c++ library collection

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 devfix Namespace Reference

Namespaces

- base

  Root namespace of devfix base library.
- dsp
- net

  Root namespace of devfix network library.

5.2 devfix::base Namespace Reference

Root namespace of devfix base library.

Namespaces

- _math
- error

  Namespace for general errors like timeouts or io failures.
- io

  Namespace for io tool, for instance streams.

Classes

- struct math
- struct strout
- struct strutil

Typedefs

- template<class T >
  using up = std::unique_ptr< T >

  Alias for std::unique_ptr.
- template<class T >
  using sp = std::shared_ptr< T >

  Alias for std::shared_ptr.

Functions

- template<typename T >
  constexpr const T * get_from_multistring (const char *str, const wchar_t *wstr)
- template<>
  constexpr const char * get_from_multistring< char > (const char *str, const wchar_t *wstr)
- template<>
  constexpr const wchar_t * get_from_multistring< wchar_t > (const char *str, const wchar_t *wstr)

5.2.1 Detailed Description

Root namespace of devfix base library.

This namespace provide classes for network communication and address conversion.

5.2.2 Typedef Documentation

5.2.2.1 sp

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

Alias for std::shared_ptr.

5.2.2.2 up

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

Alias for std::unique_ptr.

5.2.3 Function Documentation

5.2.3.1 get_from_multistring()

```
template<typename T >
constexpr const T* devfix::base::get_from_multistring (
            const char * str,
            const wchar_t * wstr )
```

5.2.3.2 get_from_multistring< char >()

```
template<>
constexpr const char* devfix::base::get_from_multistring< char > (
            const char * str,
            const wchar_t * wstr )
```

5.2.3.3 get_from_multistring< wchar_t >()

```
template<>
constexpr const wchar_t* devfix::base::get_from_multistring< wchar_t > (
            const char * str,
            const wchar_t * wstr )
```

5.3 devfix::base::_math Namespace Reference

Classes

- struct Table

5.4 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.

5.4.1 Detailed Description

Namespace for general errors like timeouts or io failures.

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

5.5 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

  Adapter class to create an outputstream from function pointers.
- struct source

  Adapter class to create an inputstream from function pointers.

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

5.5.1 Detailed Description

Namespace for io tool, for instance streams.

5.5.2 Typedef Documentation

5.5.2.1 available_t

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

5.5.2.2 close_t

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

5.5.2.3 flush_t

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

5.5.2.4 is_closed_t

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

5.5.2.5 read_t

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

5.5.2.6 skip_t

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

5.5.2.7 write_t

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

5.5.3 Variable Documentation

5.5.3.1 DEFAULT_CLOSE

```
const close_t devfix::base::io::DEFAULT_CLOSE
```

Initial value:

```
= []()
    {}
```

5.5.3.2 DEFAULT_IS_CLOSED

```
const is_closed_t devfix::base::io::DEFAULT_IS_CLOSED
```

Initial value:

```
= []()
    { return false; }
```

5.6 devfix::dsp Namespace Reference

Classes

- struct fft
- struct spectrogram
- struct window

5.7 devfix::net Namespace Reference

Root namespace of devfix network library.

Classes

- struct inetaddress

  Class for management and conversion of internet addresses.
- struct netbuilder

  Builder class for platform independent instantiation.
- struct serversocket

  This class implements server sockets.
- struct socket

  This class implements client sockets (also called just "sockets").
- struct socketexception

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

5.7.1 Detailed Description

Root namespace of devfix network library.

This namespace contains only lightweight code, necessary for other libraries and projects.

Chapter 6

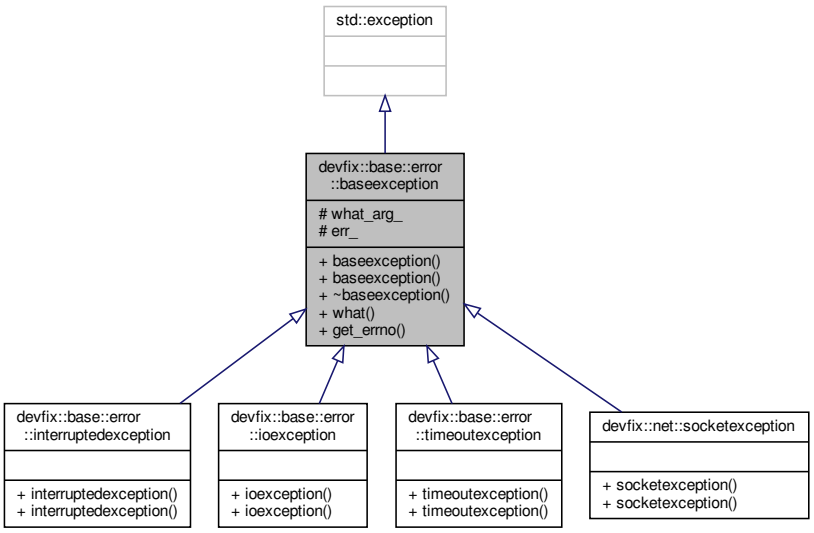
Class Documentation

6.1 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_

6.1.1 Detailed Description

Abstract error base class.

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

6.1.2 Constructor & Destructor Documentation

6.1.2.1 baseexception() [1/2]

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

Delete simple constructor, always enforce a failure description.

6.1.2.2 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().

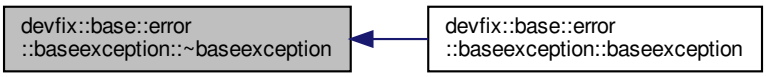
Parameters

what_arg	failure description
err	c error code (errno)

6.1.2.3 ~baseexception()

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

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



6.1.3 Member Function Documentation

6.1.3.1 get_errno()

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

6.1.3.2 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

6.1.4 Member Data Documentation

6.1.4.1 err_

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

6.1.4.2 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

6.2 devfix::dsp::fft Struct Reference

`#include <fft.h>`

Public Types

- using math = devfix::base::math

Static Public Member Functions

- template<typename FloatT >
  static void transform_inplace (std::complex< FloatT > *field, std::size_t len)
- template<typename FloatT >
  static void transform_inplace (std::vector< std::complex< FloatT >> &vec)
- template<typename FloatT , std::size_t N>
  static void transform_inplace (std::array< std::complex< FloatT >, N > &arr)

6.2.1 Member Typedef Documentation

6.2.1.1 math

```
using devfix::dsp::fft::math = devfix::base::math
```

6.2.2 Member Function Documentation

6.2.2.1 transform_inplace() [1/3]

```
template<typename FloatT >
static void devfix::dsp::fft::transform_inplace (
            std::complex< FloatT > * field,
            std::size_t len ) [inline], [static]
```

6.2.2.2 transform_inplace() [2/3]

```
template<typename FloatT >
static void devfix::dsp::fft::transform_inplace (
            std::vector< std::complex< FloatT >> & vec ) [inline], [static]
```

6.2.2.3 transform_inplace() [3/3]

```
template<typename FloatT , std::size_t N>
static void devfix::dsp::fft::transform_inplace (
            std::array< std::complex< FloatT >, N > & arr )  [inline], [static]
```

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

- devfix/dsp/fft.h

6.3 devfix::net::inetaddress Struct Reference

Class for management and conversion of internet addresses.

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

Public Types

- enum family_t : char { family_t::UNSPECIFIED = 0, family_t::IPV4 = 1 }

  Supported underlying protocols.
- typedef std::uint32_t address_t

  ipv4 internet address
- typedef std::uint16_t port_t

  protocol port

Public Member Functions

- inetaddress ()=default

  Creates a default inetaddress, zeroed address, port and unspecified family.
- inetaddress (const std::string &host, port_t port, family_t family=family_t::IPV4)

  Creates an inetaddress by given hostname (dns or ip address).
- std::string get_host () const noexcept

  Convert binary host address to string.
- address_t get_address () const
- port_t get_port () const
- family_t get_family () const

6.3.1 Detailed Description

Class for management and conversion of internet addresses.

6.3.2 Member Typedef Documentation

6.3.2.1 address_t

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

ipv4 internet address

6.3.2.2 port_t

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

protocol port

6.3.3 Member Enumeration Documentation

6.3.3.1 family_t

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

Supported underlying protocols.

Enumerator

UNSPECIFIED	
IPV4	

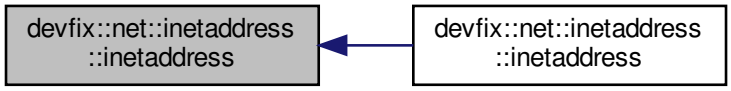
6.3.4 Constructor & Destructor Documentation

6.3.4.1 inetaddress() [1/2]

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

Creates a default *inetaddress*, zeroed address, port and unspecified family.

Here is the caller graph for this function:



6.3.4.2 inetaddress() [2/2]

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

Creates an *inetaddress* by given hostname (dns or ip address).

Parameters

host	hostname, gets converted to ipv4
port	protocol port
family	underlying protocol

6.3.5 Member Function Documentation

6.3.5.1 get_address()

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

Returns

binary address

6.3.5.2 get_family()

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

Returns

family of underlying protocol

6.3.5.3 get_host()

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

Convert binary host address to string.

Returns

address as string

6.3.5.4 get_port()

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

Returns

protocol port

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

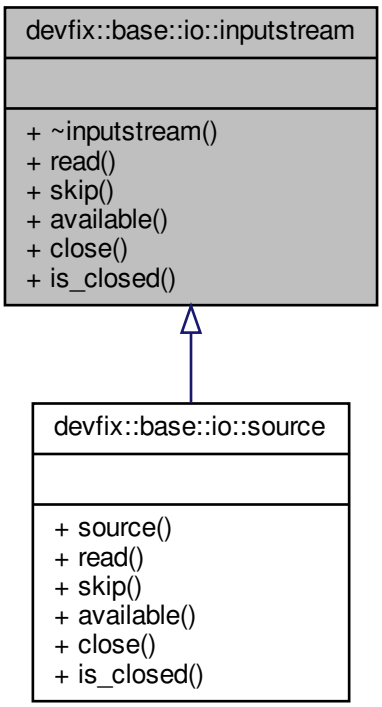
- devfix/net/inetaddress.h
- devfix/net/inetaddress.cpp

6.4 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.

6.4.1 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.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ~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.

6.4.3 Member Function Documentation

6.4.3.1 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.

6.4.3.2 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.

6.4.3.3 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.

6.4.3.4 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

buf	the buffer into which the data is read.
len	the maximum number of bytes to read.

Implemented in devfix::base::io::source.

6.4.3.5 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

n	the number of bytes to be skipped.

Implemented in devfix::base::io::source.

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

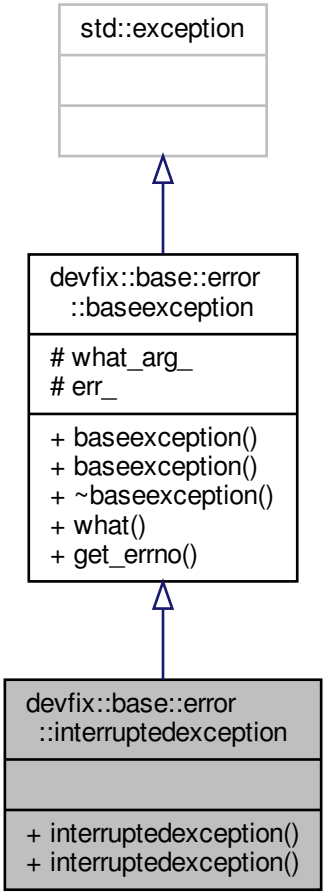
- devfix/base/io/inputstream.h

6.5 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

6.5.1 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.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 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

what_arg	explanatory std::string
err	c error code (errno)

6.5.2.2 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

what_arg	explanatory c-string
err	c error code (errno)

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

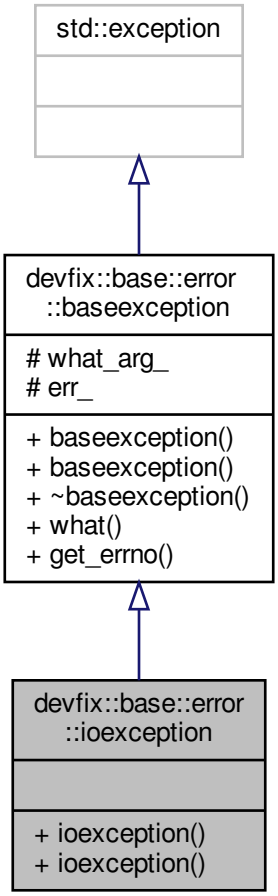
- devfix/base/error/interruptedexception.h

6.6 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

6.6.1 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.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 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

what_arg	explanatory std::string
err	c error code (errno)

6.6.2.2 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

what_arg	explanatory c-string
err	c error code (errno)

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

- devfix/base/error/ioexception.h

6.7 devfix::base::math Struct Reference

```
#include <math.h>
```

Public Types

- template<typename T , std::size_t N, typename G , typename ... Args>
  using Table = _math::Table< T, N, G, Args... >

Static Public Member Functions

- static constexpr std::uint32_t reverse_bits (std::uint32_t val, std::size_t bits)
- template<typename T , class = typename std::enable_if<std::is_unsigned<T>::value>::type>
  static constexpr T popcount (T val)

6.7.1 Member Typedef Documentation

6.7.1.1 Table

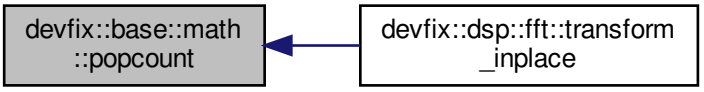
```
template<typename T , std::size_t N, typename G , typename ...  Args>
using devfix::base::math::Table = _math::Table<T, N, G, Args...>
```

6.7.2 Member Function Documentation

6.7.2.1 popcount()

```
template<typename T , class = typename std::enable_if<std::is_unsigned<T>::value>::type>
static constexpr T devfix::base::math::popcount (
            T val )  [inline], [static]
```

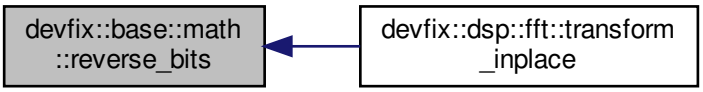
Here is the caller graph for this function:



6.7.2.2 reverse_bits()

```
static constexpr std::uint32_t devfix::base::math::reverse_bits (
            std::uint32_t val,
            std::size_t bits )  [inline], [static]
```

Here is the caller graph for this function:



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

- devfix/base/math.h

6.8 devfix::net::netbuilder Struct Reference

Builder class for platform independent instantiation.

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

Public Member Functions

- netbuilder ()=delete

  Allow no instances of builder class.

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.
- static std::unique_ptr< serversocket > create_serversocket (inetaddress adr, bool reuse_address=false)

  Creates a server socket and binds it to the supplied local address.

6.8.1 Detailed Description

Builder class for platform independent instantiation.

This class is the only one which has access to the constructors of network classes.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 netbuilder()

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

Allow no instances of builder class.

6.8.3 Member Function Documentation

6.8.3.1 create_serversocket()

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

Creates a server socket and binds it to the supplied local address.

Parameters

adr	local address for access restriction and port to listen on
reuse_address	if true allow bind to a port which remains in TIME_WAIT state

Returns

socket in listen state with platform specific implementation

6.8.3.2 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.

Parameters

adr	remote address

Returns

connected socket with platform specific implementation

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

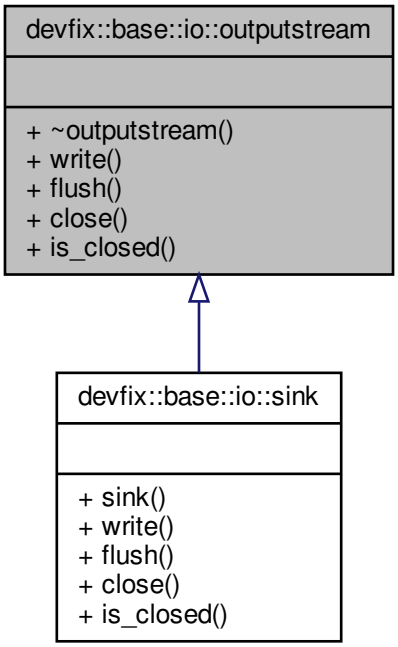
- devfix/net/netbuilder.h
- devfix/net/netbuilder.cpp

6.9 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.

6.9.1 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.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 ~outputstream()

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

6.9.3 Member Function Documentation

6.9.3.1 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.

6.9.3.2 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.

6.9.3.3 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.

6.9.3.4 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

buf	the data.
len	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

6.10 devfix::net::serversocket Struct Reference

This class implements server sockets.

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

Public Member Functions

- virtual ~serversocket ()=default
- virtual std::unique_ptr< socket > accept ()=0

  Listens for a connection to be made to this socket and accepts it.
- virtual const inetaddress & get_address () const noexcept=0

  Get the inetaddress the server is listening on.

- virtual bool get_reuse_address () const noexcept=0

  Get if binding to a port which remains in TIME_WAIT state is allowed.
- virtual void set_accept_timeout (socket::timeout_t timeout)=0

  Set the accept timeout.
- virtual socket::timeout_t get_accept_timeout () const noexcept=0

  Get the accept timeout.
- virtual void close ()=0

  Closes this serversocket and releases any system resources associated.
- virtual bool is_closed () const noexcept=0

  Returns if the serversocket is closed or available for further calls of io operations like accept().

6.10.1 Detailed Description

This class implements server sockets.

A server socket waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 ~serversocket()

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

6.10.3 Member Function Documentation

6.10.3.1 accept()

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

Listens for a connection to be made to this socket and accepts it.

The method blocks until a connection is made. A new Socket s is created and returned.

Returns

client socket

6.10.3.2 close()

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

Closes this *serversocket* and releases any system resources associated.

The general contract of close is that it closes the serversocket. A closed *serversocket* cannot perform io operations and cannot be reopened.

6.10.3.3 get_accept_timeout()

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

Get the accept timeout.

If the timeout ist zero the feature is disabled.

Returns

timeout

6.10.3.4 get_address()

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

Get the *inetaddress* the server is listening on.

The ip address of the inetaddress can be used to restrict the access of clients to the server.

Returns

bound inetaddress

6.10.3.5 get_reuse_address()

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

Get if binding to a port which remains in TIME_WAIT state is allowed.

Returns

true if allowed

6.10.3.6 is_closed()

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

Returns if the *serversocket* is closed or available for further calls of io operations like accept().

Returns

true if the *serversocket* got previously closed.

6.10.3.7 set_accept_timeout()

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

Set the accept timeout.

If a call of accept() take longer than this timeout, an timeoutexception is thrown. A timeout of zero disables this feature.

Parameters

timeout	

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

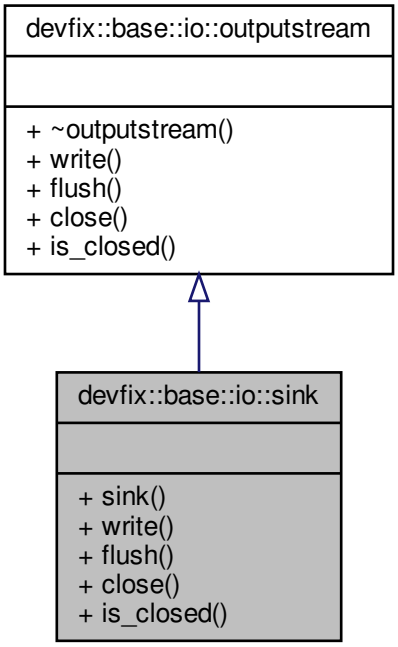
- devfix/net/serversocket.h

6.11 devfix::base::io::sink Struct Reference

Adapter class to create an *outputstream* from function pointers.

```
#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_C↩ LOSED)

  Create an outputstream from function pointers to the member the functions of the interface.
- void write (const void ∗buf, std::size_t len) override

  Writes len bytes from the specified buffer to this output stream.
- void flush () override

  Flushes this outputstream and forces any buffered output bytes to be written out.
- void close () override

  Closes this outputstream and releases any system resources associated with this stream.
- bool is_closed () override

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

6.11.1 Detailed Description

Adapter class to create an *outputstream* from function pointers.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 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 )
```

Create an *outputstream* from function pointers to the member the functions of the interface.

Parameters

write	function pointer to implementation of write()
flush	function pointer to implementation of flush()
close	function pointer to implementation of close()
is_closed	function pointer to implementation of is_closed()

6.11.3 Member Function Documentation

6.11.3.1 close()

```
void devfix::base::io::sink::close ( )  [override], [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.

6.11.3.2 flush()

```
void devfix::base::io::sink::flush ( )  [override], [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.

6.11.3.3 is_closed()

```
bool devfix::base::io::sink::is_closed ( )  [override], [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.

6.11.3.4 write()

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

buf	the data.
len	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

6.12 devfix::net::socket Struct Reference

This class implements client sockets (also called just "sockets").

```
#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

  Get local address with local ip address and the tcp input port.
- virtual const inetaddress & get_remote_address () const noexcept=0

  Get remote address with remote ip address and the tcp output stream.
- virtual base::io::inputstream & get_inputstream () const noexcept=0

  Get the instance of the socket input stream, which refers to the socket file descriptor access.
- virtual base::io::outputstream & get_outputstream () const noexcept=0

  Get the instance of the socket output stream, which refers to the socket file descriptor access.
- virtual void set_interrupted (bool interrupted) noexcept=0

  Set the socket as interrupted.
- virtual bool get_interrupted () const noexcept=0
- virtual void set_timeout (timeout_t timeout) noexcept=0

  Set the socket timeout.
- virtual timeout_t get_timeout () const noexcept=0

  Get the socket timeout.

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 time until refresh in milliseconds
- static constexpr timeout_t DEFAULT_WRITE_BLOCKING_TIME = 100

  default write time until refresh in milliseconds

6.12.1 Detailed Description

This class implements client sockets (also called just "sockets").

A socket is an endpoint for communication between two machines. The actual work of the socket is performed by an instance of the platform specific implementation class.

6.12.2 Member Typedef Documentation

6.12.2.1 timeout_t

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

6.12.3 Constructor & Destructor Documentation

6.12.3.1 ~socket()

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

6.12.4 Member Function Documentation

6.12.4.1 get_inputstream()

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

Get the instance of the socket input stream, which refers to the socket file descriptor access.

Returns

socket *inputstream*

6.12.4.2 get_interrupted()

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

Returns

true if socket should get interrupted

6.12.4.3 get_local_address()

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

Get local address with local ip address and the tcp input port.

Returns

local *inetaddress*

6.12.4.4 get_outputstream()

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

Get the instance of the socket output stream, which refers to the socket file descriptor access.

Returns

socket *outputstream*

6.12.4.5 get_remote_address()

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

Get remote address with remote ip address and the tcp output stream.

Returns

remote *inetaddress*

6.12.4.6 get_timeout()

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

Get the socket timeout.

Returns

timeout

6.12.4.7 set_interrupted()

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

Set the socket as interrupted.

If set to true, any read call returns after the read blocking time expired and throws an *interruptedexception*. The flag must be cleared (set to false) by hand.

Parameters

interrupted	if true, socket should get interrupted

6.12.4.8 set_timeout()

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

Set the socket timeout.

If a call of read() on the *inputstream* takes longer than this timeout, an *timeoutexception* is thrown.

Parameters

timeout	for reading

6.12.5 Member Data Documentation

6.12.5.1 DEFAULT_READ_BLOCKING_TIME

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

default read time until refresh in milliseconds

6.12.5.2 DEFAULT_TIMEOUT

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

default read timeout in milliseconds

6.12.5.3 DEFAULT_WRITE_BLOCKING_TIME

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

default write time until refresh in milliseconds

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

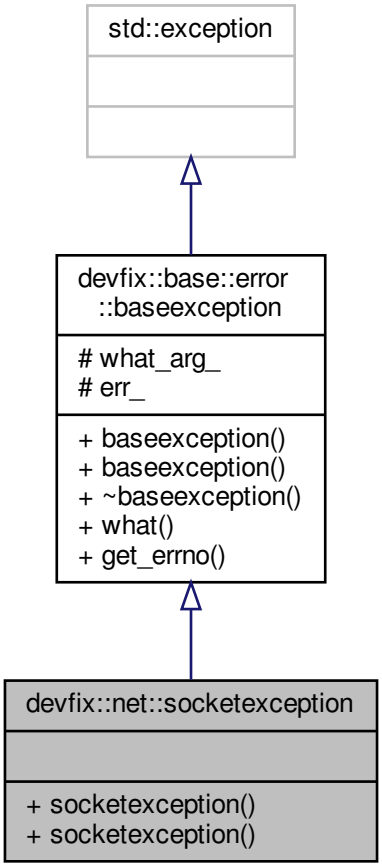
- devfix/net/socket.h

6.13 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

6.13.1 Detailed Description

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

6.13.2 Constructor & Destructor Documentation

6.13.2.1 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

what_arg	explanatory std::string
err	c error code (errno)

6.13.2.2 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

what_arg	explanatory c-string
err	c error code (errno)

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

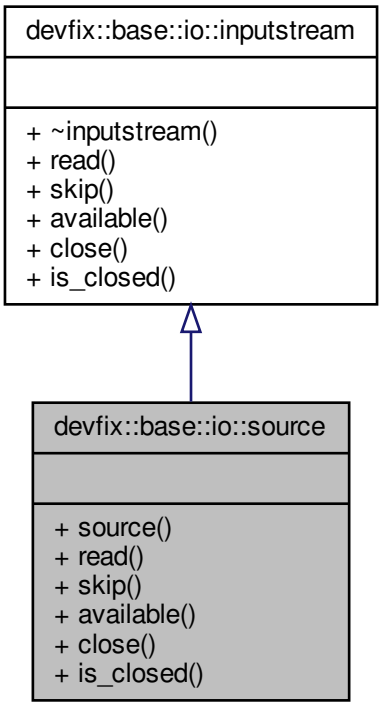
- devfix/net/socketexception.h

6.14 devfix::base::io::source Struct Reference

Adapter class to create an *inputstream* from function pointers.

```
#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)

  Create an inputstream from function pointers to the member the functions of the interface.
- 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.

6.14.1 Detailed Description

Adapter class to create an *inputstream* from function pointers.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 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 )
```

Create an *inputstream* from function pointers to the member the functions of the interface.

Parameters

read	function pointer to implementation of read()
skip	function pointer to implementation of skip()
available	function pointer to implementation of available()
close	function pointer to implementation of close()
is_closed	function pointer to implementation of is_closed()

6.14.3 Member Function Documentation

6.14.3.1 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.

6.14.3.2 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.

6.14.3.3 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.

6.14.3.4 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

buf	the buffer into which the data is read.
len	the maximum number of bytes to read.

Implements devfix::base::io::inputstream.

6.14.3.5 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

n	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

6.15 devfix::dsp::spectrogram< FloatT, N, win_fun > Struct Template Reference

```
#include <spectrogram.h>
```

Public Types

- using complex_t = std::complex< FloatT >

Public Member Functions

- spectrogram (std::size_t window_distance)
- void add (const complex_t *data, std::size_t len)
- void add (const std::vector< complex_t > &vec)
- template<std::size_t N_elems>
  void add (const std::array< complex_t, N_elems > &arr)
- std::array< complex_t, N > pop ()
- std::size_t size () const

6.15.1 Member Typedef Documentation

6.15.1.1 complex_t

```
template<typename FloatT, std::size_t N, FloatT(*)(std::size_t) win_fun>
using devfix::dsp::spectrogram< FloatT, N, win_fun >::complex_t = std::complex<FloatT>
```

6.15.2 Constructor & Destructor Documentation

6.15.2.1 spectrogram()

```
template<typename FloatT, std::size_t N, FloatT(*)(std::size_t) win_fun>
devfix::dsp::spectrogram< FloatT, N, win_fun >::spectrogram (
            std::size_t window_distance )  [inline], [explicit]
```

6.15.3 Member Function Documentation

6.15.3.1 add() [1/3]

```
template<typename FloatT, std::size_t N, FloatT(*)(std::size_t) win_fun>
void devfix::dsp::spectrogram< FloatT, N, win_fun >::add (
            const complex_t * data,
            std::size_t len )  [inline]
```

Here is the caller graph for this function:

6.15.3.2 add() [2/3]

```
template<typename FloatT, std::size_t N, FloatT(*)(std::size_t) win_fun>
void devfix::dsp::spectrogram< FloatT, N, win_fun >::add (
            const std::vector< complex_t > & vec )  [inline]
```

6.15.3.3 add() [3/3]

```
template<typename FloatT, std::size_t N, FloatT(*)(std::size_t) win_fun>
template<std::size_t N_elems>
void devfix::dsp::spectrogram< FloatT, N, win_fun >::add (
            const std::array< complex_t, N_elems > & arr )  [inline]
```

6.15.3.4 pop()

```
template<typename FloatT, std::size_t N, FloatT(*)(std::size_t) win_fun>
std::array<complex_t, N> devfix::dsp::spectrogram< FloatT, N, win_fun >::pop ( )  [inline]
```

6.15.3.5 size()

```
template<typename FloatT, std::size_t N, FloatT(*)(std::size_t) win_fun>
std::size_t devfix::dsp::spectrogram< FloatT, N, win_fun >::size ( ) const  [inline]
```

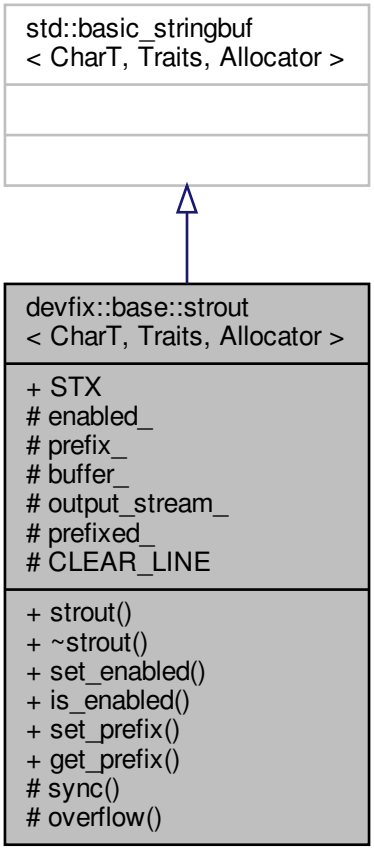
The documentation for this struct was generated from the following file:

- devfix/dsp/spectrogram.h

6.16 devfix::base::strout< CharT, Traits, Allocator > Struct Template Reference

```
#include <strout.h>
```

Inheritance diagram for devfix::base::strout< CharT, Traits, Allocator >:



Public Member Functions

- strout (std::basic_ostream< CharT > &output_stream)
- ~strout ()
- void set_enabled (bool enabled)
- bool is_enabled () const
- void set_prefix (const std::basic_string< CharT > &prefix)
- const std::basic_string< CharT > & get_prefix () const

Static Public Attributes

- static constexpr CharT STX = static_cast<CharT>('\x02')

  Start of Text, clear whole line before new text gets displayed.

Protected Types

- using int_type = typename std::basic_stringbuf< CharT, Traits, Allocator >::int_type

Protected Member Functions

- int sync () override
- int_type overflow (int_type c) override

Protected Attributes

- bool enabled_ = true
- std::basic_string< CharT > prefix_
- std::basic_stringstream< CharT > buffer_
- std::basic_ostream< CharT > & output_stream_
- bool prefixed_ = true

Static Protected Attributes

- static constexpr std::basic_string_view< CharT > CLEAR_LINE = MULTISTRING(CharT, "\033[2K\r")

6.16.1 Member Typedef Documentation

6.16.1.1 int_type

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
using devfix::base::strout< CharT, Traits, Allocator >::int_type = typename std::basic_↩
stringbuf<CharT, Traits, Allocator>::int_type  [protected]
```

6.16.2 Constructor & Destructor Documentation

6.16.2.1 strout()

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
devfix::base::strout< CharT, Traits, Allocator >::strout (
            std::basic_ostream< CharT > & output_stream )  [inline], [explicit]
```

6.16.2.2 ~strout()

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
devfix::base::strout< CharT, Traits, Allocator >::~strout ( )  [inline]
```

6.16.3 Member Function Documentation

6.16.3.1 get_prefix()

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
const std::basic_string<CharT>& devfix::base::strout< CharT, Traits, Allocator >::get_prefix
( ) const  [inline]
```

6.16.3.2 is_enabled()

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
bool devfix::base::strout< CharT, Traits, Allocator >::is_enabled ( ) const  [inline]
```

6.16.3.3 overflow()

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
int_type devfix::base::strout< CharT, Traits, Allocator >::overflow (
            int_type c )  [inline], [override], [protected]
```

6.16.3.4 set_enabled()

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
void devfix::base::strout< CharT, Traits, Allocator >::set_enabled (
            bool enabled )  [inline]
```

6.16.3.5 set_prefix()

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
void devfix::base::strout< CharT, Traits, Allocator >::set_prefix (
            const std::basic_string< CharT > & prefix )  [inline]
```

6.16.3.6 sync()

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
int devfix::base::strout< CharT, Traits, Allocator >::sync ( )  [inline], [override], [protected]
```

6.16.4 Member Data Documentation

6.16.4.1 buffer_

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
std::basic_stringstream<CharT> devfix::base::strout< CharT, Traits, Allocator >::buffer_↩
[protected]
```

6.16.4.2 CLEAR_LINE

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
constexpr std::basic_string_view<CharT> devfix::base::strout< CharT, Traits, Allocator >::C↩
LEAR_LINE = MULTISTRING(CharT, "\033[2K\r")  [static], [protected]
```

6.16.4.3 enabled_

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
bool devfix::base::strout< CharT, Traits, Allocator >::enabled_ = true  [protected]
```

6.16.4.4 output_stream_

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
std::basic_ostream<CharT>& devfix::base::strout< CharT, Traits, Allocator >::output_stream_
[protected]
```

6.16.4.5 prefix_

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
std::basic_string<CharT> devfix::base::strout< CharT, Traits, Allocator >::prefix_  [protected]
```

6.16.4.6 prefixed_

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
bool devfix::base::strout< CharT, Traits, Allocator >::prefixed_ = true  [protected]
```

6.16.4.7 STX

```
template<class CharT, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<↩
CharT>>
constexpr CharT devfix::base::strout< CharT, Traits, Allocator >::STX = static_cast<CharT>('\x02')
[static]
```

Start of Text, clear whole line before new text gets displayed.

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

- devfix/base/strout.h

6.17 devfix::base::strutil Struct Reference

`#include <strutil.h>`

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

- devfix/base/strutil.h

6.18 devfix::base::_math::Table< T, N, G, Args > Struct Template Reference

`#include <math.h>`

Public Member Functions

- constexpr Table (G gen, Args ... args)

Public Attributes

- T values [N]

6.18.1 Constructor & Destructor Documentation

6.18.1.1 Table()

```
template<typename T , std::size_t N, typename G , typename ...  Args>
constexpr devfix::base::_math::Table< T, N, G, Args >::Table (
            G gen,
            Args ...  args )  [inline]
```

6.18.2 Member Data Documentation

6.18.2.1 values

```
template<typename T , std::size_t N, typename G , typename ...  Args>
T devfix::base::_math::Table< T, N, G, Args >::values[N]
```

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

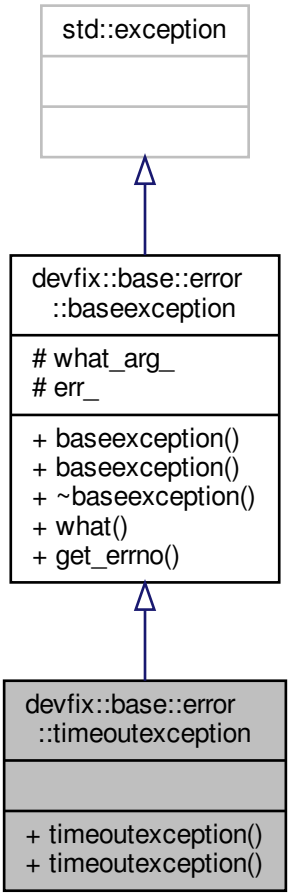
- devfix/base/math.h

6.19 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

6.19.1 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.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 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

what_arg	explanatory std::string
err	c error code (errno)

6.19.2.2 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

what_arg	explanatory c-string
err	c error code (errno)

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

- devfix/base/error/timeoutexception.h

6.20 devfix::dsp::window Struct Reference

```
#include <window.h>
```

Static Public Member Functions

- template<typename FloatT , FloatT(*)(std::size_t, std::size_t) win_fun>
  static constexpr FloatT calc_amplitude_gain (std::size_t n)
- template<typename FloatT >
  static constexpr FloatT hanning (std::size_t N, std::size_t k)
- template<typename FloatT >
  static constexpr FloatT flattop (std::size_t n, std::size_t k)

6.20.1 Member Function Documentation

6.20.1.1 calc_amplitude_gain()

```
template<typename FloatT , FloatT(*)(std::size_t, std::size_t) win_fun>
static constexpr FloatT devfix::dsp::window::calc_amplitude_gain (
            std::size_t n )  [inline], [static]
```

6.20.1.2 flattop()

```
template<typename FloatT >
static constexpr FloatT devfix::dsp::window::flattop (
            std::size_t n,
            std::size_t k )  [inline], [static]
```

6.20.1.3 hanning()

```
template<typename FloatT >
static constexpr FloatT devfix::dsp::window::hanning (
            std::size_t N,
            std::size_t k )  [inline], [static]
```

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

- devfix/dsp/window.h

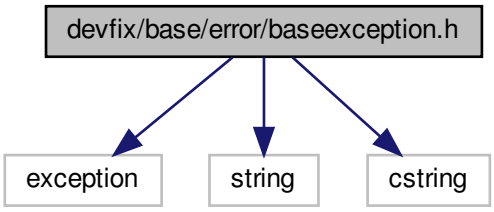
Chapter 7

File Documentation

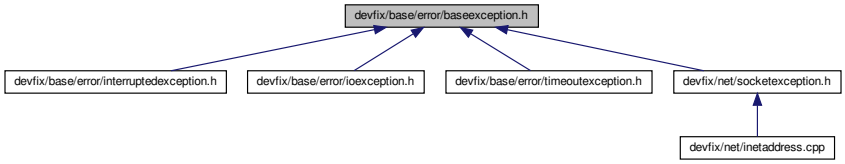
7.1 devfix/base/error/baseexception.h File Reference

```
#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))

7.1.1 Macro Definition Documentation

7.1.1.1 exception_guard

```
#define exception_guard(
              err,
              exception_class ) exception_guard_m(err, exception_class, std::strerror(errno))
```

7.1.1.2 exception_guard_m

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

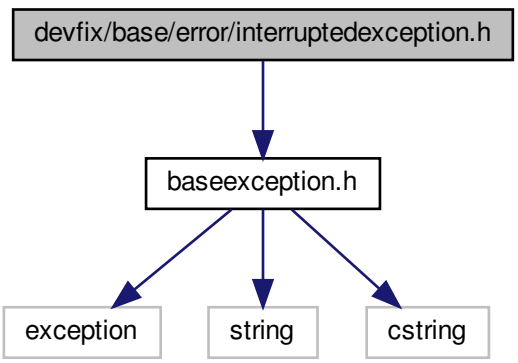
Value:

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

7.2 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

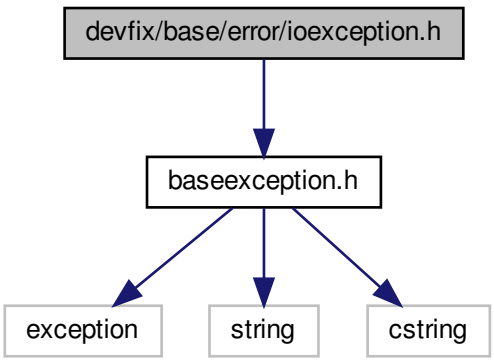
- devfix::base::error

  Namespace for general errors like timeouts or io failures.

7.3 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.

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

Namespaces

- devfix::base::error

  Namespace for general errors like timeouts or io failures.

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

Namespaces

- devfix::base::io

  Namespace for io tool, for instance streams.

7.6 devfix/base/namespace.h File Reference

Namespaces

- devfix
- devfix::base

  Root namespace of devfix base library.

7.7 devfix/net/namespace.h File Reference

Namespaces

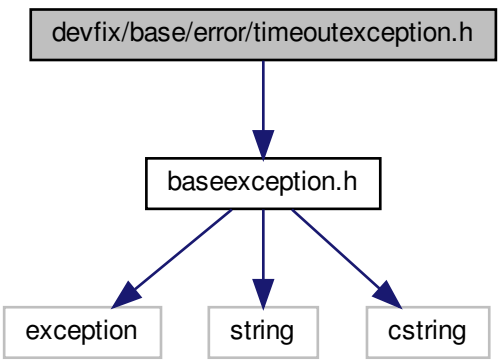
- devfix
- devfix::net

  Root namespace of devfix network library.

7.8 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.

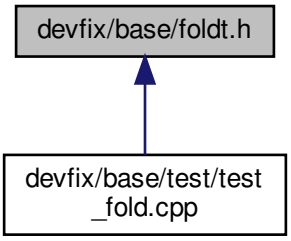
Namespaces

- devfix::base::error

  Namespace for general errors like timeouts or io failures.

7.9 devfix/base/foldt.h File Reference

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



Functions

- template<typename Y , typename F , typename , typename ... Xs>
  Y foldt (Y n, F f, Xs... xs)

  Uses a given combining operation, recombines the results of recursively processing its constituent parts, building up a return value.
- template<typename Y , typename F >
  Y foldt (Y n, F)
- template<typename Y , typename F , typename X , typename ... Xs>
  Y foldt (Y n, F f, X x, Xs... xs)

7.9.1 Function Documentation

7.9.1.1 foldt() [1/3]

```
template<typename Y , typename F , typename , typename ...  Xs>
Y foldt (
            Y n,
            F f,
            Xs...  xs )
```

Uses a given combining operation, recombines the results of recursively processing its constituent parts, building up a return value.

Template Parameters

Y	return type
F	type of combining function
Xs	type argument parameter pack

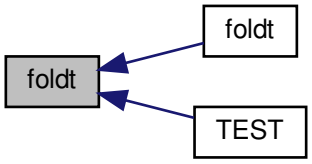
Parameters

n	neutral element
f	combining function
xs	argument parameter pack

Returns

result

Here is the caller graph for this function:



7.9.1.2 foldt() [2/3]

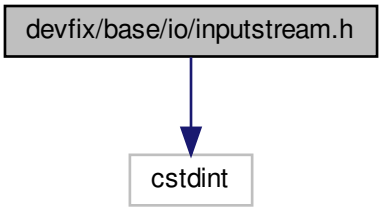
```
template<typename Y , typename F >
Y foldt (
             Y n,
             F  )
```

7.9.1.3 foldt() [3/3]

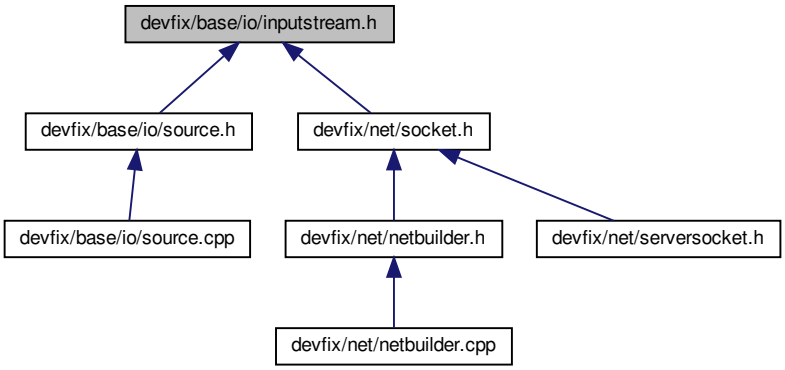
```
template<typename Y , typename F , typename X , typename ...  Xs>
Y foldt (
             Y n,
             F f,
             X x,
             Xs...  xs )
```

7.10 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.

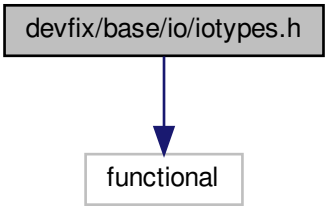
Namespaces

- devfix::base::io

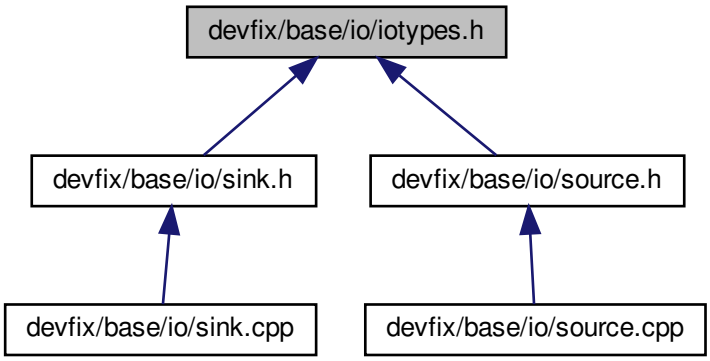
  Namespace for io tool, for instance streams.

7.11 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

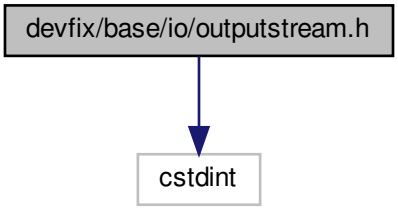
Variables

- const close_t devfix::base::io::DEFAULT_CLOSE
- const is_closed_t devfix::base::io::DEFAULT_IS_CLOSED

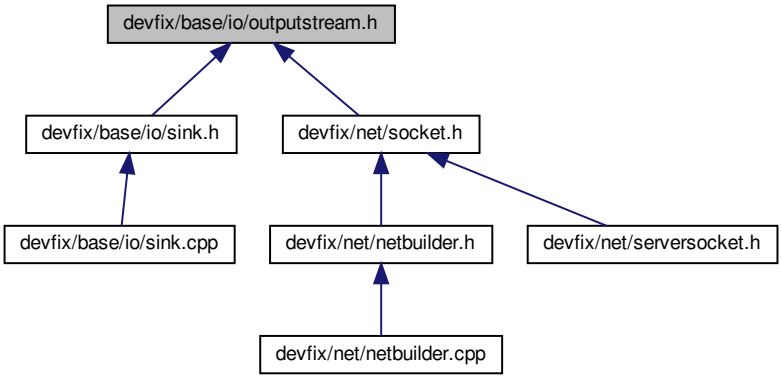
7.12 devfix/base/io/outputstream.h File Reference

```
#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

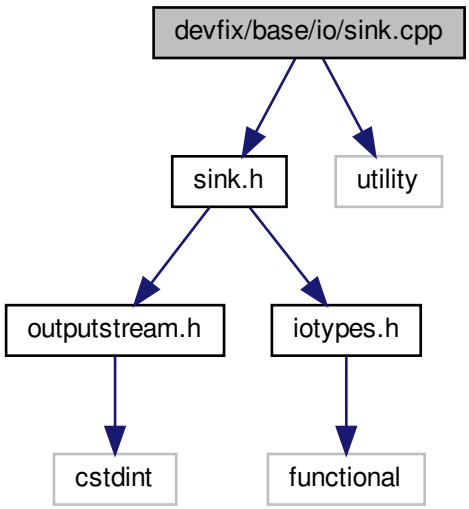
- devfix::base::io

  Namespace for io tool, for instance streams.

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

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

Include dependency graph for sink.cpp:



Namespaces

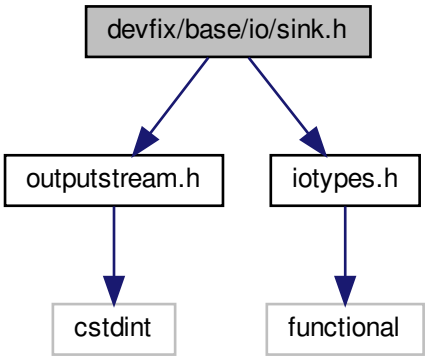
- devfix::base::io

  Namespace for io tool, for instance streams.

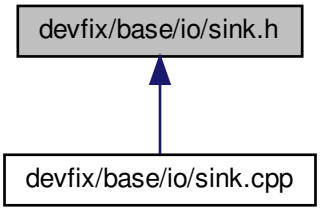
7.14 devfix/base/io/sink.h File Reference

```
#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

  Adapter class to create an outputstream from function pointers.

Namespaces

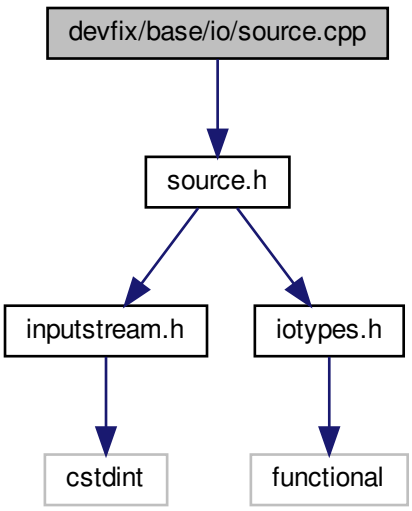
- devfix::base::io

  Namespace for io tool, for instance streams.

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

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

Include dependency graph for source.cpp:



Namespaces

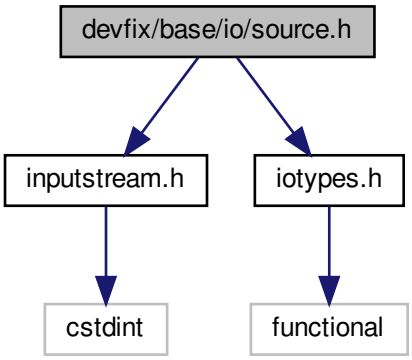
- devfix::base::io

  Namespace for io tool, for instance streams.

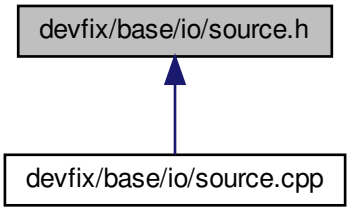
7.16 devfix/base/io/source.h File Reference

```
#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

  Adapter class to create an inputstream from function pointers.

Namespaces

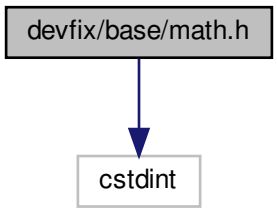
- devfix::base::io

  Namespace for io tool, for instance streams.

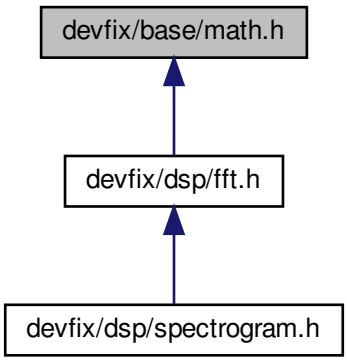
7.17 devfix/base/math.h File Reference

```
#include <cstdint>
```

Include dependency graph for math.h:



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



Classes

- struct devfix::base::_math::Table< T, N, G, Args >
- struct devfix::base::math

Namespaces

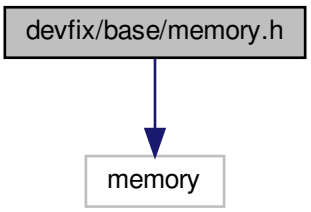
- devfix::base

  Root namespace of devfix base library.
- devfix::base::_math

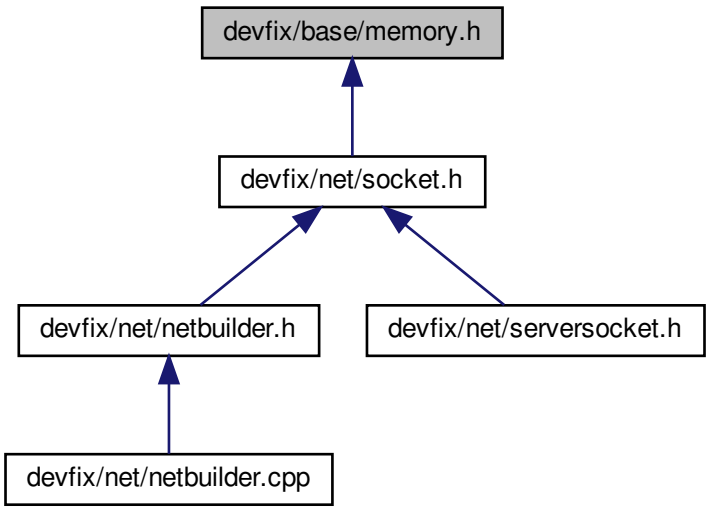
7.18 devfix/base/memory.h File Reference

```
#include <memory>
```

Include dependency graph for memory.h:



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



Namespaces

- devfix::base

  Root namespace of devfix base library.

Typedefs

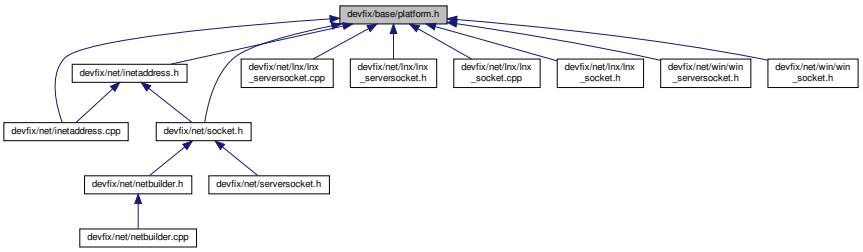
- template<class T >
  using devfix::base::up = std::unique_ptr< T >

  Alias for std::unique_ptr.
- template<class T >
  using devfix::base::sp = std::shared_ptr< T >

  Alias for std::shared_ptr.

7.19 devfix/base/platform.h File Reference

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



Macros

- #define PLATFORM_LINUX 0
- #define PLATFORM_WINDOWS 0
- #define ERROR_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]) + "\""
- #define NOT_USED(x) static_cast<void>(x)

7.19.1 Macro Definition Documentation

7.19.1.1 __FILENAME__

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

7.19.1.2 ERROR_PLATFORM_UNSUPPORTED

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

7.19.1.3 NOT_USED

```
#define NOT_USED(
              x ) static_cast<void>(x)
```

7.19.1.4 PLATFORM_LINUX

```
#define PLATFORM_LINUX 0
```

7.19.1.5 PLATFORM_WINDOWS

```
#define PLATFORM_WINDOWS 0
```

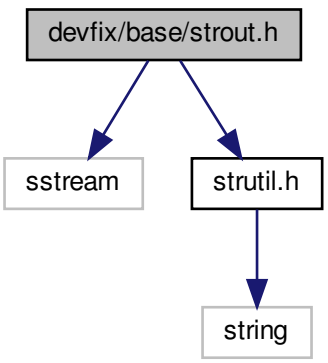
7.19.1.6 SOURCE_LINE

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

7.20 devfix/base/strout.h File Reference

```
#include <sstream>
#include "strutil.h"
```

Include dependency graph for strout.h:



Classes

- struct devfix::base::strout< CharT, Traits, Allocator >

Namespaces

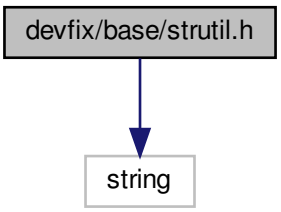
- devfix::base

  Root namespace of devfix base library.

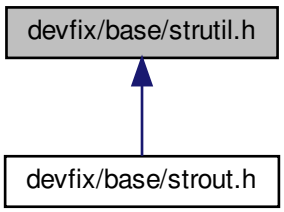
7.21 devfix/base/strutil.h File Reference

```
#include <string>
```

Include dependency graph for strutil.h:



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



Classes

- struct devfix::base::strutil

Namespaces

- devfix::base

  Root namespace of devfix base library.

Macros

- #define MULTISTRING(CharT, str) devfix::base::get_from_multistring<CharT>(str, L##str)

Functions

- template<typename T >
  constexpr const T * devfix::base::get_from_multistring (const char *str, const wchar_t *wstr)
- template<>
  constexpr const char * devfix::base::get_from_multistring< char > (const char *str, const wchar_t *wstr)
- template<>
  constexpr const wchar_t * devfix::base::get_from_multistring< wchar_t > (const char *str, const wchar_t *wstr)

7.21.1 Macro Definition Documentation

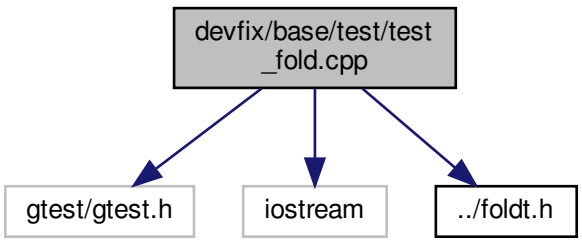
7.21.1.1 MULTISTRING

```
#define MULTISTRING(
              CharT,
              str ) devfix::base::get_from_multistring<CharT>(str, L##str)
```

7.22 devfix/base/test/test_fold.cpp File Reference

```
#include <gtest/gtest.h>
#include <iostream>
#include "../foldt.h"
```

Include dependency graph for test_fold.cpp:



Functions

- template<typename T >
  constexpr T add (T a, T b)
- template<typename T , T d>
  constexpr T incIfDiv (T a, T b)
- TEST (Fold, Template)

7.22.1 Function Documentation

7.22.1.1 add()

```
template<typename T >
constexpr T add (
            T a,
            T b )
```

7.22.1.2 incIfDiv()

```
template<typename T , T d>
constexpr T incIfDiv (
            T a,
            T b )
```

7.22.1.3 TEST()

```
TEST (
            Fold ,
            Template  )
```

7.23 devfix/base/test/test_math.cpp File Reference

7.24 devfix/base/test/test_strout.cpp File Reference

7.25 devfix/dsp/cmake-build-debug/CMakeFiles/3.15.3/CompilerIdC/CMakeCCompilerId.c 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 C_DIALECT

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

7.25.1 Macro Definition Documentation

7.25.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

7.25.1.2 C_DIALECT

```
#define C_DIALECT
```

7.25.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

7.25.1.4 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))
```

7.25.1.5 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))
```

7.25.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

7.25.1.7 STRINGIFY

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

7.25.1.8 STRINGIFY_HELPER

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

7.25.2 Function Documentation

7.25.2.1 main()

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

7.25.3 Variable Documentation

7.25.3.1 info_arch

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

7.25.3.2 info_compiler

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

7.25.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
=
  "INFO" ":" "dialect_default[" C_DIALECT "]"
```

7.25.3.4 info_platform

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

7.26 devfix/net/cmake-build-debug/CMakeFiles/3.15.3/CompilerIdC/CMakeCCompilerId.c 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 C_DIALECT

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

7.26.1 Macro Definition Documentation

7.26.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

7.26.1.2 C_DIALECT

```
#define C_DIALECT
```

7.26.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

7.26.1.4 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))
```

7.26.1.5 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))
```

7.26.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

7.26.1.7 STRINGIFY

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

7.26.1.8 STRINGIFY_HELPER

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

7.26.2 Function Documentation

7.26.2.1 main()

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

7.26.3 Variable Documentation

7.26.3.1 info_arch

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

7.26.3.2 info_compiler

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

7.26.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
=
  "INFO" ":" "dialect_default[" C_DIALECT "]"
```

7.26.3.4 info_platform

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

7.27 devfix/dsp/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

7.27.1 Macro Definition Documentation

7.27.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

7.27.1.2 COMPILER_ID

```
#define COMPILER_ID ""
```

7.27.1.3 CXX_STD

```
#define CXX_STD __cplusplus
```

7.27.1.4 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))
```

7.27.1.5 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))
```

7.27.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

7.27.1.7 STRINGIFY

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

7.27.1.8 STRINGIFY_HELPER

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

7.27.2 Function Documentation

7.27.2.1 main()

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

7.27.3 Variable Documentation

7.27.3.1 info_arch

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

7.27.3.2 info_compiler

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

7.27.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

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


  "98"
"]"
```

7.27.3.4 info_platform

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

7.28 devfix/net/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

7.28.1 Macro Definition Documentation

7.28.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

7.28.1.2 COMPILER_ID

```
#define COMPILER_ID ""
```

7.28.1.3 CXX_STD

```
#define CXX_STD __cplusplus
```

7.28.1.4 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))
```

7.28.1.5 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))
```

7.28.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

7.28.1.7 STRINGIFY

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

7.28.1.8 STRINGIFY_HELPER

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

7.28.2 Function Documentation

7.28.2.1 main()

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

7.28.3 Variable Documentation

7.28.3.1 info_arch

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

7.28.3.2 info_compiler

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

7.28.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

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

  "98"
"]"
```

7.28.3.4 info_platform

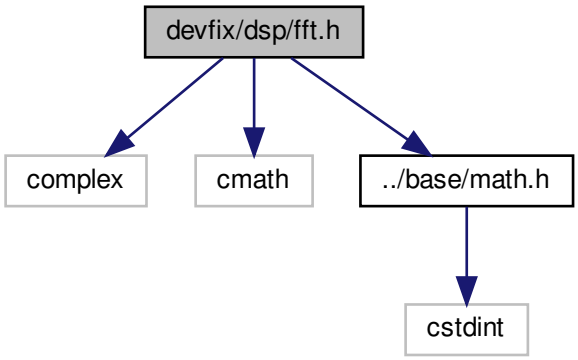
```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

7.29 devfix/dsp/fft.h File Reference

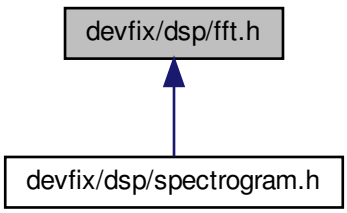
```
#include <complex>
#include <cmath>
```

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

Include dependency graph for fft.h:



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



Classes

- struct devfix::dsp::fft

Namespaces

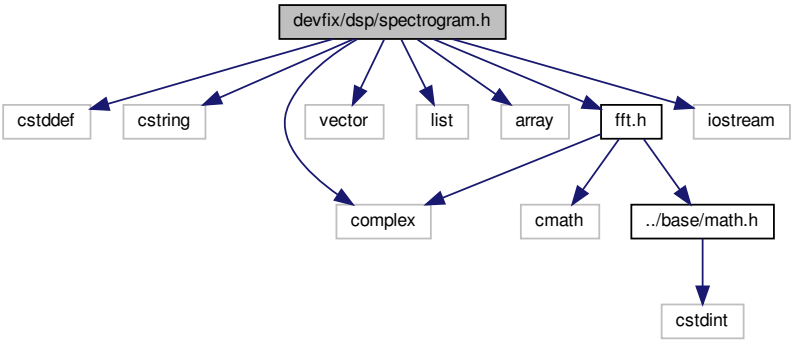
- devfix::dsp

7.30 devfix/dsp/spectrogram.h File Reference

```
#include <cstddef>
#include <cstring>
```

```
#include <complex>
#include <vector>
#include <list>
#include <array>
#include "fft.h"
#include <iostream>
```

Include dependency graph for spectrogram.h:



Classes

- struct devfix::dsp::spectrogram< FloatT, N, win_fun >

Namespaces

- devfix::dsp

7.31 devfix/dsp/test/test_fft.cpp File Reference

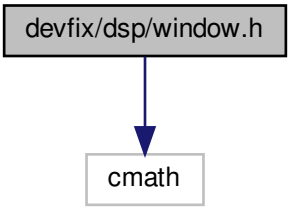
7.32 devfix/dsp/test/test_spectrogram.cpp File Reference

7.33 devfix/dsp/test/test_window.cpp File Reference

7.34 devfix/dsp/window.h File Reference

```
#include <cmath>
```

Include dependency graph for window.h:



Classes

- struct devfix::dsp::window

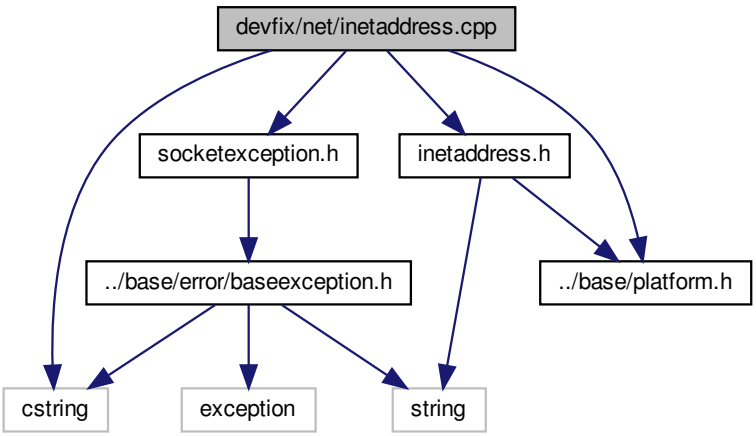
Namespaces

- devfix::dsp

7.35 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

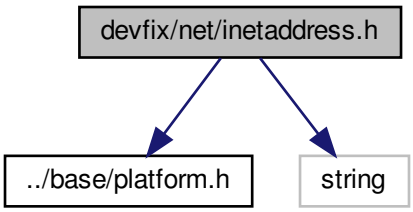
- devfix::net

  Root namespace of devfix network library.

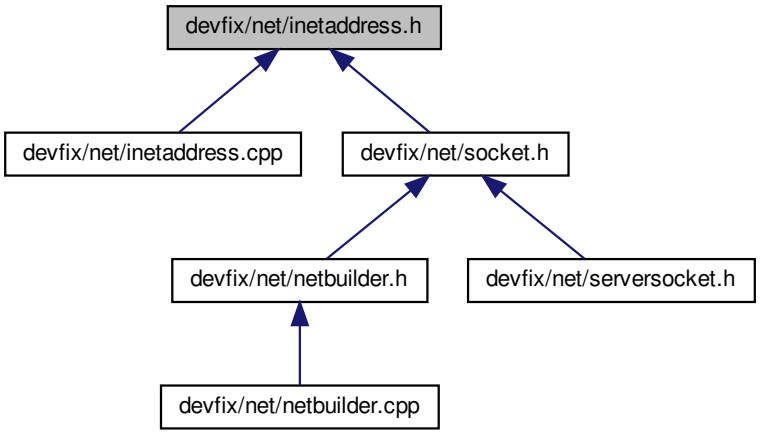
7.36 devfix/net/inetaddress.h File Reference

```
#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

  Class for management and conversion of internet addresses.

Namespaces

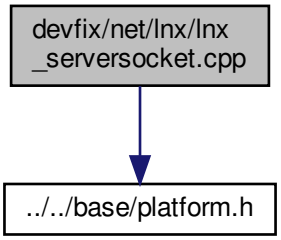
- devfix::net

  Root namespace of devfix network library.

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

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

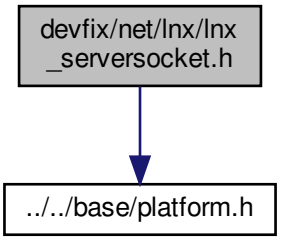
Include dependency graph for lnx_serversocket.cpp:



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

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

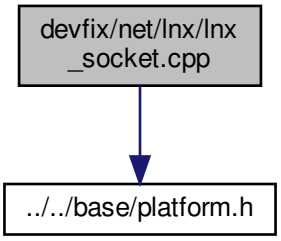
Include dependency graph for lnx_serversocket.h:



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

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

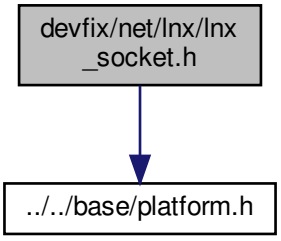
Include dependency graph for lnx_socket.cpp:



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

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

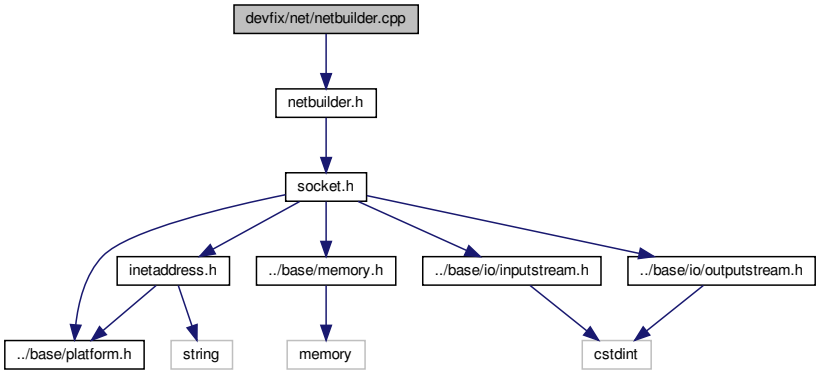
Include dependency graph for lnx_socket.h:



7.41 devfix/net/netbuilder.cpp File Reference

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

Include dependency graph for netbuilder.cpp:



Namespaces

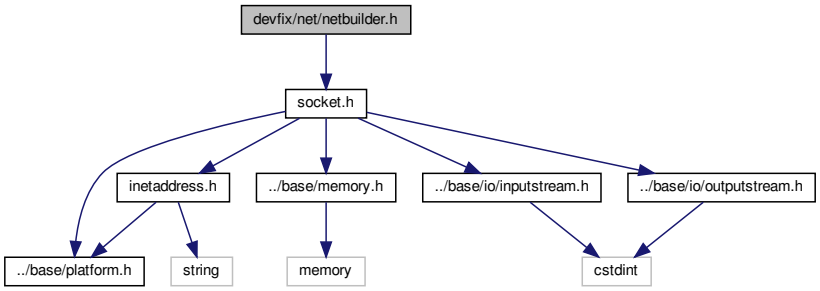
- devfix::net

  Root namespace of devfix network library.

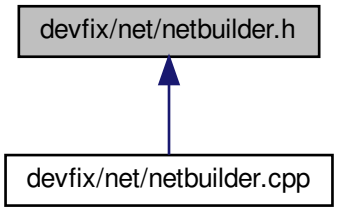
7.42 devfix/net/netbuilder.h File Reference

```
#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

  Builder class for platform independent instantiation.

Namespaces

- devfix::net

  Root namespace of devfix network library.

Variables

- PLATPLATFORM_UNSUPPORTED

7.42.1 Variable Documentation

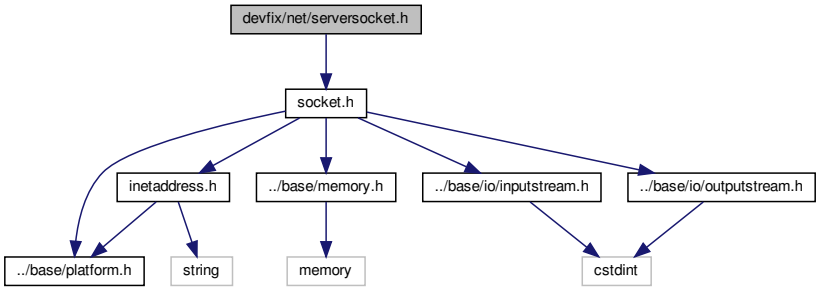
7.42.1.1 PLATPLATFORM_UNSUPPORTED

`PLATPLATFORM_UNSUPPORTED`

7.43 devfix/net/serversocket.h File Reference

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

Include dependency graph for serversocket.h:



Classes

- struct devfix::net::serversocket

  This class implements server sockets.

Namespaces

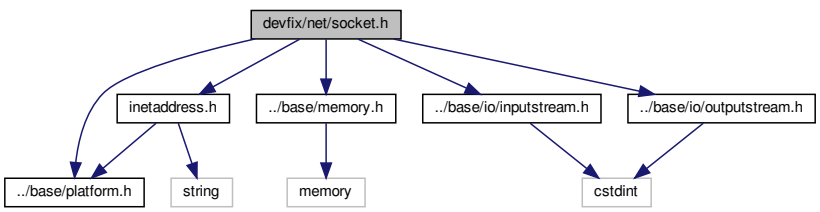
- devfix::net

  Root namespace of devfix network library.

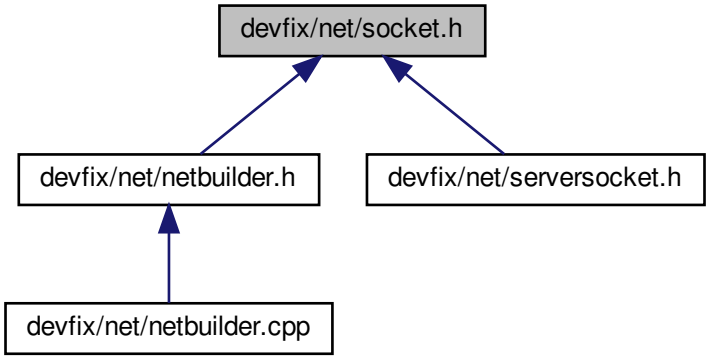
7.44 devfix/net/socket.h File Reference

```
#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

  This class implements client sockets (also called just "sockets").

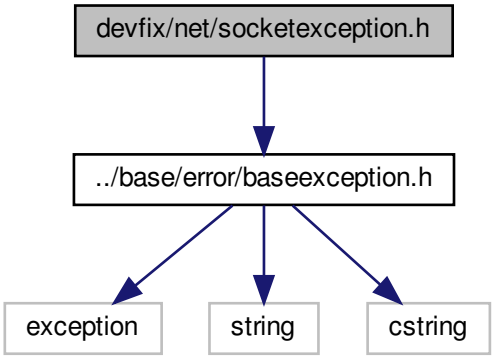
Namespaces

- devfix::net

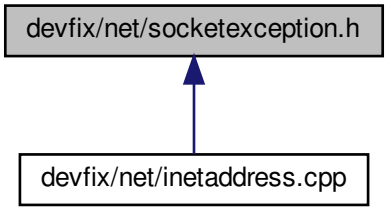
  Root namespace of devfix network library.

7.45 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

  Root namespace of devfix network library.

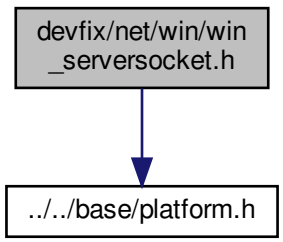
7.46 devfix/net/test/test_inetaddress.cpp File Reference

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

7.48 devfix/net/win/win_serversocket.h File Reference

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

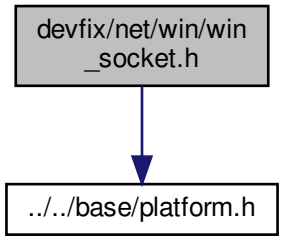
Include dependency graph for win_serversocket.h:



7.49 devfix/net/win/win_socket.h File Reference

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

Include dependency graph for win_socket.h:



Index