

c++ library collection

Generated by Doxygen 1.8.13

Contents

1	CppLibs	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	devfix Namespace Reference	11
6.2	devfix::base Namespace Reference	11
6.2.1	Typedef Documentation	11
6.2.1.1	sp	11
6.2.1.2	up	12
6.3	devfix::base::error Namespace Reference	12
6.3.1	Detailed Description	12
6.4	devfix::base::io Namespace Reference	12
6.4.1	Detailed Description	13
6.4.2	Typedef Documentation	13
6.4.2.1	available_t	13
6.4.2.2	close_t	13
6.4.2.3	flush_t	13
6.4.2.4	is_closed_t	13
6.4.2.5	read_t	13
6.4.2.6	skip_t	14
6.4.2.7	write_t	14
6.4.3	Variable Documentation	14
6.4.3.1	DEFAULT_CLOSE	14
6.4.3.2	DEFAULT_IS_CLOSED	14
6.5	devfix::net Namespace Reference	14

7 Class Documentation	15
7.1 devfix::base::error::baseexception Struct Reference	15
7.1.1 Detailed Description	16
7.1.2 Constructor & Destructor Documentation	16
7.1.2.1 baseexception() [1/2]	16
7.1.2.2 baseexception() [2/2]	16
7.1.2.3 ~baseexception()	16
7.1.3 Member Function Documentation	17
7.1.3.1 get_errno()	17
7.1.3.2 what()	17
7.1.4 Member Data Documentation	17
7.1.4.1 err_	17
7.1.4.2 what_arg_	17
7.2 devfix::net::inetaddress Struct Reference	18
7.2.1 Member Typedef Documentation	18
7.2.1.1 address_t	18
7.2.1.2 port_t	18
7.2.2 Member Enumeration Documentation	18
7.2.2.1 family_t	18
7.2.3 Constructor & Destructor Documentation	19
7.2.3.1 inetaddress() [1/2]	19
7.2.3.2 inetaddress() [2/2]	19
7.2.4 Member Function Documentation	19
7.2.4.1 get_address()	19
7.2.4.2 get_family()	20
7.2.4.3 get_host()	20
7.2.4.4 get_port()	20
7.3 devfix::base::io::inputstream Struct Reference	21
7.3.1 Detailed Description	22
7.3.2 Constructor & Destructor Documentation	22

7.3.2.1	<code>~inputstream()</code>	22
7.3.3	Member Function Documentation	22
7.3.3.1	<code>available()</code>	22
7.3.3.2	<code>close()</code>	23
7.3.3.3	<code>is_closed()</code>	23
7.3.3.4	<code>read()</code>	23
7.3.3.5	<code>skip()</code>	24
7.4	<code>devfix::base::error::interruptedexception</code> Struct Reference	24
7.4.1	Detailed Description	25
7.4.2	Constructor & Destructor Documentation	26
7.4.2.1	<code>interruptedexception()</code> [1/2]	26
7.4.2.2	<code>interruptedexception()</code> [2/2]	26
7.5	<code>devfix::base::error::ioexception</code> Struct Reference	26
7.5.1	Detailed Description	27
7.5.2	Constructor & Destructor Documentation	27
7.5.2.1	<code>ioexception()</code> [1/2]	28
7.5.2.2	<code>ioexception()</code> [2/2]	28
7.6	<code>devfix::net::netbuilder</code> Struct Reference	28
7.6.1	Constructor & Destructor Documentation	29
7.6.1.1	<code>netbuilder()</code>	29
7.6.2	Member Function Documentation	29
7.6.2.1	<code>create_serversocket()</code>	29
7.6.2.2	<code>create_socket()</code>	29
7.7	<code>devfix::base::io::outputstream</code> Struct Reference	30
7.7.1	Detailed Description	31
7.7.2	Constructor & Destructor Documentation	31
7.7.2.1	<code>~outputstream()</code>	31
7.7.3	Member Function Documentation	31
7.7.3.1	<code>close()</code>	31
7.7.3.2	<code>flush()</code>	32

7.7.3.3	<code>is_closed()</code>	32
7.7.3.4	<code>write()</code>	32
7.8	<code>devfix::net::serversocket</code> Struct Reference	33
7.8.1	Constructor & Destructor Documentation	33
7.8.1.1	<code>~serversocket()</code>	33
7.8.2	Member Function Documentation	33
7.8.2.1	<code>accept()</code>	33
7.8.2.2	<code>close()</code>	33
7.8.2.3	<code>get_accept_timeout()</code>	33
7.8.2.4	<code>get_address()</code>	34
7.8.2.5	<code>get_reuse_address()</code>	34
7.8.2.6	<code>is_closed()</code>	34
7.8.2.7	<code>set_accept_timeout()</code>	34
7.9	<code>devfix::base::io::sink</code> Struct Reference	34
7.9.1	Constructor & Destructor Documentation	35
7.9.1.1	<code>sink()</code>	35
7.9.2	Member Function Documentation	36
7.9.2.1	<code>close()</code>	36
7.9.2.2	<code>flush()</code>	36
7.9.2.3	<code>is_closed()</code>	36
7.9.2.4	<code>write()</code>	36
7.10	<code>devfix::net::socket</code> Struct Reference	37
7.10.1	Member Typedef Documentation	37
7.10.1.1	<code>timeout_t</code>	37
7.10.2	Constructor & Destructor Documentation	38
7.10.2.1	<code>~socket()</code>	38
7.10.3	Member Function Documentation	38
7.10.3.1	<code>get_inputstream()</code>	38
7.10.3.2	<code>get_local_address()</code>	38
7.10.3.3	<code>get_outputstream()</code>	38

7.10.3.4	get_remote_address()	38
7.10.3.5	get_timeout()	38
7.10.3.6	interrupted()	39
7.10.3.7	set_interrupted()	39
7.10.3.8	set_timeout()	39
7.10.4	Member Data Documentation	39
7.10.4.1	DEFAULT_READ_BLOCKING_TIME	39
7.10.4.2	DEFAULT_TIMEOUT	39
7.11	devfix::net::socketexception Struct Reference	40
7.11.1	Detailed Description	40
7.11.2	Constructor & Destructor Documentation	41
7.11.2.1	socketexception() [1/2]	41
7.11.2.2	socketexception() [2/2]	41
7.12	devfix::base::io::source Struct Reference	41
7.12.1	Constructor & Destructor Documentation	42
7.12.1.1	source()	43
7.12.2	Member Function Documentation	43
7.12.2.1	available()	43
7.12.2.2	close()	43
7.12.2.3	is_closed()	44
7.12.2.4	read()	44
7.12.2.5	skip()	44
7.13	devfix::base::error::timeoutexception Struct Reference	45
7.13.1	Detailed Description	46
7.13.2	Constructor & Destructor Documentation	46
7.13.2.1	timeoutexception() [1/2]	46
7.13.2.2	timeoutexception() [2/2]	46

8	File Documentation	49
8.1	cmake-build-debug/CMakeFiles/3.15.3/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference	49
8.1.1	Macro Definition Documentation	49
8.1.1.1	ARCHITECTURE_ID	49
8.1.1.2	COMPILER_ID	50
8.1.1.3	CXX_STD	50
8.1.1.4	DEC	50
8.1.1.5	HEX	50
8.1.1.6	PLATFORM_ID	50
8.1.1.7	STRINGIFY	51
8.1.1.8	STRINGIFY_HELPER	51
8.1.2	Function Documentation	51
8.1.2.1	main()	51
8.1.3	Variable Documentation	51
8.1.3.1	info_arch	51
8.1.3.2	info_compiler	51
8.1.3.3	info_language_dialect_default	52
8.1.3.4	info_platform	52
8.2	devfix/base/error/baseexception.h File Reference	52
8.2.1	Macro Definition Documentation	53
8.2.1.1	exception_guard	53
8.2.1.2	exception_guard_m	53
8.3	devfix/base/error/interruptedexception.h File Reference	54
8.4	devfix/base/error/ioexception.h File Reference	54
8.5	devfix/base/error/namespace.h File Reference	55
8.6	devfix/base/io/namespace.h File Reference	55
8.7	devfix/base/error/timeoutexception.h File Reference	56
8.8	devfix/base/io/inputstream.h File Reference	56
8.9	devfix/base/io/iotypes.h File Reference	57
8.10	devfix/base/io/outputstream.h File Reference	59

8.11	devfix/base/io/sink.cpp File Reference	60
8.12	devfix/base/io/sink.h File Reference	60
8.13	devfix/base/io/source.cpp File Reference	62
8.14	devfix/base/io/source.h File Reference	62
8.15	devfix/base/memory.h File Reference	64
8.16	devfix/base/platform.h File Reference	65
8.16.1	Macro Definition Documentation	65
8.16.1.1	__FILENAME__	65
8.16.1.2	PLATFORM_LINUX	65
8.16.1.3	PLATFORM_UNSUPPORTED	65
8.16.1.4	SOURCE_LINE	66
8.17	devfix/net/inetaddress.cpp File Reference	66
8.17.1	Variable Documentation	66
8.17.1.1	PLATFORM_UNSUPPORTED	66
8.18	devfix/net/inetaddress.h File Reference	67
8.19	devfix/net/lx/lx_serversocket.cpp File Reference	68
8.20	devfix/net/lx/lx_serversocket.h File Reference	68
8.21	devfix/net/lx/lx_socket.cpp File Reference	68
8.22	devfix/net/lx/lx_socket.h File Reference	69
8.23	devfix/net/netbuilder.cpp File Reference	69
8.24	devfix/net/netbuilder.h File Reference	70
8.24.1	Variable Documentation	71
8.24.1.1	PLATFORM_UNSUPPORTED	71
8.25	devfix/net/serversocket.h File Reference	72
8.26	devfix/net/socket.h File Reference	72
8.27	devfix/net/socketexception.h File Reference	73
8.28	devfix/net/test/test_inetaddress.cpp File Reference	74
8.28.1	Function Documentation	75
8.28.1.1	TEST()	75
8.29	devfix/net/test/test_socket.cpp File Reference	75
8.29.1	Function Documentation	76
8.29.1.1	TEST() [1/2]	76
8.29.1.2	TEST() [2/2]	76
8.29.2	Variable Documentation	76
8.29.2.1	TEST_ARRAY	76
8.29.2.2	TEST_DOUBLE	76
8.29.2.3	TEST_FLOAT	76
8.29.2.4	TEST_LONG	76
8.29.2.5	TEST_PORT	77
8.30	README.md File Reference	77
8.31	testrunner.cpp File Reference	77
8.31.1	Function Documentation	77
8.31.1.1	main()	77

Chapter 1

CppLibs

TODO

- input stream and output stream separate closable (?)

Build

run `./configure make -C build`

Documentation

available: [here](#)

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

devfix	11
devfix::base	11
devfix::base::error	
Namespace for general errors like timeouts or io failures	12
devfix::base::io	
Namespace for io tool, for instance streams	12
devfix::net	14

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

exception	
devfix::base::error::baseexception	15
devfix::base::error::interruptedexception	24
devfix::base::error::ioexception	26
devfix::base::error::timeoutexception	45
devfix::net::socketexception	40
devfix::net::inetaddress	18
devfix::base::io::inputstream	21
devfix::base::io::source	41
devfix::net::netbuilder	28
devfix::base::io::outputstream	30
devfix::base::io::sink	34
devfix::net::serversocket	33
devfix::net::socket	37

Chapter 4

Class Index

4.1 Class List

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

devfix::base::error::baseexception	
Abstract error base class	15
devfix::net::inetaddress	18
devfix::base::io::inputstream	
Superclass of all classes representing an input stream of bytes	21
devfix::base::error::interruptedexception	
Thrown when an operation is interrupted, either before or during the activity	24
devfix::base::error::ioexception	
Signals that an I/O error of some sort has occurred	26
devfix::net::netbuilder	28
devfix::base::io::outputstream	
Superclass of all classes representing an output stream of bytes	30
devfix::net::serversocket	33
devfix::base::io::sink	34
devfix::net::socket	37
devfix::net::socketexception	
Thrown to indicate that there is an error creating or accessing a Socket	40
devfix::base::io::source	41
devfix::base::error::timeoutexception	
Exception thrown when a blocking operation times out	45

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

testrunner.cpp	77
cmake-build-debug/CMakeFiles/3.15.3/CompilerIdCXX/CMakeCXXCompilerId.cpp	49
devfix/base/memory.h	64
devfix/base/platform.h	65
devfix/base/error/baseexception.h	52
devfix/base/error/interruptedexception.h	54
devfix/base/error/ioexception.h	54
devfix/base/error/namespace.h	55
devfix/base/error/timeoutexception.h	56
devfix/base/io/inputstream.h	56
devfix/base/io/iotypes.h	57
devfix/base/io/namespace.h	55
devfix/base/io/outputstream.h	59
devfix/base/io/sink.cpp	60
devfix/base/io/sink.h	60
devfix/base/io/source.cpp	62
devfix/base/io/source.h	62
devfix/net/inetaddress.cpp	66
devfix/net/inetaddress.h	67
devfix/net/netbuilder.cpp	69
devfix/net/netbuilder.h	70
devfix/net/serversocket.h	72
devfix/net/socket.h	72
devfix/net/socketexception.h	73
devfix/net/lnx/lnx_serversocket.cpp	68
devfix/net/lnx/lnx_serversocket.h	68
devfix/net/lnx/lnx_socket.cpp	68
devfix/net/lnx/lnx_socket.h	69
devfix/net/test/test_inetaddress.cpp	74
devfix/net/test/test_socket.cpp	75

Chapter 6

Namespace Documentation

6.1 devfix Namespace Reference

Namespaces

- [base](#)
- [net](#)

6.2 devfix::base Namespace Reference

Namespaces

- [error](#)
Namespace for general errors like timeouts or io failures.
- [io](#)
Namespace for io tool, for instance streams.

Typedefs

- `template<class T >`
using [up](#) = `std::unique_ptr< T >`
- `template<class T >`
using [sp](#) = `std::shared_ptr< T >`

6.2.1 Typedef Documentation

6.2.1.1 sp

```
template<class T >  
using devfix::base::sp = typedef std::shared_ptr<T>
```

6.2.1.2 up

```
template<class T >
using devfix::base::up = typedef std::unique_ptr<T>
```

6.3 devfix::base::error Namespace Reference

Namespace for general errors like timeouts or io failures.

Classes

- struct [baseexception](#)
Abstract error base class.
- struct [interruptedexception](#)
Thrown when an operation is interrupted, either before or during the activity.
- struct [ioexception](#)
Signals that an I/O error of some sort has occurred.
- struct [timeoutexception](#)
Exception thrown when a blocking operation times out.

6.3.1 Detailed Description

Namespace for general errors like timeouts or io failures.

More specific exceptions are in the namespace of their corresponding functionality.

6.4 devfix::base::io Namespace Reference

Namespace for io tool, for instance streams.

Classes

- struct [inputstream](#)
Superclass of all classes representing an input stream of bytes.
- struct [outputstream](#)
Superclass of all classes representing an output stream of bytes.
- struct [sink](#)
- struct [source](#)

Typedefs

- typedef std::function< void()> [close_t](#)
- typedef std::function< bool()> [is_closed_t](#)
- typedef std::function< void(void *, std::size_t)> [read_t](#)
- typedef std::function< void(std::size_t)> [skip_t](#)
- typedef std::function< std::size_t()> [available_t](#)
- typedef std::function< void(const void *, std::size_t)> [write_t](#)
- typedef std::function< void()> [flush_t](#)

Variables

- const `close_t` `DEFAULT_CLOSE` = []() {}
- const `is_closed_t` `DEFAULT_IS_CLOSED` = []() { return false; }

6.4.1 Detailed Description

Namespace for io tool, for instance streams.

6.4.2 Typedef Documentation

6.4.2.1 `available_t`

```
typedef std::function<std::size_t()> devfix::base::io::available_t
```

6.4.2.2 `close_t`

```
typedef std::function<void()> devfix::base::io::close_t
```

6.4.2.3 `flush_t`

```
typedef std::function<void()> devfix::base::io::flush_t
```

6.4.2.4 `is_closed_t`

```
typedef std::function<bool()> devfix::base::io::is_closed_t
```

6.4.2.5 `read_t`

```
typedef std::function<void(void *, std::size_t)> devfix::base::io::read_t
```

6.4.2.6 skip_t

```
typedef std::function<void(std::size_t)> devfix::base::io::skip_t
```

6.4.2.7 write_t

```
typedef std::function<void(const void *, std::size_t)> devfix::base::io::write_t
```

6.4.3 Variable Documentation

6.4.3.1 DEFAULT_CLOSE

```
const close_t devfix::base::io::DEFAULT_CLOSE = []() {}
```

6.4.3.2 DEFAULT_IS_CLOSED

```
const is_closed_t devfix::base::io::DEFAULT_IS_CLOSED = []() { return false; }
```

6.5 devfix::net Namespace Reference

Classes

- struct [inetaddress](#)
- struct [netbuilder](#)
- struct [serversocket](#)
- struct [socket](#)
- struct [socketexception](#)

Thrown to indicate that there is an error creating or accessing a Socket.

Chapter 7

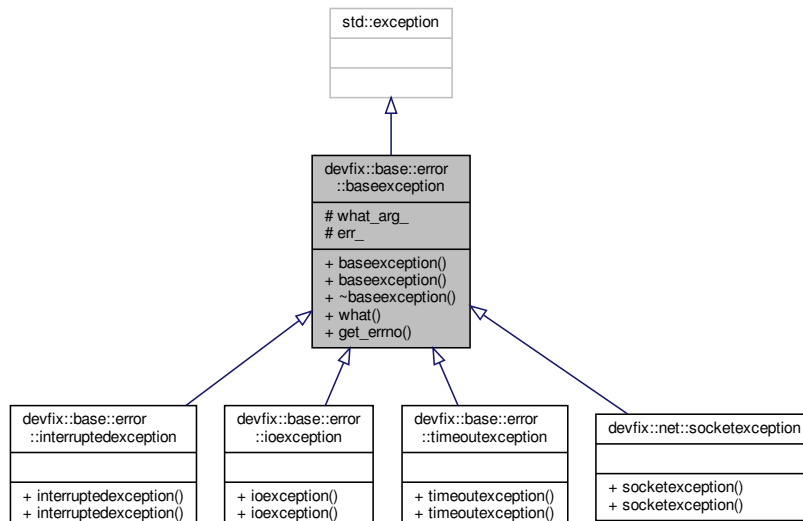
Class Documentation

7.1 devfix::base::error::baseexception Struct Reference

Abstract error base class.

```
#include <baseexception.h>
```

Inheritance diagram for devfix::base::error::baseexception:



Public Member Functions

- `baseexception()` = delete
- `baseexception(std::string what_arg, int err=-1)`
- `~baseexception()` override=default
- `const char * what()` const noexcept final
- `int get_errno()` const noexcept

Protected Attributes

- `std::string` [what_arg_](#)
failure description
- `int` [err_](#)

7.1.1 Detailed Description

Abstract error base class.

This class is the parent of more specific exceptions and cannot be thrown directly.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 `baseexception()` [1/2]

```
devfix::base::error::baseexception::baseexception ( ) [delete]
```

Delete simple constructor, always enforce a failure description.

7.1.2.2 `baseexception()` [2/2]

```
devfix::base::error::baseexception::baseexception (
    std::string what_arg,
    int err = -1 ) [inline], [explicit]
```

Constructs the error object with `what_arg` as explanatory `std::string` that can be accessed through [what\(\)](#).

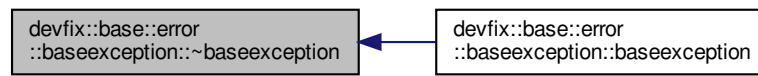
Parameters

<i>what_arg</i>	failure description
-----------------	---------------------

7.1.2.3 `~baseexception()`

```
devfix::base::error::baseexception::~~baseexception ( ) [override], [default]
```

Virtual constructor to make class abstract. Here is the caller graph for this function:



7.1.3 Member Function Documentation

7.1.3.1 get_errno()

```
int devfix::base::error::baseexception::get_errno ( ) const [inline], [noexcept]
```

7.1.3.2 what()

```
const char* devfix::base::error::baseexception::what ( ) const [inline], [final], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Returns

explanatory string

7.1.4 Member Data Documentation

7.1.4.1 err_

```
int devfix::base::error::baseexception::err_ [protected]
```

7.1.4.2 what_arg_

```
std::string devfix::base::error::baseexception::what_arg_ [protected]
```

failure description

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

- devfix/base/error/[baseexception.h](#)

7.2 devfix::net::inetaddress Struct Reference

```
#include <inetaddress.h>
```

Public Types

- enum [family_t](#) : char { [family_t::UNSUPPORTED](#) = 0, [family_t::IPV4](#) = 1 }
- typedef std::uint32_t [address_t](#)
- typedef std::uint16_t [port_t](#)

Public Member Functions

- [inetaddress](#) ()=default
- [inetaddress](#) (const std::string &host, [port_t](#) port, [family_t](#) family=[family_t::IPV4](#))
- std::string [get_host](#) () const noexcept
- [address_t](#) [get_address](#) () const
- [port_t](#) [get_port](#) () const
- [family_t](#) [get_family](#) () const

7.2.1 Member Typedef Documentation

7.2.1.1 address_t

```
typedef std::uint32_t devfix::net::inetaddress::address_t
```

7.2.1.2 port_t

```
typedef std::uint16_t devfix::net::inetaddress::port_t
```

7.2.2 Member Enumeration Documentation

7.2.2.1 family_t

```
enum devfix::net::inetaddress::family_t : char [strong]
```

Enumerator

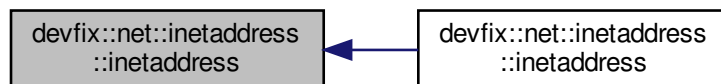
UNSUPPORTED	
IPV4	

7.2.3 Constructor & Destructor Documentation

7.2.3.1 inetaddress() [1/2]

```
devfix::net::inetaddress::inetaddress ( ) [default]
```

Here is the caller graph for this function:



7.2.3.2 inetaddress() [2/2]

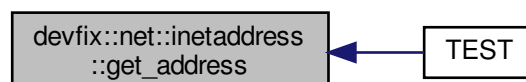
```
devfix::net::inetaddress::inetaddress (
    const std::string & host,
    port_t port,
    family_t family = family_t::IPV4 )
```

7.2.4 Member Function Documentation

7.2.4.1 get_address()

```
inetaddress::address_t devfix::net::inetaddress::get_address ( ) const
```

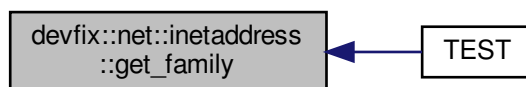
Here is the caller graph for this function:



7.2.4.2 get_family()

```
inetaddress::family_t devfix::net::inetaddress::get_family ( ) const
```

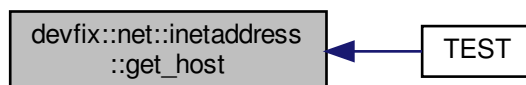
Here is the caller graph for this function:



7.2.4.3 get_host()

```
std::string devfix::net::inetaddress::get_host ( ) const [noexcept]
```

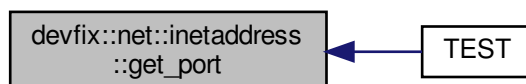
Here is the caller graph for this function:



7.2.4.4 get_port()

```
inetaddress::port_t devfix::net::inetaddress::get_port ( ) const
```

Here is the caller graph for this function:



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

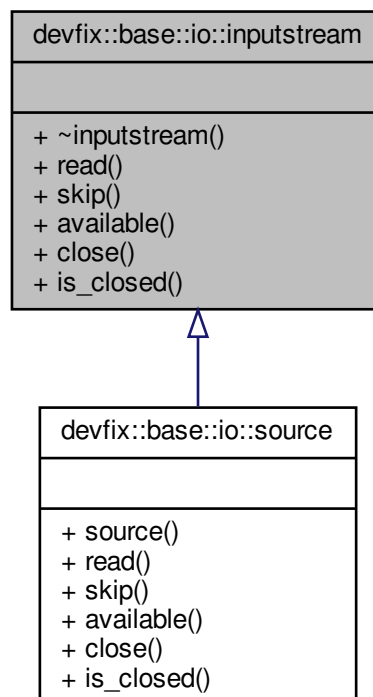
- [devfix/net/inetaddress.h](#)
- [devfix/net/inetaddress.cpp](#)

7.3 devfix::base::io::inputstream Struct Reference

Superclass of all classes representing an input stream of bytes.

```
#include <inputstream.h>
```

Inheritance diagram for devfix::base::io::inputstream:



Public Member Functions

- virtual `~inputstream()`=default
Default virtual destructor.
- virtual void `read` (void *buf, std::size_t len)=0
Reads bytes from the input stream and stores them into the buffer.
- virtual void `skip` (std::size_t n)=0
Skips over and discards n bytes of data from this input stream.
- virtual std::size_t `available` ()=0
Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.
- virtual void `close` ()=0
Closes this input stream and releases any system resources associated with the stream.
- virtual bool `is_closed` ()=0
Returns if the inputstream is closed or available for further calls of input operations.

7.3.1 Detailed Description

Superclass of all classes representing an input stream of bytes.

Applications that need to define a subclass of `InputStream` must always provide a method that returns the next byte of input.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 `~inputstream()`

```
virtual devfix::base::io::inputstream::~~inputstream ( ) [virtual], [default]
```

Default virtual destructor.

Needed for correct deletion of instances of a derived classes through a pointer to base class.

7.3.3 Member Function Documentation

7.3.3.1 `available()`

```
virtual std::size_t devfix::base::io::inputstream::available ( ) [pure virtual]
```

Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.

A single read or skip of this many bytes will not block, but may read or skip fewer bytes.

Note that while some implementations of *inputstream* will return the total number of bytes in the stream, many will not. It is never correct to use the return value of this method to allocate a buffer intended to hold all data in this stream.

A subclass' implementation of this method may choose to throw an `IOException` if this input stream has been closed by invoking the [close\(\)](#) method.

This method should be overridden by subclasses.

Returns

an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking or 0 when it reaches the end of the input stream.

Implemented in [devfix::base::io::source](#).

7.3.3.2 close()

```
virtual void devfix::base::io::inputstream::close ( ) [pure virtual]
```

Closes this input stream and releases any system resources associated with the stream.

A closed stream cannot perform input operations and cannot be reopened.

Implemented in [devfix::base::io::source](#).

7.3.3.3 is_closed()

```
virtual bool devfix::base::io::inputstream::is_closed ( ) [pure virtual]
```

Returns if the *inputstream* is closed or available for further calls of input operations.

Returns

true if the *inputstream* got previously closed.

Implemented in [devfix::base::io::source](#).

7.3.3.4 read()

```
virtual void devfix::base::io::inputstream::read (
    void * buf,
    std::size_t len ) [pure virtual]
```

Reads bytes from the input stream and stores them into the buffer.

This method blocks until input data is available, end of file is detected, or another error is thrown.

If len is zero, then no bytes are read. If no byte is available because the stream is at end of file, an error is thrown.

The first byte read is stored into element b[0], the next one into b[1], and so on. If no error was thrown, the number of bytes read is always equal to len.

Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>buf</i>	the buffer into which the data is read.
<i>len</i>	the maximum number of bytes to read.

Implemented in [devfix::base::io::source](#).

7.3.3.5 skip()

```
virtual void devfix::base::io::inputstream::skip (
    std::size_t n ) [pure virtual]
```

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility.

Parameters

<i>n</i>	the number of bytes to be skipped.
----------	------------------------------------

Implemented in [devfix::base::io::source](#).

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

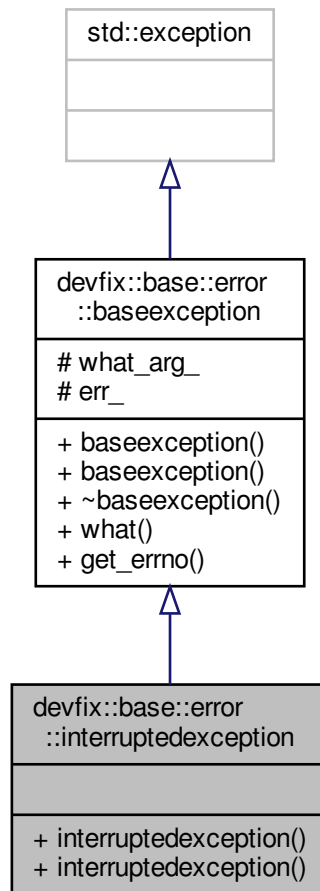
- [devfix/base/io/inputstream.h](#)

7.4 devfix::base::error::interruptedexception Struct Reference

Thrown when an operation is interrupted, either before or during the activity.

```
#include <interruptedexception.h>
```

Inheritance diagram for devfix::base::error::interruptedexception:



Public Member Functions

- [interruptedexception](#) (const std::string &what_arg, int err=-1)
- [interruptedexception](#) (const char *what_arg, int err=-1)

Additional Inherited Members

7.4.1 Detailed Description

Thrown when an operation is interrupted, either before or during the activity.

Occasionally a method may wish to test whether the current operation has been interrupted, and if so, to immediately throw this error.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 `interruptedexception()` [1/2]

```
devfix::base::error::interruptedexception::interruptedexception (
    const std::string & what_arg,
    int err = -1 ) [inline], [explicit]
```

Constructs the error object with `what_arg` as explanatory string that can be accessed through [what\(\)](#).

Parameters

<i>what_arg</i>	explanatory std::string
<i>err</i>	c error code (errno)

7.4.2.2 `interruptedexception()` [2/2]

```
devfix::base::error::interruptedexception::interruptedexception (
    const char * what_arg,
    int err = -1 ) [inline], [explicit]
```

Constructs the error object with `what_arg` as explanatory string that can be accessed through [what\(\)](#).

Parameters

<i>what_arg</i>	explanatory c-string
<i>err</i>	c error code (errno)

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

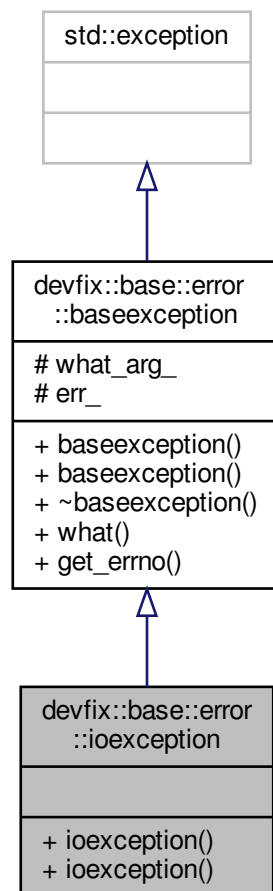
- [devfix/base/error/interruptedexception.h](#)

7.5 `devfix::base::error::ioexception` Struct Reference

Signals that an I/O error of some sort has occurred.

```
#include <ioexception.h>
```

Inheritance diagram for devfix::base::error::ioexception:



Public Member Functions

- [ioexception](#) (const std::string &what_arg, int err=-1)
- [ioexception](#) (const char *what_arg, int err=-1)

Additional Inherited Members

7.5.1 Detailed Description

Signals that an I/O error of some sort has occurred.

This class is the general class of exceptions produced by failed or interrupted I/O operations.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 `ioexception()` [1/2]

```
devfix::base::error::ioexception::ioexception (
    const std::string & what_arg,
    int err = -1 ) [inline], [explicit]
```

Constructs the error object with `what_arg` as explanatory string that can be accessed through [what\(\)](#).

Parameters

<i>what_arg</i>	explanatory std::string
<i>err</i>	c error code (errno)

7.5.2.2 `ioexception()` [2/2]

```
devfix::base::error::ioexception::ioexception (
    const char * what_arg,
    int err = -1 ) [inline], [explicit]
```

Constructs the error object with `what_arg` as explanatory string that can be accessed through [what\(\)](#).

Parameters

<i>what_arg</i>	explanatory c-string
<i>err</i>	c error code (errno)

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

- [devfix/base/error/ioexception.h](#)

7.6 `devfix::net::netbuilder` Struct Reference

```
#include <netbuilder.h>
```

Public Member Functions

- [netbuilder](#) ()=delete

Static Public Member Functions

- static std::unique_ptr< [socket](#) > [create_socket](#) (inetaddress adr)
Creates a socket and connects it to the specified remote internet address. The Socket will also bind() to the local address and port supplied.
- static std::unique_ptr< [serversocket](#) > [create_serversocket](#) (inetaddress adr, bool reuse_address=false)

7.6.1 Constructor & Destructor Documentation

7.6.1.1 netbuilder()

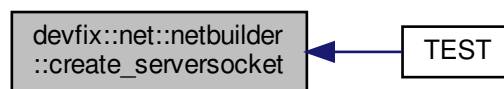
```
devfix::net::netbuilder::netbuilder ( ) [delete]
```

7.6.2 Member Function Documentation

7.6.2.1 create_serversocket()

```
std::unique_ptr< serversocket > devfix::net::netbuilder::create_serversocket (
    inetaddress adr,
    bool reuse_address = false ) [static]
```

Here is the caller graph for this function:



7.6.2.2 create_socket()

```
std::unique_ptr< socket > devfix::net::netbuilder::create_socket (
    inetaddress adr ) [static]
```

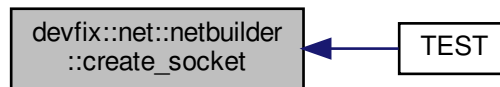
Creates a socket and connects it to the specified remote internet address. The Socket will also bind() to the local address and port supplied.

Parameters

<i>inetaddress</i>	remote address
--------------------	----------------

Returns

Here is the caller graph for this function:



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

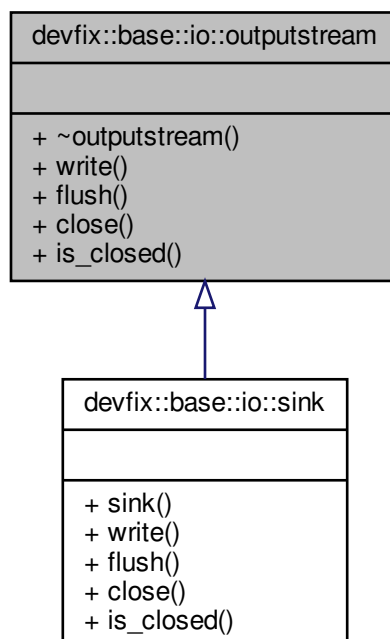
- [devfix/net/netbuilder.h](#)
- [devfix/net/netbuilder.cpp](#)

7.7 devfix::base::io::outputstream Struct Reference

Superclass of all classes representing an output stream of bytes.

```
#include <outputstream.h>
```

Inheritance diagram for devfix::base::io::outputstream:



Public Member Functions

- virtual `~outputstream()`=default
- virtual void `write` (const void *buf, std::size_t len)=0
Writes len bytes from the specified buffer to this output stream.
- virtual void `flush` ()=0
Flushes this outputstream and forces any buffered output bytes to be written out.
- virtual void `close` ()=0
Closes this outputstream and releases any system resources associated with this stream.
- virtual bool `is_closed` ()=0
Returns if the outputstream is closed or available for further calls of output operations.

7.7.1 Detailed Description

Superclass of all classes representing an output stream of bytes.

An output stream accepts output bytes and sends them to some sink. Applications that need to define a subclass of OutputStream must always provide a method that writes one byte of output.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 ~outputstream()

```
virtual devfix::base::io::outputstream::~~outputstream ( ) [virtual], [default]
```

7.7.3 Member Function Documentation

7.7.3.1 close()

```
virtual void devfix::base::io::outputstream::close ( ) [pure virtual]
```

Closes this *outputstream* and releases any system resources associated with this stream.

The general contract of close is that it closes the output stream. A closed stream cannot perform output operations and cannot be reopened.

Implemented in `devfix::base::io::sink`.

7.7.3.2 flush()

```
virtual void devfix::base::io::outputstream::flush ( ) [pure virtual]
```

Flushes this *outputstream* and forces any buffered output bytes to be written out.

The general contract of flush is that calling it is an indication that, if any bytes previously written have been buffered by the implementation of the output stream, such bytes should immediately be written to their intended destination.

If the intended destination of this stream is an abstraction provided by the underlying operating system, for example a file, then flushing the stream guarantees only that bytes previously written to the stream are passed to the operating system for writing; it does not guarantee that they are actually written to a physical device such as a disk drive.

Implemented in [devfix::base::io::sink](#).

7.7.3.3 is_closed()

```
virtual bool devfix::base::io::outputstream::is_closed ( ) [pure virtual]
```

Returns if the *outputstream* is closed or available for further calls of output operations.

Returns

true if the *outputstream* got previously closed.

Implemented in [devfix::base::io::sink](#).

7.7.3.4 write()

```
virtual void devfix::base::io::outputstream::write (
    const void * buf,
    std::size_t len ) [pure virtual]
```

Writes len bytes from the specified buffer to this output stream.

Element b[0] is the first byte written and b[len-1] is the last byte written by this operation.

Parameters

<i>buf</i>	the data.
<i>len</i>	the number of bytes to write.

Implemented in [devfix::base::io::sink](#).

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

- [devfix/base/io/outputstream.h](#)

7.8 devfix::net::serversocket Struct Reference

```
#include <serversocket.h>
```

Public Member Functions

- virtual [~serversocket](#) ()=default
- virtual std::unique_ptr< [socket](#) > [accept](#) ()=0
- virtual const [inetaddress](#) & [get_address](#) () const noexcept=0
- virtual bool [get_reuse_address](#) () const noexcept=0
- virtual void [set_accept_timeout](#) ([socket::timeout_t](#) timeout)=0
- virtual [socket::timeout_t](#) [get_accept_timeout](#) () const noexcept=0
- virtual void [close](#) ()=0
- virtual bool [is_closed](#) () const noexcept=0

7.8.1 Constructor & Destructor Documentation

7.8.1.1 ~serversocket()

```
virtual devfix::net::serversocket::~serversocket ( ) [virtual], [default]
```

7.8.2 Member Function Documentation

7.8.2.1 accept()

```
virtual std::unique_ptr<socket> devfix::net::serversocket::accept ( ) [pure virtual]
```

7.8.2.2 close()

```
virtual void devfix::net::serversocket::close ( ) [pure virtual]
```

7.8.2.3 get_accept_timeout()

```
virtual socket::timeout\_t devfix::net::serversocket::get_accept_timeout ( ) const [pure virtual],  
[noexcept]
```

7.8.2.4 get_address()

```
virtual const inetaddress& devfix::net::serversocket::get_address ( ) const [pure virtual],  
[noexcept]
```

7.8.2.5 get_reuse_address()

```
virtual bool devfix::net::serversocket::get_reuse_address ( ) const [pure virtual], [noexcept]
```

7.8.2.6 is_closed()

```
virtual bool devfix::net::serversocket::is_closed ( ) const [pure virtual], [noexcept]
```

7.8.2.7 set_accept_timeout()

```
virtual void devfix::net::serversocket::set_accept_timeout (  
    socket::timeout\_t timeout ) [pure virtual]
```

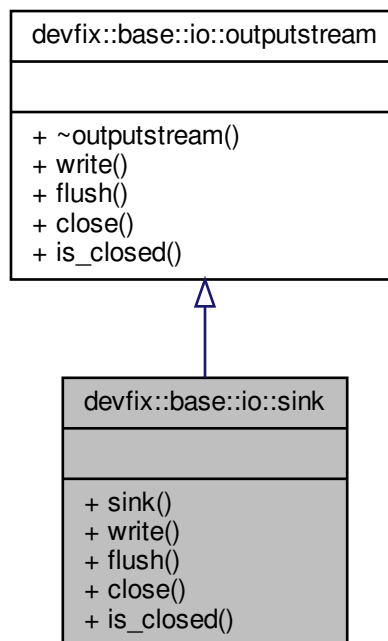
The documentation for this struct was generated from the following file:

- [devfix/net/serversocket.h](#)

7.9 devfix::base::io::sink Struct Reference

```
#include <sink.h>
```

Inheritance diagram for devfix::base::io::sink:



Public Member Functions

- `sink` (`write_t` write, `flush_t` flush, `close_t` close=DEFAULT_CLOSE, `is_closed_t` is_closed=DEFAULT_IS_CLOSED)
 - Writes `len` bytes from the specified buffer to this output stream.
- `void write` (`const void` *buf, `std::size_t` len)
 - Writes `len` bytes from the specified buffer to this output stream.
- `void flush` ()
 - Flushes this outputstream and forces any buffered output bytes to be written out.
- `void close` ()
 - Closes this outputstream and releases any system resources associated with this stream.
- `bool is_closed` ()
 - Returns if the outputstream is closed or available for further calls of output operations.

7.9.1 Constructor & Destructor Documentation

7.9.1.1 sink()

```

devfix::base::io::sink::sink (
    write_t write,
    flush_t flush,
    close_t close = DEFAULT_CLOSE,
    is_closed_t is_closed = DEFAULT_IS_CLOSED )
  
```

7.9.2 Member Function Documentation

7.9.2.1 close()

```
void devfix::base::io::sink::close ( ) [virtual]
```

Closes this *outputstream* and releases any system resources associated with this stream.

The general contract of `close` is that it closes the output stream. A closed stream cannot perform output operations and cannot be reopened.

Implements [devfix::base::io::outputstream](#).

7.9.2.2 flush()

```
void devfix::base::io::sink::flush ( ) [virtual]
```

Flushes this *outputstream* and forces any buffered output bytes to be written out.

The general contract of `flush` is that calling it is an indication that, if any bytes previously written have been buffered by the implementation of the output stream, such bytes should immediately be written to their intended destination.

If the intended destination of this stream is an abstraction provided by the underlying operating system, for example a file, then flushing the stream guarantees only that bytes previously written to the stream are passed to the operating system for writing; it does not guarantee that they are actually written to a physical device such as a disk drive.

Implements [devfix::base::io::outputstream](#).

7.9.2.3 is_closed()

```
bool devfix::base::io::sink::is_closed ( ) [virtual]
```

Returns if the *outputstream* is closed or available for further calls of output operations.

Returns

true if the *outputstream* got previously closed.

Implements [devfix::base::io::outputstream](#).

7.9.2.4 write()

```
void devfix::base::io::sink::write (
    const void * buf,
    std::size_t len ) [virtual]
```

Writes `len` bytes from the specified buffer to this output stream.

Element `b[0]` is the first byte written and `b[len-1]` is the last byte written by this operation.

Parameters

<i>buf</i>	the data.
<i>len</i>	the number of bytes to write.

Implements [devfix::base::io::outputstream](#).

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

- [devfix/base/io/sink.h](#)
- [devfix/base/io/sink.cpp](#)

7.10 devfix::net::socket Struct Reference

```
#include <socket.h>
```

Public Types

- typedef std::uint32_t [timeout_t](#)

Public Member Functions

- virtual [~socket](#) ()=default
- virtual const [inetaddress](#) & [get_local_address](#) () const noexcept=0
- virtual const [inetaddress](#) & [get_remote_address](#) () const noexcept=0
- virtual [base::io::inputstream](#) & [get_inputstream](#) () const noexcept=0
- virtual [base::io::outputstream](#) & [get_outputstream](#) () const noexcept=0
- virtual void [set_interrupted](#) (bool [interrupted](#)) noexcept=0
- virtual bool [interrupted](#) () const noexcept=0
- virtual void [set_timeout](#) ([timeout_t](#) timeout) noexcept=0
- virtual [timeout_t](#) [get_timeout](#) () const noexcept=0

Static Public Attributes

- static constexpr [timeout_t](#) [DEFAULT_TIMEOUT](#) = 3000
default read timeout in milliseconds
- static constexpr [timeout_t](#) [DEFAULT_READ_BLOCKING_TIME](#) = 100
default read timeout until refresh in milliseconds

7.10.1 Member Typedef Documentation

7.10.1.1 timeout_t

```
typedef std::uint32_t devfix::net::socket::timeout_t
```

7.10.2 Constructor & Destructor Documentation

7.10.2.1 ~socket()

```
virtual devfix::net::socket::~~socket ( ) [virtual], [default]
```

7.10.3 Member Function Documentation

7.10.3.1 get_inputstream()

```
virtual base::io::inputstream& devfix::net::socket::get_inputstream ( ) const [pure virtual],  
[noexcept]
```

7.10.3.2 get_local_address()

```
virtual const inetaddress& devfix::net::socket::get_local_address ( ) const [pure virtual],  
[noexcept]
```

7.10.3.3 get_outputstream()

```
virtual base::io::outputstream& devfix::net::socket::get_outputstream ( ) const [pure virtual],  
[noexcept]
```

7.10.3.4 get_remote_address()

```
virtual const inetaddress& devfix::net::socket::get_remote_address ( ) const [pure virtual],  
[noexcept]
```

7.10.3.5 get_timeout()

```
virtual timeout_t devfix::net::socket::get_timeout ( ) const [pure virtual], [noexcept]
```


7.10.3.6 interrupted()

```
virtual bool devfix::net::socket::interrupted ( ) const [pure virtual], [noexcept]
```

Returns

true if the socket is interrupted.

7.10.3.7 set_interrupted()

```
virtual void devfix::net::socket::set_interrupted (
    bool interrupted ) [pure virtual], [noexcept]
```

Set the socket as interrupted.

Parameters

<i>interrupted</i>	If set true, any read call returns after the read blocking time expired and throws an error.
--------------------	--

7.10.3.8 set_timeout()

```
virtual void devfix::net::socket::set_timeout (
    timeout_t timeout ) [pure virtual], [noexcept]
```

7.10.4 Member Data Documentation

7.10.4.1 DEFAULT_READ_BLOCKING_TIME

```
constexpr timeout_t devfix::net::socket::DEFAULT_READ_BLOCKING_TIME = 100 [static]
```

default read timeout until refresh in milliseconds

7.10.4.2 DEFAULT_TIMEOUT

```
constexpr timeout_t devfix::net::socket::DEFAULT_TIMEOUT = 3000 [static]
```

default read timeout in milliseconds

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

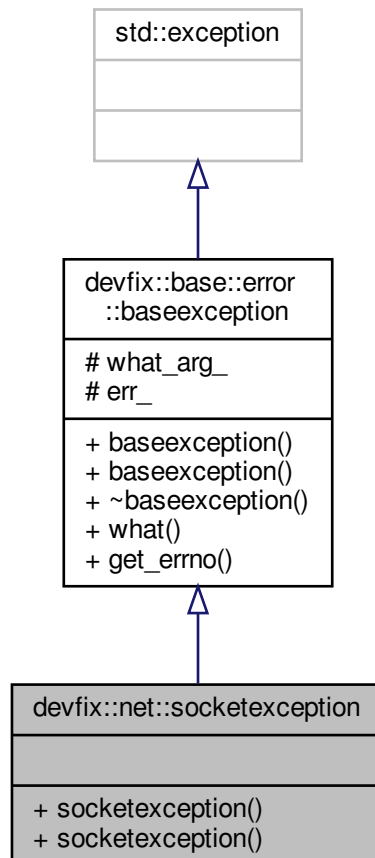
- devfix/net/[socket.h](#)

7.11 devfix::net::socketexception Struct Reference

Thrown to indicate that there is an error creating or accessing a Socket.

```
#include <socketexception.h>
```

Inheritance diagram for devfix::net::socketexception:



Public Member Functions

- [socketexception](#) (const std::string &what_arg, int err=-1)
- [socketexception](#) (const char *what_arg, int err=-1)

Additional Inherited Members

7.11.1 Detailed Description

Thrown to indicate that there is an error creating or accessing a Socket.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 socketexception() [1/2]

```
devfix::net::socketexception::socketexception (
    const std::string & what_arg,
    int err = -1 ) [inline], [explicit]
```

Constructs the error object with *what_arg* as explanatory string that can be accessed through [what\(\)](#).

Parameters

<i>what_arg</i>	explanatory std::string
<i>err</i>	c error code (errno)

7.11.2.2 socketexception() [2/2]

```
devfix::net::socketexception::socketexception (
    const char * what_arg,
    int err = -1 ) [inline], [explicit]
```

Constructs the error object with *what_arg* as explanatory string that can be accessed through [what\(\)](#).

Parameters

<i>what_arg</i>	explanatory c-string
<i>err</i>	c error code (errno)

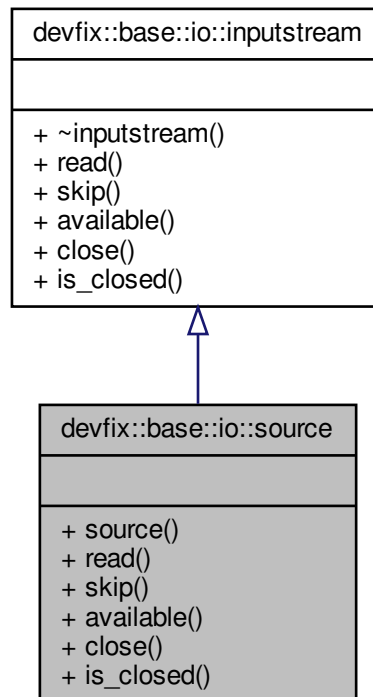
The documentation for this struct was generated from the following file:

- [devfix/net/socketexception.h](#)

7.12 devfix::base::io::source Struct Reference

```
#include <source.h>
```

Inheritance diagram for devfix::base::io::source:



Public Member Functions

- `source` (`read_t read`, `skip_t skip`, `available_t available`, `close_t close=DEFAULT_CLOSE`, `is_closed_t is_closed=DEFAULT_IS_CLOSED`)
- `void read` (`void *buf`, `std::size_t len`) override
Reads bytes from the input stream and stores them into the buffer.
- `void skip` (`std::size_t n`) override
*Skips over and discards *n* bytes of data from this input stream.*
- `std::size_t available` () override
Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.
- `void close` () override
Closes this input stream and releases any system resources associated with the stream.
- `bool is_closed` () override
Returns if the inputstream is closed or available for further calls of input operations.

7.12.1 Constructor & Destructor Documentation

7.12.1.1 source()

```
devfix::base::io::source::source (
    read_t read,
    skip_t skip,
    available_t available,
    close_t close = DEFAULT_CLOSE,
    is_closed_t is_closed = DEFAULT_IS_CLOSED )
```

7.12.2 Member Function Documentation

7.12.2.1 available()

```
std::size_t devfix::base::io::source::available ( ) [override], [virtual]
```

Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.

A single read or skip of this many bytes will not block, but may read or skip fewer bytes.

Note that while some implementations of *inputstream* will return the total number of bytes in the stream, many will not. It is never correct to use the return value of this method to allocate a buffer intended to hold all data in this stream.

A subclass' implementation of this method may choose to throw an `IOException` if this input stream has been closed by invoking the [close\(\)](#) method.

This method should be overridden by subclasses.

Returns

an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking or 0 when it reaches the end of the input stream.

Implements [devfix::base::io::inputstream](#).

7.12.2.2 close()

```
void devfix::base::io::source::close ( ) [override], [virtual]
```

Closes this input stream and releases any system resources associated with the stream.

A closed stream cannot perform input operations and cannot be reopened.

Implements [devfix::base::io::inputstream](#).

7.12.2.3 `is_closed()`

```
bool devfix::base::io::source::is_closed ( ) [override], [virtual]
```

Returns if the *inputstream* is closed or available for further calls of input operations.

Returns

true if the *inputstream* got previously closed.

Implements [devfix::base::io::inputstream](#).

7.12.2.4 `read()`

```
void devfix::base::io::source::read (
    void * buf,
    std::size_t len ) [override], [virtual]
```

Reads bytes from the input stream and stores them into the buffer.

This method blocks until input data is available, end of file is detected, or another error is thrown.

If len is zero, then no bytes are read. If no byte is available because the stream is at end of file, an error is thrown.

The first byte read is stored into element b[0], the next one into b[1], and so on. If no error was thrown, the number of bytes read is always equal to len.

Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>buf</i>	the buffer into which the data is read.
<i>len</i>	the maximum number of bytes to read.

Implements [devfix::base::io::inputstream](#).

7.12.2.5 `skip()`

```
void devfix::base::io::source::skip (
    std::size_t n ) [override], [virtual]
```

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility.

Parameters

<i>n</i>	the number of bytes to be skipped.
----------	------------------------------------

Implements [devfix::base::io::inputstream](#).

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

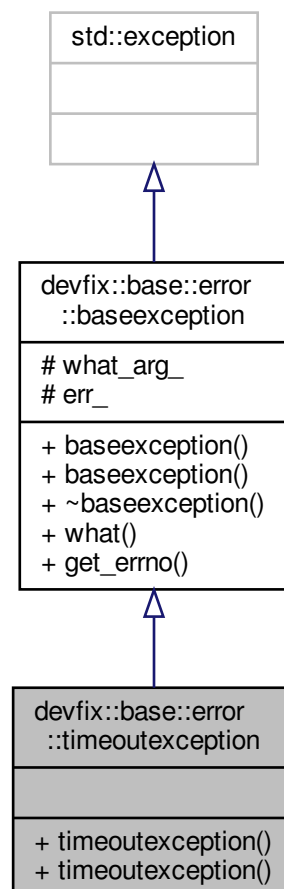
- [devfix/base/io/source.h](#)
- [devfix/base/io/source.cpp](#)

7.13 devfix::base::error::timeoutexception Struct Reference

Exception thrown when a blocking operation times out.

```
#include <timeoutexception.h>
```

Inheritance diagram for devfix::base::error::timeoutexception:



Public Member Functions

- [timeoutexception](#) (const std::string &what_arg, int err=-1)
- [timeoutexception](#) (const char *what_arg, int err=-1)

Additional Inherited Members

7.13.1 Detailed Description

Exception thrown when a blocking operation times out.

Blocking operations for which a timeout is specified need a means to indicate that the timeout has occurred. For many such operations it is possible to return a value that indicates timeout; when that is not possible or desirable then TimeoutException should be declared and thrown.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 [timeoutexception\(\)](#) [1/2]

```
devfix::base::error::timeoutexception::timeoutexception (
    const std::string & what_arg,
    int err = -1 ) [inline], [explicit]
```

Constructs the error object with what_arg as explanatory string that can be accessed through [what\(\)](#).

Parameters

<i>what_arg</i>	explanatory std::string
<i>err</i>	c error code (errno)

7.13.2.2 [timeoutexception\(\)](#) [2/2]

```
devfix::base::error::timeoutexception::timeoutexception (
    const char * what_arg,
    int err = -1 ) [inline], [explicit]
```

Constructs the error object with what_arg as explanatory string that can be accessed through [what\(\)](#).

Parameters

<i>what_arg</i>	explanatory c-string
<i>err</i>	c error code (errno)

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

- [devfix/base/error/timeoutexception.h](#)

Chapter 8

File Documentation

8.1 cmake-build-debug/CMakeFiles/3.15.3/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- `#define COMPILER_ID ""`
- `#define STRINGIFY_HELPER(X) #X`
- `#define STRINGIFY(X) STRINGIFY_HELPER(X)`
- `#define PLATFORM_ID`
- `#define ARCHITECTURE_ID`
- `#define DEC(n)`
- `#define HEX(n)`
- `#define CXX_STD __cplusplus`

Functions

- `int main (int argc, char *argv[])`

Variables

- `char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"`
- `char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"`
- `char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"`
- `const char * info_language_dialect_default`

8.1.1 Macro Definition Documentation

8.1.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

8.1.1.2 COMPILER_ID

```
#define COMPILER_ID ""
```

8.1.1.3 CXX_STD

```
#define CXX_STD __cplusplus
```

8.1.1.4 DEC

```
#define DEC(  
    n )
```

Value:

```
('0' + (((n) / 10000000) % 10)), \
('0' + (((n) / 1000000) % 10)), \
('0' + (((n) / 100000) % 10)), \
('0' + (((n) / 10000) % 10)), \
('0' + (((n) / 1000) % 10)), \
('0' + (((n) / 100) % 10)), \
('0' + (((n) / 10) % 10)), \
('0' + ((n) % 10))
```

8.1.1.5 HEX

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n) >> 28 & 0xF)), \
('0' + ((n) >> 24 & 0xF)), \
('0' + ((n) >> 20 & 0xF)), \
('0' + ((n) >> 16 & 0xF)), \
('0' + ((n) >> 12 & 0xF)), \
('0' + ((n) >> 8 & 0xF)), \
('0' + ((n) >> 4 & 0xF)), \
('0' + ((n) & 0xF))
```

8.1.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

8.1.1.7 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

8.1.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

8.1.2 Function Documentation

8.1.2.1 main()

```
int main (  
    int argc,  
    char * argv[] )
```

8.1.3 Variable Documentation

8.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

8.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

8.1.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
= "INFO" ":" "dialect_default["
```

```
"98"
```

```
"]"
```

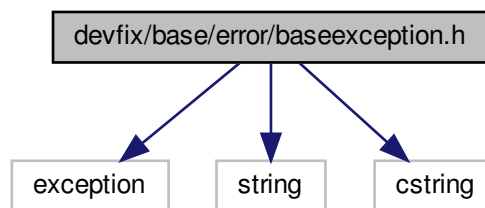
8.1.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

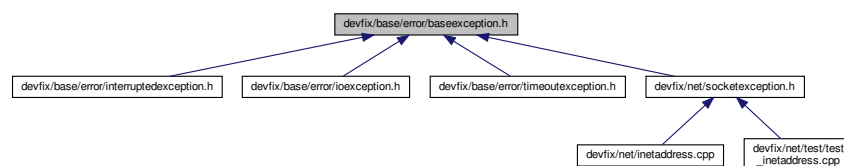
8.2 devfix/base/error/baseexception.h File Reference

```
#include <exception>
#include <string>
#include <cstring>
```

Include dependency graph for baseexception.h:



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



Classes

- struct [devfix::base::error::baseexception](#)

Abstract error base class.

Namespaces

- [devfix::base::error](#)

Namespace for general errors like timeouts or io failures.

Macros

- #define [exception_guard_m](#)(err, exception_class, message)
- #define [exception_guard](#)(err, exception_class) [exception_guard_m](#)(err, exception_class, std::strerror(errno))

8.2.1 Macro Definition Documentation

8.2.1.1 exception_guard

```
#define exception_guard(  
    err,  
    exception_class ) exception\_guard\_m(err, exception_class, std::strerror(errno))
```

8.2.1.2 exception_guard_m

```
#define exception_guard_m(  
    err,  
    exception_class,  
    message )
```

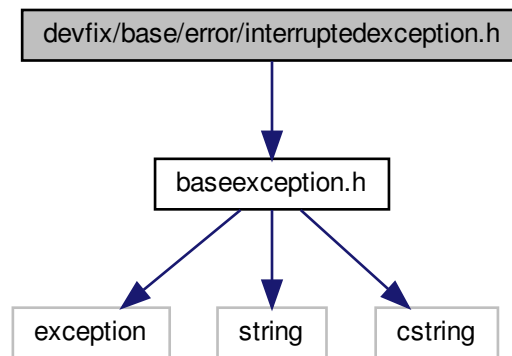
Value:

```
if (err) \  
    throw exception_class(message + std::string(" @ ") + SOURCE\_LINE, errno)
```

8.3 devfix/base/error/interruptedexception.h File Reference

```
#include "baseexception.h"
```

Include dependency graph for interruptedexception.h:



Classes

- struct [devfix::base::error::interruptedexception](#)

Thrown when an operation is interrupted, either before or during the activity.

Namespaces

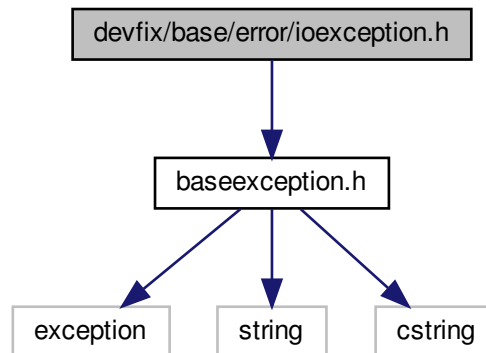
- [devfix::base::error](#)

Namespace for general errors like timeouts or io failures.

8.4 devfix/base/error/ioexception.h File Reference

```
#include "baseexception.h"
```


Include dependency graph for ioexception.h:



Classes

- struct [devfix::base::error::ioexception](#)
Signals that an I/O error of some sort has occurred.

Namespaces

- [devfix::base::error](#)
Namespace for general errors like timeouts or io failures.

8.5 devfix/base/error/namespace.h File Reference

Namespaces

- [devfix::base::error](#)
Namespace for general errors like timeouts or io failures.

8.6 devfix/base/io/namespace.h File Reference

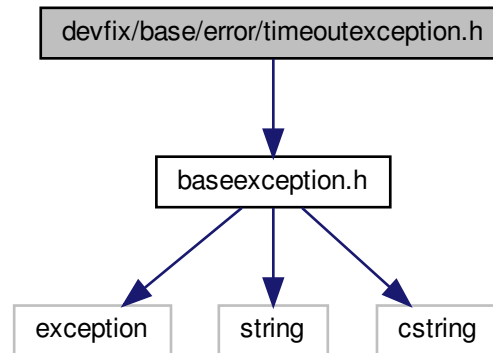
Namespaces

- [devfix::base::io](#)
Namespace for io tool, for instance streams.

8.7 devfix/base/error/timeoutexception.h File Reference

```
#include "baseexception.h"
```

Include dependency graph for timeoutexception.h:



Classes

- struct [devfix::base::error::timeoutexception](#)
Exception thrown when a blocking operation times out.

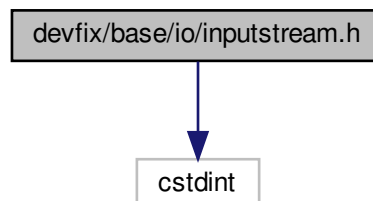
Namespaces

- [devfix::base::error](#)
Namespace for general errors like timeouts or io failures.

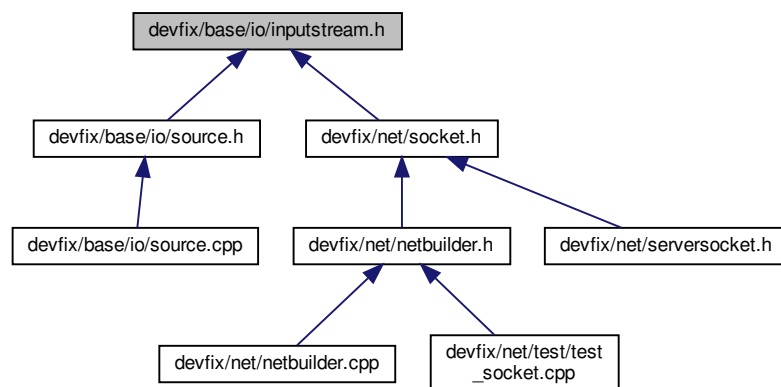
8.8 devfix/base/io/inputstream.h File Reference

```
#include <cstdint>
```

Include dependency graph for inputstream.h:



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



Classes

- struct `devfix::base::io::inputstream`
Superclass of all classes representing an input stream of bytes.

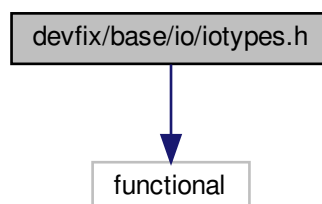
Namespaces

- `devfix::base::io`
Namespace for io tool, for instance streams.

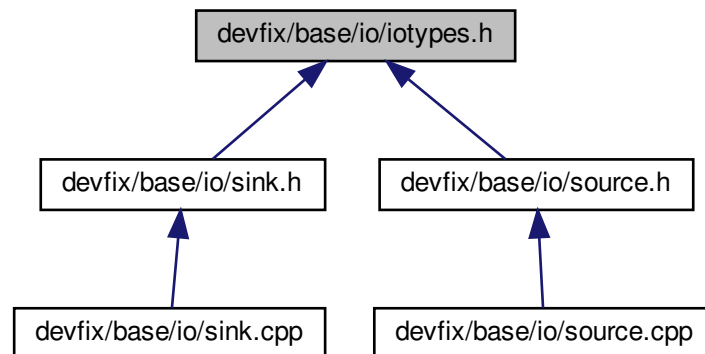
8.9 devfix/base/io/iotypes.h File Reference

```
#include <functional>
```

Include dependency graph for `iotypes.h`:



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



Namespaces

- [devfix::base::io](#)

Namespace for io tool, for instance streams.

Typedefs

- typedef std::function< void()> [devfix::base::io::close_t](#)
- typedef std::function< bool()> [devfix::base::io::is_closed_t](#)
- typedef std::function< void(void *, std::size_t)> [devfix::base::io::read_t](#)
- typedef std::function< void(std::size_t)> [devfix::base::io::skip_t](#)
- typedef std::function< std::size_t()> [devfix::base::io::available_t](#)
- typedef std::function< void(const void *, std::size_t)> [devfix::base::io::write_t](#)
- typedef std::function< void()> [devfix::base::io::flush_t](#)

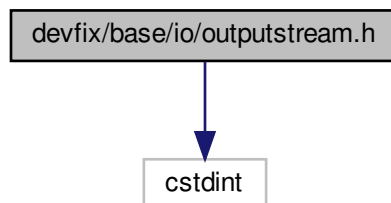
Variables

- const close_t [devfix::base::io::DEFAULT_CLOSE](#) = []() {}
- const is_closed_t [devfix::base::io::DEFAULT_IS_CLOSED](#) = []() { return false; }

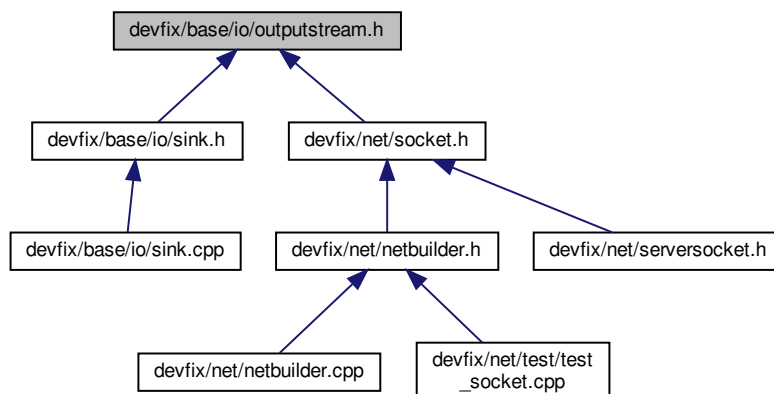
8.10 devfix/base/io/outputstream.h File Reference

```
#include <cstdint>
```

Include dependency graph for outputStream.h:



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



Classes

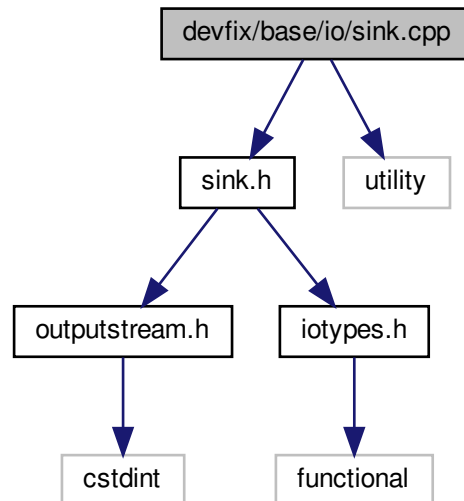
- struct [devfix::base::io::outputstream](#)
Superclass of all classes representing an output stream of bytes.

Namespaces

- [devfix::base::io](#)
Namespace for io tool, for instance streams.

8.11 devfix/base/io/sink.cpp File Reference

```
#include "sink.h"
#include <utility>
Include dependency graph for sink.cpp:
```



Namespaces

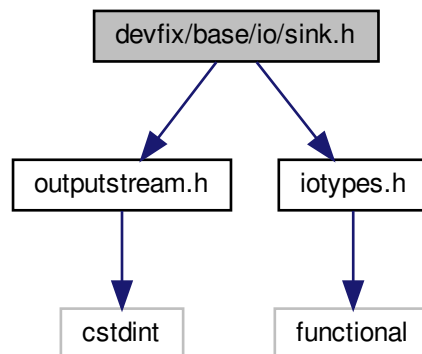
- [devfix::base::io](#)

Namespace for io tool, for instance streams.

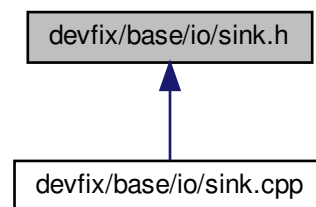
8.12 devfix/base/io/sink.h File Reference

```
#include "outputstream.h"
#include "iotypes.h"
```

Include dependency graph for sink.h:



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



Classes

- struct [devfix::base::io::sink](#)

Namespaces

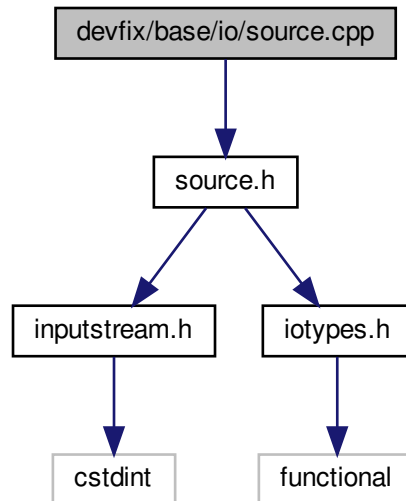
- [devfix::base::io](#)

Namespace for io tool, for instance streams.

8.13 devfix/base/io/source.cpp File Reference

```
#include "source.h"
```

Include dependency graph for source.cpp:



Namespaces

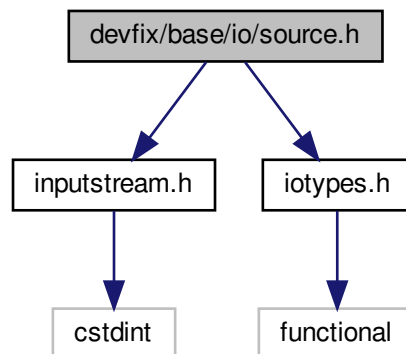
- [devfix::base::io](#)

Namespace for io tool, for instance streams.

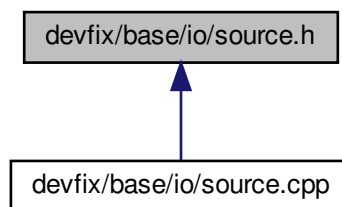
8.14 devfix/base/io/source.h File Reference

```
#include "inputstream.h"  
#include "iotypes.h"
```


Include dependency graph for source.h:



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



Classes

- struct [devfix::base::io::source](#)

Namespaces

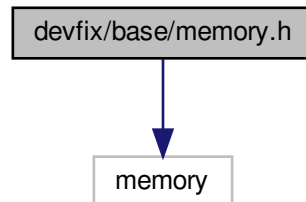
- [devfix::base::io](#)

Namespace for io tool, for instance streams.

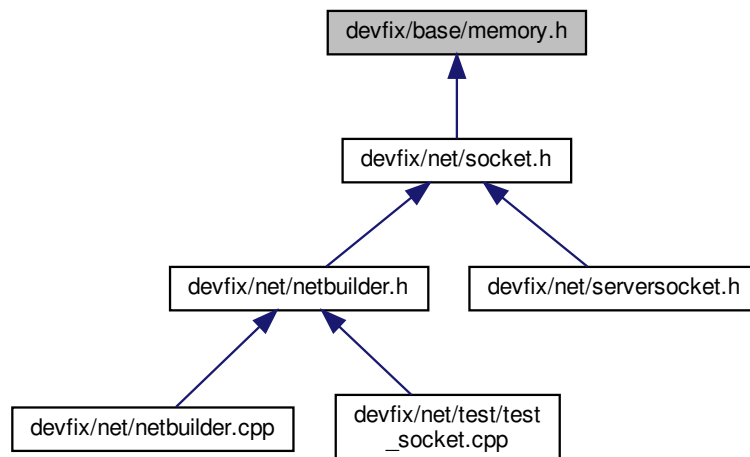
8.15 devfix/base/memory.h File Reference

```
#include <memory>
```

Include dependency graph for memory.h:



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



Namespaces

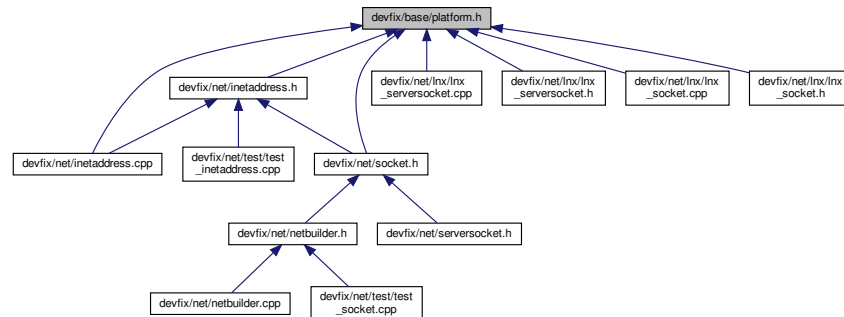
- [devfix::base](#)

Typedefs

- `template<class T >`
using [devfix::base::up](#) = `std::unique_ptr< T >`
- `template<class T >`
using [devfix::base::sp](#) = `std::shared_ptr< T >`

8.16 devfix/base/platform.h File Reference

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



Macros

- `#define PLATFORM_LINUX 0`
- `#define PLATFORM_UNSUPPORTED static_assert (false, "Platform not supported")`
- `#define __FILENAME__ &__FILE__[0]`
- `#define SOURCE_LINE std::string(__FILENAME__) + ":" + std::to_string(__LINE__) + ": in \"" + std::string(&__FUNCTION__[0]) + "\""`

8.16.1 Macro Definition Documentation

8.16.1.1 __FILENAME__

```
#define __FILENAME__ &__FILE__[0]
```

8.16.1.2 PLATFORM_LINUX

```
#define PLATFORM_LINUX 0
```

8.16.1.3 PLATFORM_UNSUPPORTED

```
#define PLATFORM_UNSUPPORTED static_assert ( false, "Platform not supported" )
```

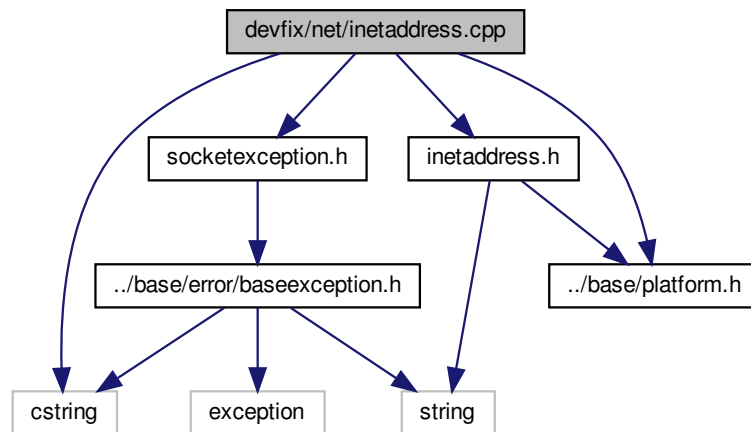
8.16.1.4 SOURCE_LINE

```
#define SOURCE_LINE std::string(__FILENAME__) + ":" + std::to_string(__LINE__) + " in \"" +  
std::string(&__FUNCTION__[0]) + "\""
```

8.17 devfix/net/inetaddress.cpp File Reference

```
#include <cstring>  
#include "../base/platform.h"  
#include "inetaddress.h"  
#include "socketexception.h"
```

Include dependency graph for inetaddress.cpp:



Namespaces

- [devfix::net](#)

Variables

- [PLATFORM_UNSUPPORTED](#)

8.17.1 Variable Documentation

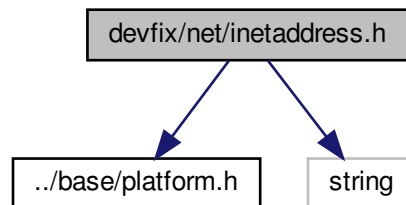
8.17.1.1 PLATFORM_UNSUPPORTED

PLATFORM_UNSUPPORTED

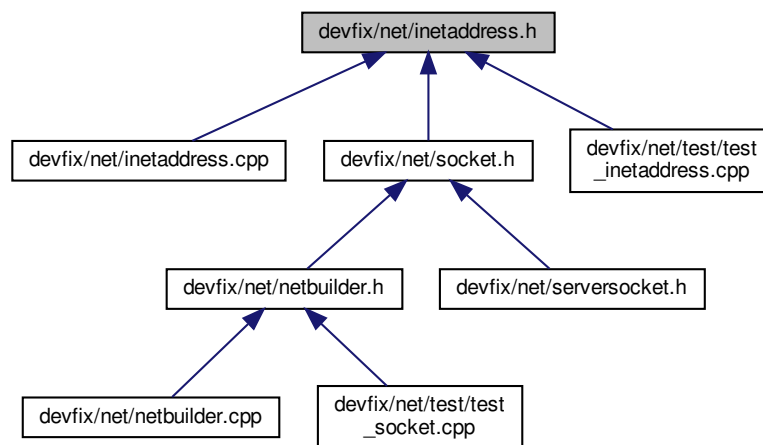
8.18 devfix/net/inetaddress.h File Reference

```
#include "../base/platform.h"
#include <string>
```

Include dependency graph for inetaddress.h:



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



Classes

- struct [devfix::net::inetaddress](#)

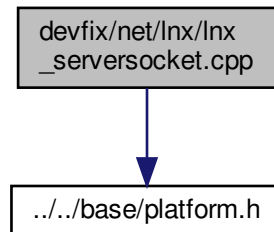
Namespaces

- [devfix::net](#)

8.19 devfix/net/lnx/lnx_serversocket.cpp File Reference

```
#include "../../base/platform.h"
```

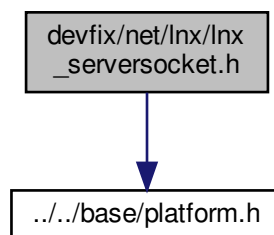
Include dependency graph for lnx_serversocket.cpp:



8.20 devfix/net/lnx/lnx_serversocket.h File Reference

```
#include "../../base/platform.h"
```

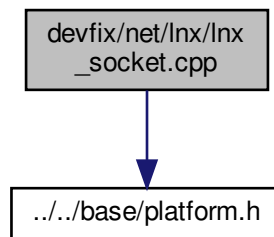
Include dependency graph for lnx_serversocket.h:



8.21 devfix/net/lnx/lnx_socket.cpp File Reference

```
#include "../../base/platform.h"
```

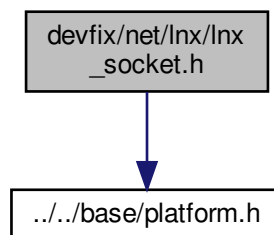
Include dependency graph for lnx_socket.cpp:



8.22 devfix/net/lnx/lnx_socket.h File Reference

```
#include "../../base/platform.h"
```

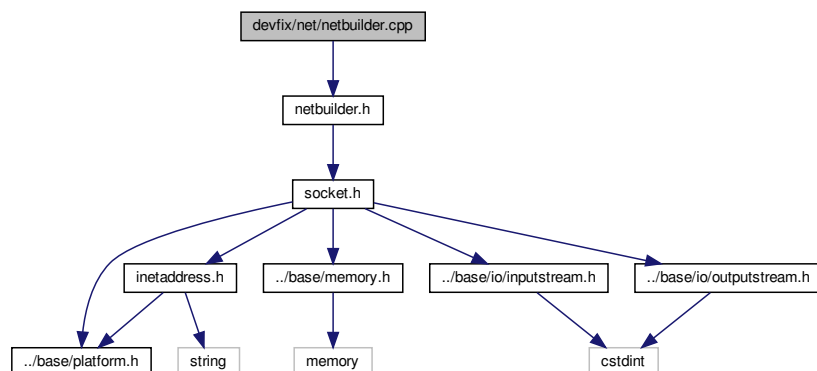
Include dependency graph for lnx_socket.h:



8.23 devfix/net/netbuilder.cpp File Reference

```
#include "netbuilder.h"
```

Include dependency graph for netbuilder.cpp:



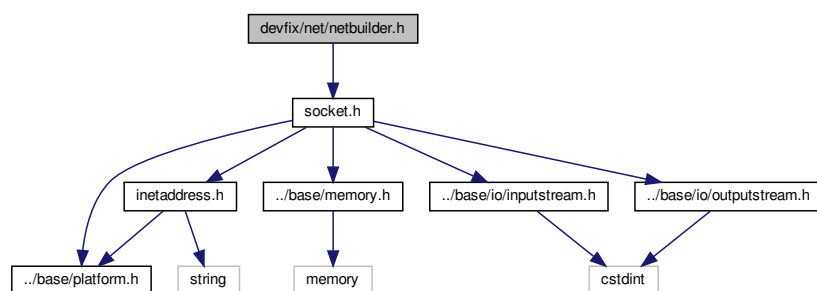
Namespaces

- [devfix::net](#)

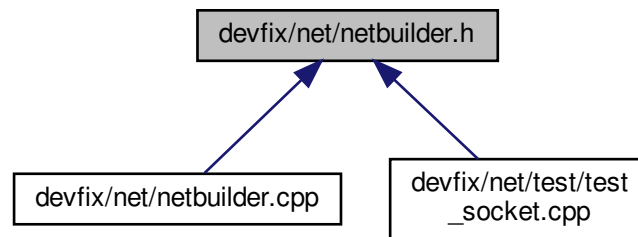
8.24 devfix/net/netbuilder.h File Reference

```
#include "socket.h"
```

Include dependency graph for netbuilder.h:



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



Classes

- struct [devfix::net::netbuilder](#)

Namespaces

- [devfix::net](#)

Variables

- [PLATFORM_UNSUPPORTED](#)

8.24.1 Variable Documentation

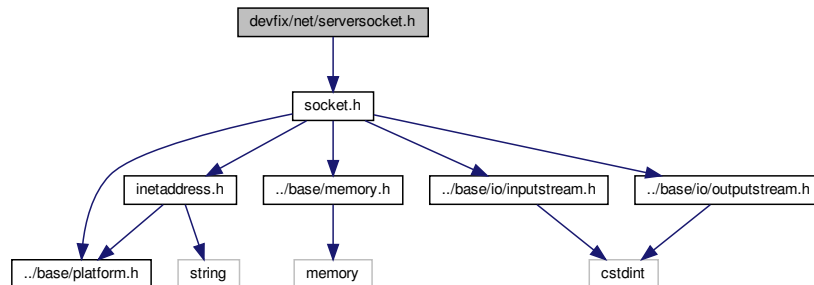
8.24.1.1 PLATFORM_UNSUPPORTED

PLATFORM_UNSUPPORTED

8.25 devfix/net/serversocket.h File Reference

```
#include "socket.h"
```

Include dependency graph for serversocket.h:



Classes

- struct [devfix::net::serversocket](#)

Namespaces

- [devfix::net](#)

8.26 devfix/net/socket.h File Reference

```
#include "inetaddress.h"
```

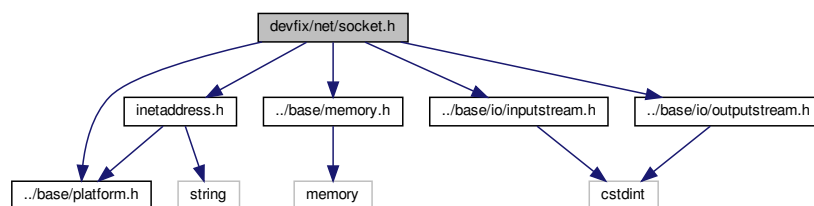
```
#include "../base/memory.h"
```

```
#include "../base/platform.h"
```

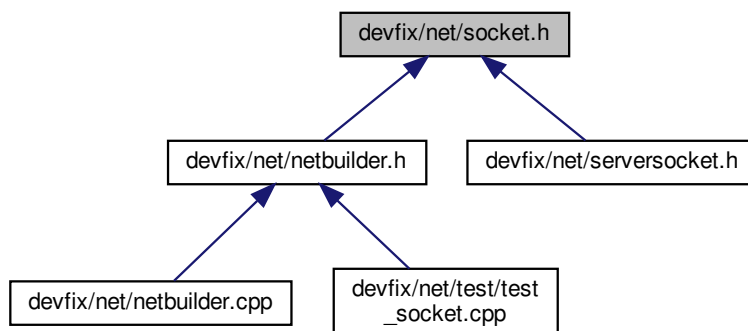
```
#include "../base/io/inputstream.h"
```

```
#include "../base/io/outputstream.h"
```

Include dependency graph for socket.h:



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



Classes

- struct `devfix::net::socket`

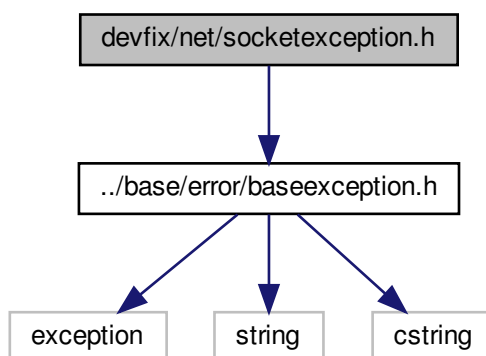
Namespaces

- `devfix::net`

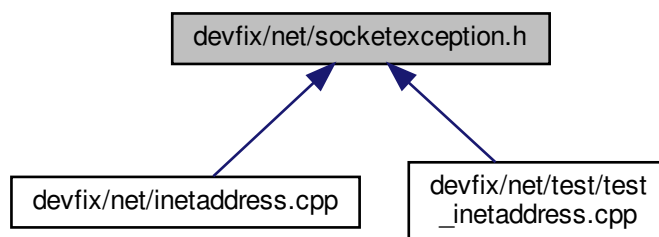
8.27 devfix/net/socketexception.h File Reference

```
#include "../base/error/baseexception.h"
```

Include dependency graph for `socketexception.h`:



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



Classes

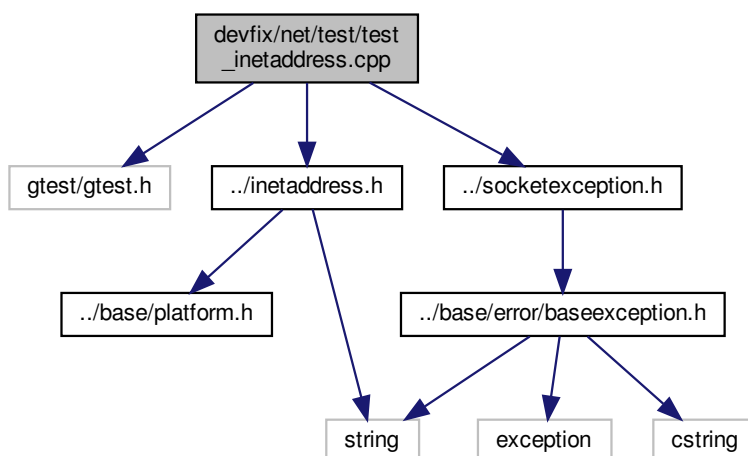
- struct `devfix::net::socketexception`
Thrown to indicate that there is an error creating or accessing a Socket.

Namespaces

- `devfix::net`

8.28 devfix/net/test/test_inetaddress.cpp File Reference

```
#include <gtest/gtest.h>
#include "../inetaddress.h"
#include "../socketexception.h"
Include dependency graph for test_inetaddress.cpp:
```



Functions

- [TEST](#) (InetAddress, CreateByHost)

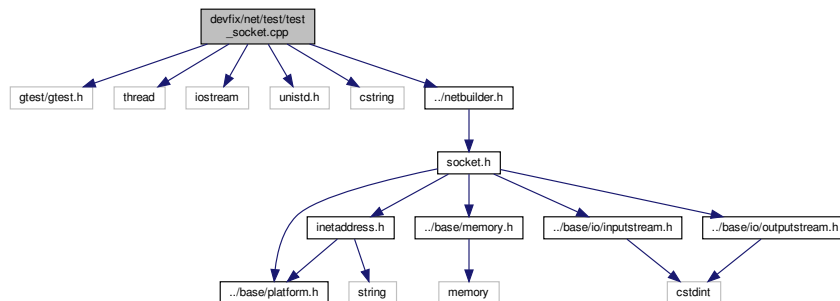
8.28.1 Function Documentation

8.28.1.1 TEST()

```
TEST (
    InetAddress ,
    CreateByHost )
```

8.29 devfix/net/test/test_socket.cpp File Reference

```
#include <gtest/gtest.h>
#include <thread>
#include <iostream>
#include <unistd.h>
#include <cstring>
#include "../netbuilder.h"
Include dependency graph for test_socket.cpp:
```



Functions

- [TEST](#) (Socket, Address)
- [TEST](#) (Socket, IO)

Variables

- constexpr `inetaddress::port_t` `TEST_PORT` = 30000
- constexpr long `TEST_LONG` = 1000000
- constexpr float `TEST_FLOAT` = 3.1415f
- constexpr double `TEST_DOUBLE` = 1.4142
- constexpr `std::array< float, 4 >` `TEST_ARRAY` = {1.0, 1.1, 1.2, 1.3}

8.29.1 Function Documentation

8.29.1.1 TEST() [1/2]

```
TEST (
    Socket ,
    Address )
```

8.29.1.2 TEST() [2/2]

```
TEST (
    Socket ,
    IO )
```

8.29.2 Variable Documentation

8.29.2.1 TEST_ARRAY

```
constexpr std::array<float, 4> TEST_ARRAY = {1.0, 1.1, 1.2, 1.3}
```

8.29.2.2 TEST_DOUBLE

```
constexpr double TEST_DOUBLE = 1.4142
```

8.29.2.3 TEST_FLOAT

```
constexpr float TEST_FLOAT = 3.1415f
```

8.29.2.4 TEST_LONG

```
constexpr long TEST_LONG = 1000000
```

8.29.2.5 TEST_PORT

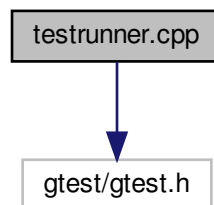
```
constexpr inetaddress::port\_t TEST_PORT = 30000
```

8.30 README.md File Reference

8.31 testrunner.cpp File Reference

```
#include <gtest/gtest.h>
```

Include dependency graph for testrunner.cpp:



Functions

- int [main](#) (int argc, char **argv)

8.31.1 Function Documentation

8.31.1.1 main()

```
int main (  
    int argc,  
    char ** argv )
```


Index

- `__FILENAME__`
 - `platform.h`, 65
- `~baseexception`
 - `devfix::base::error::baseexception`, 16
- `~inputstream`
 - `devfix::base::io::inputstream`, 22
- `~outputstream`
 - `devfix::base::io::outputstream`, 31
- `~serversocket`
 - `devfix::net::serversocket`, 33
- `~socket`
 - `devfix::net::socket`, 38
- `ARCHITECTURE_ID`
 - `CMakeCXXCompilerId.cpp`, 49
- `accept`
 - `devfix::net::serversocket`, 33
- `address_t`
 - `devfix::net::inetaddress`, 18
- `available`
 - `devfix::base::io::inputstream`, 22
 - `devfix::base::io::source`, 43
- `available_t`
 - `devfix::base::io`, 13
- `baseexception`
 - `devfix::base::error::baseexception`, 16
- `baseexception.h`
 - `exception_guard`, 53
 - `exception_guard_m`, 53
- `CMakeCXXCompilerId.cpp`
 - `ARCHITECTURE_ID`, 49
 - `COMPILER_ID`, 49
 - `CXX_STD`, 50
 - `DEC`, 50
 - `HEX`, 50
 - `info_arch`, 51
 - `info_compiler`, 51
 - `info_language_dialect_default`, 51
 - `info_platform`, 52
 - `main`, 51
 - `PLATFORM_ID`, 50
 - `STRINGIFY_HELPER`, 51
 - `STRINGIFY`, 50
- `COMPILER_ID`
 - `CMakeCXXCompilerId.cpp`, 49
- `CXX_STD`
 - `CMakeCXXCompilerId.cpp`, 50
- `close`
 - `devfix::base::io::inputstream`, 22
 - `devfix::base::io::outputstream`, 31
 - `devfix::base::io::sink`, 36
 - `devfix::base::io::source`, 43
 - `devfix::net::serversocket`, 33
- `close_t`
 - `devfix::base::io`, 13
- `cmake-build-debug/CMakeFiles/3.15.3/CompilerIdCXX`
 - `X/CMakeCXXCompilerId.cpp`, 49
- `create_serversocket`
 - `devfix::net::netbuilder`, 29
- `create_socket`
 - `devfix::net::netbuilder`, 29
- `DEFAULT_CLOSE`
 - `devfix::base::io`, 14
- `DEFAULT_IS_CLOSED`
 - `devfix::base::io`, 14
- `DEFAULT_READ_BLOCKING_TIME`
 - `devfix::net::socket`, 39
- `DEFAULT_TIMEOUT`
 - `devfix::net::socket`, 39
- `DEC`
 - `CMakeCXXCompilerId.cpp`, 50
- `devfix`, 11
- `devfix/base/error/baseexception.h`, 52
- `devfix/base/error/interruptedexception.h`, 54
- `devfix/base/error/ioexception.h`, 54
- `devfix/base/error/namespace.h`, 55
- `devfix/base/error/timeoutexception.h`, 56
- `devfix/base/io/inputstream.h`, 56
- `devfix/base/io/iotypes.h`, 57
- `devfix/base/io/namespace.h`, 55
- `devfix/base/io/outputstream.h`, 59
- `devfix/base/io/sink.cpp`, 60
- `devfix/base/io/sink.h`, 60
- `devfix/base/io/source.cpp`, 62
- `devfix/base/io/source.h`, 62
- `devfix/base/memory.h`, 64
- `devfix/base/platform.h`, 65
- `devfix/net/inetaddress.cpp`, 66
- `devfix/net/inetaddress.h`, 67
- `devfix/net/Inx/Inx_serversocket.cpp`, 68
- `devfix/net/Inx/Inx_serversocket.h`, 68
- `devfix/net/Inx/Inx_socket.cpp`, 68
- `devfix/net/Inx/Inx_socket.h`, 69
- `devfix/net/netbuilder.cpp`, 69
- `devfix/net/netbuilder.h`, 70
- `devfix/net/serversocket.h`, 72
- `devfix/net/socket.h`, 72

- devfix/net/socketexception.h, 73
- devfix/net/test/test_inetaddress.cpp, 74
- devfix/net/test/test_socket.cpp, 75
- devfix::base, 11
 - sp, 11
 - up, 11
- devfix::base::error, 12
- devfix::base::error::baseexception, 15
 - ~baseexception, 16
 - baseexception, 16
 - err_, 17
 - get_errno, 17
 - what, 17
 - what_arg_, 17
- devfix::base::error::interruptedexception, 24
 - interruptedexception, 26
- devfix::base::error::ioexception, 26
 - ioexception, 27, 28
- devfix::base::error::timeoutexception, 45
 - timeoutexception, 46
- devfix::base::io, 12
 - available_t, 13
 - close_t, 13
 - DEFAULT_CLOSE, 14
 - DEFAULT_IS_CLOSED, 14
 - flush_t, 13
 - is_closed_t, 13
 - read_t, 13
 - skip_t, 13
 - write_t, 14
- devfix::base::io::inputstream, 21
 - ~inputstream, 22
 - available, 22
 - close, 22
 - is_closed, 23
 - read, 23
 - skip, 23
- devfix::base::io::outputstream, 30
 - ~outputstream, 31
 - close, 31
 - flush, 31
 - is_closed, 32
 - write, 32
- devfix::base::io::sink, 34
 - close, 36
 - flush, 36
 - is_closed, 36
 - sink, 35
 - write, 36
- devfix::base::io::source, 41
 - available, 43
 - close, 43
 - is_closed, 43
 - read, 44
 - skip, 44
 - source, 42
- devfix::net, 14
- devfix::net::inetaddress, 18
 - address_t, 18
 - family_t, 18
 - get_address, 19
 - get_family, 19
 - get_host, 20
 - get_port, 20
 - inetaddress, 19
 - port_t, 18
- devfix::net::netbuilder, 28
 - create_serversocket, 29
 - create_socket, 29
 - netbuilder, 29
- devfix::net::serversocket, 33
 - ~serversocket, 33
 - accept, 33
 - close, 33
 - get_accept_timeout, 33
 - get_address, 33
 - get_reuse_address, 34
 - is_closed, 34
 - set_accept_timeout, 34
- devfix::net::socket, 37
 - ~socket, 38
 - DEFAULT_READ_BLOCKING_TIME, 39
 - DEFAULT_TIMEOUT, 39
 - get_inputstream, 38
 - get_local_address, 38
 - get_outputstream, 38
 - get_remote_address, 38
 - get_timeout, 38
 - interrupted, 38
 - set_interrupted, 39
 - set_timeout, 39
 - timeout_t, 37
- devfix::net::socketexception, 40
 - socketexception, 41
- err_
 - devfix::base::error::baseexception, 17
- exception_guard
 - baseexception.h, 53
- exception_guard_m
 - baseexception.h, 53
- family_t
 - devfix::net::inetaddress, 18
- flush
 - devfix::base::io::outputstream, 31
 - devfix::base::io::sink, 36
- flush_t
 - devfix::base::io, 13
- get_accept_timeout
 - devfix::net::serversocket, 33
- get_address
 - devfix::net::inetaddress, 19
 - devfix::net::serversocket, 33
- get_errno
 - devfix::base::error::baseexception, 17

- get_family
 - devfix::net::inetaddress, 19
- get_host
 - devfix::net::inetaddress, 20
- get_inputstream
 - devfix::net::socket, 38
- get_local_address
 - devfix::net::socket, 38
- get_outputstream
 - devfix::net::socket, 38
- get_port
 - devfix::net::inetaddress, 20
- get_remote_address
 - devfix::net::socket, 38
- get_reuse_address
 - devfix::net::serversocket, 34
- get_timeout
 - devfix::net::socket, 38
- HEX
 - CMakeCXXCompilerId.cpp, 50
- inetaddress
 - devfix::net::inetaddress, 19
- inetaddress.cpp
 - PLATFORM_UNSUPPORTED, 66
- info_arch
 - CMakeCXXCompilerId.cpp, 51
- info_compiler
 - CMakeCXXCompilerId.cpp, 51
- info_language_dialect_default
 - CMakeCXXCompilerId.cpp, 51
- info_platform
 - CMakeCXXCompilerId.cpp, 52
- interrupted
 - devfix::net::socket, 38
- interruptedexception
 - devfix::base::error::interruptedexception, 26
- ioexception
 - devfix::base::error::ioexception, 27, 28
- is_closed
 - devfix::base::io::inputstream, 23
 - devfix::base::io::outputstream, 32
 - devfix::base::io::sink, 36
 - devfix::base::io::source, 43
 - devfix::net::serversocket, 34
- is_closed_t
 - devfix::base::io, 13
- main
 - CMakeCXXCompilerId.cpp, 51
 - testrunner.cpp, 77
- netbuilder
 - devfix::net::netbuilder, 29
- netbuilder.h
 - PLATFORM_UNSUPPORTED, 71
- PLATFORM_ID
 - CMakeCXXCompilerId.cpp, 50
- PLATFORM_LINUX
 - platform.h, 65
- PLATFORM_UNSUPPORTED
 - inetaddress.cpp, 66
 - netbuilder.h, 71
 - platform.h, 65
- platform.h
 - __FILENAME__, 65
 - PLATFORM_LINUX, 65
 - PLATFORM_UNSUPPORTED, 65
 - SOURCE_LINE, 65
- port_t
 - devfix::net::inetaddress, 18
- README.md, 77
- read
 - devfix::base::io::inputstream, 23
 - devfix::base::io::source, 44
- read_t
 - devfix::base::io, 13
- SOURCE_LINE
 - platform.h, 65
- STRINGIFY_HELPER
 - CMakeCXXCompilerId.cpp, 51
- STRINGIFY
 - CMakeCXXCompilerId.cpp, 50
- set_accept_timeout
 - devfix::net::serversocket, 34
- set_interrupted
 - devfix::net::socket, 39
- set_timeout
 - devfix::net::socket, 39
- sink
 - devfix::base::io::sink, 35
- skip
 - devfix::base::io::inputstream, 23
 - devfix::base::io::source, 44
- skip_t
 - devfix::base::io, 13
- socketexception
 - devfix::net::socketexception, 41
- source
 - devfix::base::io::source, 42
- sp
 - devfix::base, 11
- TEST_ARRAY
 - test_socket.cpp, 76
- TEST_DOUBLE
 - test_socket.cpp, 76
- TEST_FLOAT
 - test_socket.cpp, 76
- TEST_LONG
 - test_socket.cpp, 76
- TEST_PORT
 - test_socket.cpp, 76
- TEST

- test_inetaddress.cpp, [75](#)
 - test_socket.cpp, [76](#)
- test_inetaddress.cpp
 - TEST, [75](#)
- test_socket.cpp
 - TEST_ARRAY, [76](#)
 - TEST_DOUBLE, [76](#)
 - TEST_FLOAT, [76](#)
 - TEST_LONG, [76](#)
 - TEST_PORT, [76](#)
 - TEST, [76](#)
- testrunner.cpp, [77](#)
 - main, [77](#)
- timeout_t
 - devfix::net::socket, [37](#)
- timeoutexception
 - devfix::base::error::timeoutexception, [46](#)
- up
 - devfix::base, [11](#)
- what
 - devfix::base::error::baseexception, [17](#)
- what_arg_
 - devfix::base::error::baseexception, [17](#)
- write
 - devfix::base::io::outputstream, [32](#)
 - devfix::base::io::sink, [36](#)
- write_t
 - devfix::base::io, [14](#)