti::Wave::HasParameters< T >, 34
  - Ghoti::Wave::HasServerParameters, 37
  - Ghoti::Wave::RequestParser, 67
  - Ghoti::Wave::ResponseParser, 75
  - Ghoti::Wave::Server, 84
  - Ghoti::Wave::ServerSession, 92
- setPort
  - Ghoti::Wave::Message, 51
  - Ghoti::Wave::Server, 85
- setReady
  - Ghoti::Wave::Message, 51
- setStatusCode
  - Ghoti::Wave::Message, 51
- setTarget
  - Ghoti::Wave::Message, 52
- setTransport
  - Ghoti::Wave::Message, 52
- setVersion
  - Ghoti::Wave::Message, 53

- SINGLETON\_FIELD\_VALUE
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- size
  - Ghoti::Wave::Blob, [12](#)
- src/blob.cpp, [119](#)
- src/client.cpp, [120](#)
- src/clientSession.cpp, [121](#)
- src/hasClientParameters.cpp, [121](#)
- src/hasServerParameters.cpp, [122](#)
- src/message.cpp, [123](#)
- src/parser.cpp, [124](#)
- src/parsing.cpp, [127](#)
- src/response.cpp, [133](#)
- src/server.cpp, [133](#)
- src/serverSession.cpp, [134](#)
- start
  - Ghoti::Wave::Client, [18](#)
  - Ghoti::Wave::Server, [86](#)
- START\_FAILED
  - Ghoti::Wave::Server, [79](#)
- START\_NEW\_INPUT
  - parser.cpp, [126](#)
- stop
  - Ghoti::Wave::Client, [18](#)
  - Ghoti::Wave::Server, [86](#)
- STREAM
  - Ghoti::Wave::Message, [41](#)
- test/test-hasParameters.cpp, [135](#)
- test/test.cpp, [135](#)
- Transport
  - Ghoti::Wave::Message, [41](#)
- truncate
  - Ghoti::Wave::Blob, [12](#)
- Type
  - Ghoti::Wave::Message, [41](#)
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [64](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- UNDECLARED
  - Ghoti::Wave::Message, [41](#)
- UNQUOTED\_FIELD\_VALUE
  - Ghoti::Wave::Parser, [57](#)
  - Ghoti::Wave::RequestParser, [63](#)
  - Ghoti::Wave::ResponseParser, [72](#)
- write
  - Ghoti::Wave::ClientSession, [26](#)
  - Ghoti::Wave::ServerSession, [93](#)
- writeSequence
  - Ghoti::Wave::ClientSession, [26](#)