QNX Foundation Classes

Version 0.8
10/2/2007 8:41:00 AM

Table of Contents

QNX Foundation Classes Namespace Index

QNX Foundation Classes Namespace List

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

QNX Foundation Classes Class Index

QNX Foundation Classes Class List

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

QNX Foundation Classes Namespace Documentation

PosixAPI Namespace Reference

This namespace contains Posix API classes.

Classes

- class **Timeout**
  Timeout *base class.*

- class **PosixFailure**
  PosixFailure *base class.*

- class **PosixInitFailure**
  PosixInitFailure *exception.*

- class **GetTimeFailure**
  GetTimeFailure *exception.*

- class **Rootable**
  Interface rootable.

- class **Lockable**
  Interface: ***Lockable****.*

- class **Signalable**
  Interface: ***Signalable****.*

- class **Broadcastable**
  Interface: ***Broadcastable****.*

- class **TimerCreateFailure**
  TimerCreateFailure *class.*

- class **TimerDestroyFailure**
  TimerDestroyFailure *class.*

- class **TimerSetFailure**
  TimerSetFailure *class.*

- class **Timer**
  Posix ***Timer*** *class.*

- class **TimerRealtime**
  Posix Realtime ***Timer*** *class.*

- class **MutexFailure**
  MutexFailure *base class.*

- class **MutexLockFailure**
  MutexLockFailure *class.*

- class **MutexUnlockFailure**
  MutexUnlockFailure *class.*

- class **MutexTrylockFailure**
  MutexTrylockFailure *class.*

- class **MutexTimedLockFailure**
  MutexTimedLockFailure *class.*

- class **MutexTimedLockTimeout**

MutexTimedLockTimeout *class.*

- class **CondVarFailure**
  CondVarFailure *base class.*

- class **WaitFailure**
  WaitFailure *class.*

- class **TimedWaitTimeout**
  TimedWaitTimeout *class.*

- class **BroadcastFailure**
  Broadcast failure class.

- class **MutexAttr**
  Posix ***Mutex*** *attributes.*

- class **Mutex**
  Mutex *class.*

- class **CondVar**
  CondVar *class.*

- class **ThreadJoinTimeout**
  Thread *join timeout class.*

- class **ThreadStartFailure**
  Thread *class.*

- class **Thread**
  Thread *class.*

Enumerations

- enum **MutexAttrProtocol**
  Mutex protocol types.

- enum **MutexAttrProtocol**
  Mutex protocol types.

Detailed Description

This namespace contains Posix API classes.

Warning:
The classes are not posix, the back and bindings are (i.e. the class is portable, but clients of these classes are not).

Author:
Rennie Allen `rennieallen@gmail.com`

QNXAPI Namespace Reference

This namespace contains QNX specific API classes.

Classes

- class **UsageFailure**
  UsageFailure *class.*

- class **Decomposable**
  Decomposable *class.*

- class **Communicable**
  Communicable *class.*

- class **IoSelector**
  IoSelector *class.*

- class **SimpleVect**
  Simple vector class.

- class **VectLoader**
  Simple vector loader visitor.

- class **SimpleVectLoader**
  Default vector loader visitor (for ***SimpleVect****).*

- class **MsgReadFailure**
  Message could not be read.

- class **MsgSendFailure**
  Message could not be sent.

- class **MsgSendPulseFailure**
  Pulse could not be sent.

- class **MsgReplyFailure**
  Message could not be replied to.

- class **ChannelFailure**
  Base class for channel failures.

- class **ChannelCreateFailure**
  Channel *could not be created.*

- class **ChannelConnectAttachFailure**
  Channel *could not be attached to.*

- class **ChannelReceiveFailure**
  Channel *could not receive a message.*

- class **ChannelSendPulseFailure**
  A pulse could not be sent on a channel.

- class **NameConnectionFailure**
  A ***NameConnection*** *failure occured.*

- class **Reception**
  Reception *class. Holds a receive context.*

- class **Connection**
  Connection *class. Represents a connection to a channel.*

- class **ReplyVector**
  ReplyVector *class.*

- class **MessageVector**
  Message vector class.

- class **Channel**
  A ***Channel*** *is a connection point through which IPC messages flow.*

- class **NameChannel**
  NameChannel *class.*

- class **NameConnection**
  NameConnection *class.*

- class **BinaryLockFailure**
  BinaryLockFailure *class.*

- class **RdWrLockFailure**
  RdWrLockFailure *class.*

- class **ScopedLock**
  Scoped lock (exclusive).

- class **ScopedReservation**
  Scoped reservation (non-exclusive lock).

- class **BinaryLock**
  Binary lock (obtain two lockable entities or none).

- class **RwLockAttr**
  Read/Write lock attributes.

- class **RdWrLock**
  Read/Write lock.

- class **ResmgrException**
  Resource Manager non-fatal exception base class.

- class **ResmgrFatalException**
  Resource Manager fatal exception base class.

- class **DeviceFatalException**
  Resource Manager Device fatal exception base class.

- class **SymLinkException**
  SymLinkException *(thrown when a redirect is encountered deep in a path).*

- class **Resmgr**
  *The **Resmgr** class.*

- class **SigEvent**
  *The **SigEvent** class.*

- class **DeliverableSigEvent**
  *Deliverable **SigEvent**.*

- class **SigEventPulse**
  ***SigEvent** pulse variant.*

- class **ControlDevice**
  *The **Sysmon ControlDevice** class.*

- class **Sysmon**
  Provides a fault tolerant process manager.

- class **Thread**
  ***Thread** class - QNX extensions.*

- class **ThreadPool**
  Provides a thread pool manager intended to be utilized via composition.

- class **PoolContext**
  ***PoolContext** class.*

- class **TimerBankFailure**
  ***TimerBankFailure** class.*

- class **TimerBank**
  TimerBank *class.*

- class **TimerTreeFailure**
  Exception encountered during setting of a tree timer.

- class **TimerTree**
  Manages a tree of 100 timers driven by a single OS timer.

Typedefs

- typedef TMP::var::variant< **SimpleVect** > **iovItem**
  Default message type handled.

- typedef **ChannelFailure NameChannelFailure**
  Base class for name channel failures.

- typedef **ChannelCreateFailure NameChannelCreateFailure**
  Name channel could not be created.

- typedef **ChannelConnectAttachFailure NameChannelConnectAttachFailure**
  Name channel could not be attached to.

- typedef **ChannelReceiveFailure NameChannelReceiveFailure**
  Name channel could not receive a message.

- typedef **ChannelSendPulseFailure NameChannelSendPulseFailure**
  A pulse could not be sent on a name channel.

- typedef **ResmgrException NodeException**
  Resmgr *exceptions and NodeExceptions equivalent.*

- typedef **ResmgrException OcbException**
  Resmgr *exceptions and OcbExceptions equivalent.*

- typedef TMP::var::variant< **SimpleVect** > **iovItem**
  Default message type handled.

- typedef **ChannelFailure NameChannelFailure**
  Base class for name channel failures.

- typedef **ChannelCreateFailure NameChannelCreateFailure**
  Name channel could not be created.

- typedef **ChannelConnectAttachFailure NameChannelConnectAttachFailure**
  Name channel could not be attached to.

- typedef **ChannelReceiveFailure NameChannelReceiveFailure**
  Name channel could not receive a message.

- typedef **ChannelSendPulseFailure NameChannelSendPulseFailure**
  A pulse could not be sent on a name channel.

- typedef **ResmgrException NodeException**
  Resmgr *exceptions and NodeExceptions equivalent.*

- typedef **ResmgrException OcbException**
  Resmgr *exceptions and OcbExceptions equivalent.*

Functions

- int **CheckAccess** (resmgr_context_t *ctp, unsigned access, const IOFUNC_ATTR_T *attr, struct _client_info &client_info) throw ()
  Check access rights.

- int **CheckDirAccess** (resmgr_context_t *ctp, const IOFUNC_ATTR_T *attr, struct _client_info &client_info) throw ()
  Check access rights.

- int **ReadVerify** (resmgr_context_t *ctp, io_read_t *msg, **Resmgr::Ocb** &**ocb**) throw ()
  Check access rights.

- int **WriteVerify** (resmgr_context_t *ctp, io_write_t *msg, **Resmgr::Ocb** &**ocb**) throw ()
  Check access rights.

- bool **CheckXtype** (uint32_t xtype) throw ()
  Check the X type.

- int **CreateNode** (resmgr_context_t *ctp, IOFUNC_ATTR_T *attr, mode_t &mode, iofunc_mount_t *mount, struct _client_info &info) throw ()
  Create a resource manager mount point.

- std::string **BreakPath** (std::string &component, std::string const &path, const char *const delimiters="/")
  Break a path into 2 parts, a component (which is the current head of the path), and a remainder, which is everything below the head of the path.

Detailed Description

This namespace contains QNX specific API classes.

Author:

Rennie Allen `renniеallen@gmail.com`

Typedef Documentation

typedef ChannelConnectAttachFailure QNXAPI::NameChannelConnectAttachFailure

Name channel could not be attached to.

Thrown when an error occurs as the result of an attempt to connect to a name channel.

Definition at line 277 of file ipc.typedef ChannelConnectAttachFailure QNXAPI::NameChannelConnectAttachFailure

Name channel could not be attached to.

Thrown when an error occurs as the result of an attempt to connect to a name channel.

Definition at line 277 of file ipc.svn-base.typedef ChannelCreateFailure QNXAPI::NameChannelCreateFailure

Name channel could not be created.

Thrown when an error occurs as the result of a name channel creation attempt.

Definition at line 270 of file ipc.typedef ChannelCreateFailure QNXAPI::NameChannelCreateFailure

Name channel could not be created.

Thrown when an error occurs as the result of a name channel creation attempt.

Definition at line 270 of file ipc.svn-base.typedef ChannelReceiveFailure QNXAPI::NameChannelReceiveFailure

Name channel could not receive a message.

Thrown when an error occurs while a name channel is attempting a receive.

Definition at line 284 of file ipc.typedef ChannelReceiveFailure QNXAPI::NameChannelReceiveFailure

Name channel could not receive a message.

Thrown when an error occurs while a name channel is attempting a receive.

Definition at line 284 of file ipc.svn-base.typedef ChannelSendPulseFailure QNXAPI::NameChannelSendPulseFailure

A pulse could not be sent on a name channel.

Thrown when an error occurs as the result of a name channel sending a pulse.

Definition at line 291 of file ipc.typedef ChannelSendPulseFailure QNXAPI::NameChannelSendPulseFailure

A pulse could not be sent on a name channel.

Thrown when an error occurs as the result of a name channel sending a pulse.

Definition at line 291 of file ipc.svn-base.

Function Documentation

std::string QNXAPI::BreakPath (std::string &* **component**, *std::string const &* **path**, *const char *const* **delimiters** = *"/")

Break a path into 2 parts, a component (which is the current head of the path), and a remainder, which is everything below the head of the path.

Parameters:

component reference to string that will contain the component part.
path the path to be broken.
delimiters token that delimits path (if other than the default "/").

Returns:

string containing the remainder of path (can be passed into another call to **BreakPath()**.

int QNXAPI::CheckAccess (resmgr_context_t *** **ctp**, *unsigned* **access**, *const IOFUNC_ATTR_T *** **attr**, *struct _client_info &* **client_info**) *throw ()

Check access rights.

Verify that the currently registered client info (registered to the Ocb) provides sufficient credentials to access the resource described by the attributes. This method provides default semantics identical to that described in the QNX help docs for iofunc_check_access, and may be overridden to provide different semantics.

Parameters:

ctp resource manager context pointer.
access access requested.
attr attributes of resource in question.
client_info client credentials.

Returns:

error code from errno.h

int QNXAPI::CheckDirAccess (resmgr_context_t *** **ctp**, *const IOFUNC_ATTR_T *** **attr**, *struct _client_info &* **client_info**) *throw ()

Check access rights.

Verify that the currently registered client info (registered to the Ocb) provides sufficient credentials to scan the directory represented by the attributes structure (attr).

Parameters:

ctp resource manager context pointer.
attr attributes of resource in question.
client_info client credentials.

Returns:

error code from errno.h

bool QNXAPI::CheckXtype (uint32_t xtype) throw ()

Check the X type.

Parameters:

xtype the x type.

Returns:

true always

int QNXAPI::CreateNode (resmgr_context_t * ctp, IOFUNC_ATTR_T * attr, mode_t & mode, iofunc_mount_t * mount, struct _client_info & info) throw ()

Create a resource manager mount point.

Parameters:

ctp resource manager context pointer.
attr parent of mount points attributes.
mode mode of mount point.
mount pointer to a QNX 'C' library iofunc_mount_t.
info client credentials.

Returns:

error code from errno.h

int QNXAPI::ReadVerify (resmgr_context_t * ctp, io_read_t * msg, Resmgr::Ocb & ocb) throw ()

Check access rights.

Verify that the currently registered client info (registered to the Ocb) provides sufficient credentials to read the resource described by the attributes.

Parameters:

ctp resource manager context pointer.
msg pointer to a QNX 'C' library io_read_t.
ocb ocb containing credentials.

Returns:

error code from errno.h

int QNXAPI::WriteVerify (resmgr_context_t * ctp, io_write_t * msg, Resmgr::Ocb & ocb) throw ()

Check access rights.

Verify that the currently registered client info (registered to the Ocb) provides sufficient credentials to write the resource described by the attributes.

Parameters:

ctp resource manager context pointer.
msg pointer to a QNX 'C' library io_write_t.
ocb ocb containing credentials.

Returns:

error code from errno.h

TMP Namespace Reference

This namespace contains Template Meta-Progamming classes.

Detailed Description

This namespace contains Template Meta-Progamming classes.

Author:

Eugene Gladyshev

UnixAPI Namespace Reference

This namespace contains Unix API classes.

Namespaces

- namespace **Syslog**
  Reserved for system log classes.

-

Detailed Description

This namespace contains Unix API classes.

Author:

Rennie Allen `rennieallen@gmail.com`

UnixAPI::Syslog Namespace Reference

Reserved for system log classes.

Classes

- class **StreamBuffer**
  *The **Syslog StreamBuffer** class.*

- class **LogStream**
  *The **Syslog LogStream** class a logging adapter. Data from the stream buffer is supplied to the logging system adapter. By default to logging system adapters are provided:.*

Enumerations

- enum **LogOption**
  Log options.

- enum **Facility**
  Log facilities.

- enum **Priority**
  Log priority.

- enum **LogOption**
  Log options.

- enum **Facility**
  Log facilities.

- enum **Priority**
  Log priority.

Functions

- template<class logAdapterT> void **do_loglevel** (std::ios_base &s, **Priority** n)
- void **do_format** (std::ios_base &s, const char *fmt)

Variables

- const int **BufferSize** = 1024
  Default 1KB buffer.

- const int **BufferSize** = 1024
  Default 1KB buffer.

Detailed Description

Reserved for system log classes.

Author:

Rennie Allen `rennieallen@gmail.com`

Function Documentation

void UnixAPI::Syslog::do_format (std::ios_base & s, const char * fmt)

Format manipulator interface.

Implements printf() style format control for numbers.

Parameters:

s Stream
fmt printf() style format control string.

template<class logAdapterT> void UnixAPI::Syslog::do_loglevel (std::ios_base & s, Priority n)

Loglevel manipulator interface.

Sets the priority level of the stream.

Parameters:

s Stream
n (see **UnixAPI::Syslog::Priority**).

Definition at line 201 of file syslog.svn-base.

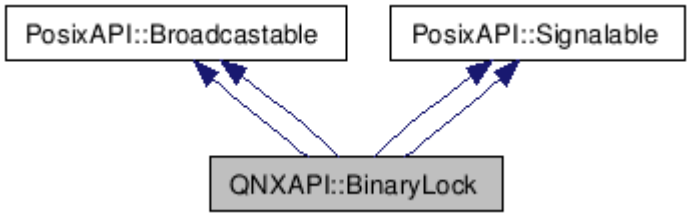
QNX Foundation Classes Class Documentation

QNXAPI::BinaryLock Class Reference

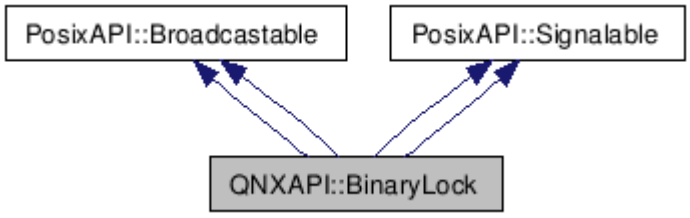
Binary lock (obtain two lockable entities or none).

Inherits **PosixAPI::Broadcastable**, **PosixAPI::Signalable**, **PosixAPI::Broadcastable**, and **PosixAPI::Signalable**.

Inheritance diagram for QNXAPI::BinaryLock:



Collaboration diagram for QNXAPI::BinaryLock:



Detailed Description

Binary lock (obtain two lockable entities or none).

Definition at line 121 of file lock.svn-base.

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

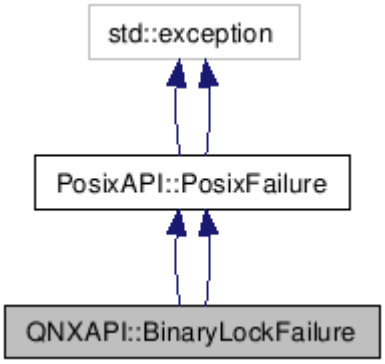
- include/qfc/.svn/text-base/lock.svn-base
- include/qfc/lock

QNXAPI::BinaryLockFailure Class Reference

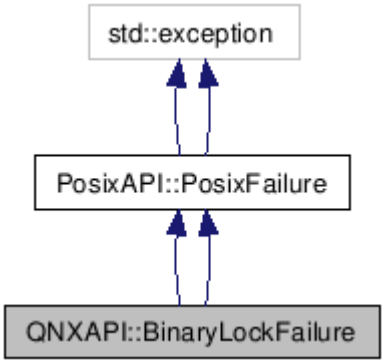
BinaryLockFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::BinaryLockFailure:



Collaboration diagram for QNXAPI::BinaryLockFailure:



Public Member Functions

- **BinaryLockFailure** (int err)
- **BinaryLockFailure** (int err)

Detailed Description

BinaryLockFailure class.

Exception encountered during binary lock operations.

Definition at line 32 of file lock.svn-base.

Constructor & Destructor Documentation

QNXAPI::BinaryLockFailure::BinaryLockFailure (int err) [inline]

Construct a **BinaryLockFailure** class.

Parameters:

err standard error code.

Definition at line 40 of file lock.svn-base.QNXAPI::BinaryLockFailure::BinaryLockFailure (int err) [inline]

Construct a **BinaryLockFailure** class.

Parameters:

err standard error code.

Definition at line 40 of file lock.

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

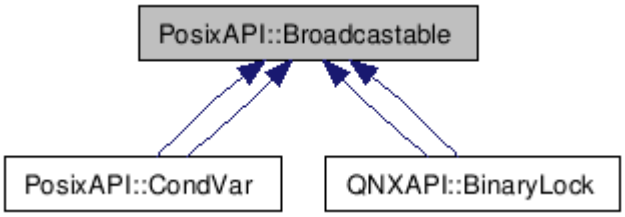
- include/qfc/.svn/text-base/lock.svn-base
- include/qfc/lock

PosixAPI::Broadcastable Class Reference

Interface: **Broadcastable**.

Inherited by **PosixAPI::CondVar**, **PosixAPI::CondVar**, **QNXAPI::BinaryLock**, and **QNXAPI::BinaryLock**.

Inheritance diagram for PosixAPI::Broadcastable:



Detailed Description

Interface: **Broadcastable**.

Definition at line 74 of file interfaces.svn-base.

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

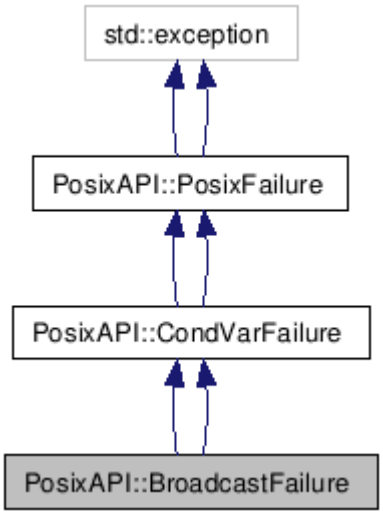
- include/qfc/.svn/text-base/interfaces.svn-base
- include/qfc/interfaces

PosixAPI::BroadcastFailure Class Reference

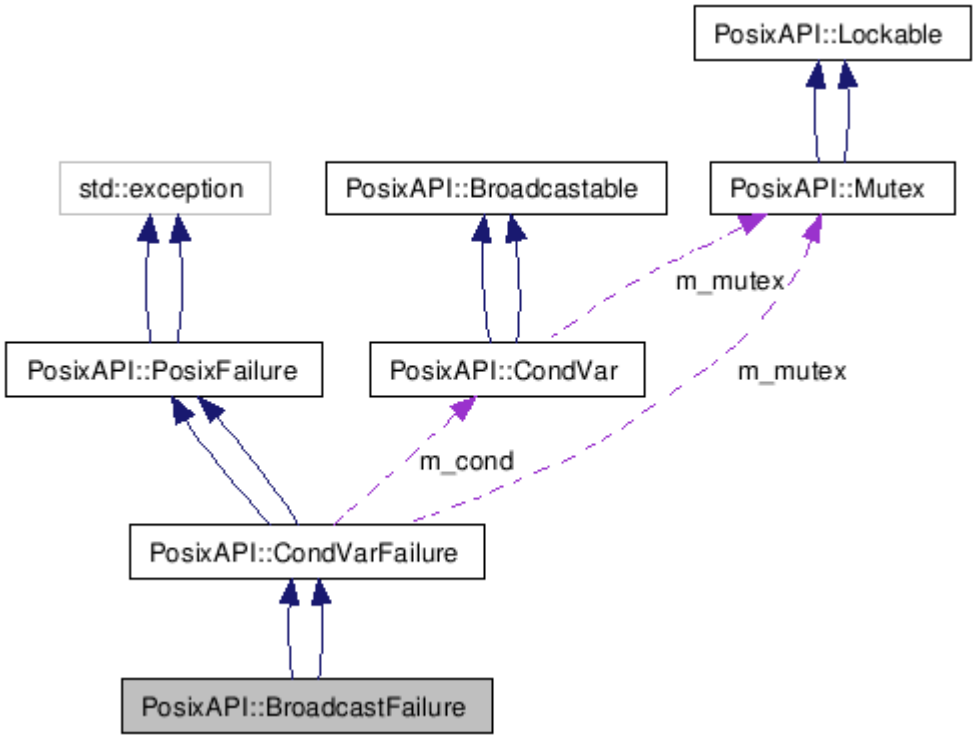
Broadcast failure class.

Inherits **PosixAPI::CondVarFailure**, and **PosixAPI::CondVarFailure**.

Inheritance diagram for PosixAPI::BroadcastFailure:



Collaboration diagram for PosixAPI::BroadcastFailure:



Public Member Functions

- **BroadcastFailure** (const **CondVar** &cond, const **Mutex** &mutex, int err)
  *Construct an instance of a **BroadcastFailure** class.*

- **BroadcastFailure** (const **CondVar** &cond, const **Mutex** &mutex, int err)
  *Construct an instance of a **BroadcastFailure** class.*

Detailed Description

Broadcast failure class.

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 218 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::BroadcastFailure::BroadcastFailure (const CondVar &* cond*, const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **BroadcastFailure** class.

Parameters:

cond **CondVar** that experienced exception
mutex **Mutex** associated with **CondVar**
err Posix error code of failure

Definition at line 228 of file synchron.svn-base.PosixAPI::BroadcastFailure::BroadcastFailure (const CondVar &* cond*, const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **BroadcastFailure** class.

Parameters:

cond **CondVar** that experienced exception
mutex **Mutex** associated with **CondVar**
err Posix error code of failure

Definition at line 228 of file synchron.

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

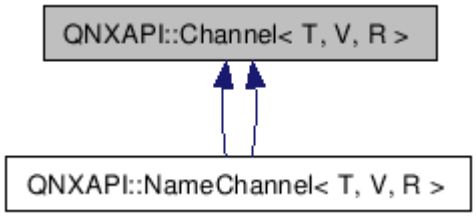
- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

QNXAPI::Channel< T, V, R > Class Template Reference

A **Channel** is a connection point through which IPC messages flow.

Inherited by **QNXAPI::NameChannel< T, V, R >**, and **QNXAPI::NameChannel< T, V, R >**.

Inheritance diagram for QNXAPI::Channel< T, V, R >:



Public Member Functions

- **Channel** (**MessageVector**< T, V, R > &msg, sigc::slot< int, **Reception** & > cbMsg, sigc::slot< void, **Reception** & > cbPulse)
- virtual **~Channel** (void)
- void **Attach** (uint32_t flags=0, int wakeupPri=9)
- virtual int **Receive** (**Reception** &rcp)
- virtual void **Dispatch** (**Reception** &rcp)
- virtual void **Receive** (void)
- virtual void **Pulse** (**Reception** &rcp)
- virtual void **Message** (**Reception** &rcp)
- virtual void **Reply** (**Reception** &rcp, int status, **ReplyVector**< R, V > &vect)
- virtual void **Reply** (**Reception** &rcp, int status)
- virtual void **OnUnblock** (**Reception** &rcp)
- virtual void **OnDisconnect** (**Reception** &rcp)
- virtual void **OnThreadDeath** (**Reception** &rcp)
- virtual void **OnCoidDeath** (**Reception** &rcp)
- void **Wakeup** (void) const
- void **WakeupPri** (int pri) throw ()
- int **AllocatePulseCode** (void) throw ()
- void **Chid** (int chid) throw ()
- int **Chid** (void) const throw ()
- void **Coid** (int coid) throw ()
- int **Coid** (void) const throw ()
- uint8_t **MinCode** (void) const throw ()

  Obtain the minimum pulse code value for use with channel.

- **Channel** (**MessageVector**< T, V, R > &msg, sigc::slot< int, **Reception** & > cbMsg, sigc::slot< void, **Reception** & > cbPulse)
- virtual **~Channel** (void)
- void **Attach** (uint32_t flags=0, int wakeupPri=9)
- virtual int **Receive** (**Reception** &rcp)
- virtual void **Dispatch** (**Reception** &rcp)
- virtual void **Receive** (void)
- virtual void **Pulse** (**Reception** &rcp)
- virtual void **Message** (**Reception** &rcp)
- virtual void **Reply** (**Reception** &rcp, int status, **ReplyVector**< R, V > &vect)
- virtual void **Reply** (**Reception** &rcp, int status)
- virtual void **OnUnblock** (**Reception** &rcp)
- virtual void **OnDisconnect** (**Reception** &rcp)
- virtual void **OnThreadDeath** (**Reception** &rcp)
- virtual void **OnCoidDeath** (**Reception** &rcp)
- void **Wakeup** (void) const
- void **WakeupPri** (int pri) throw ()
- int **AllocatePulseCode** (void) throw ()
- void **Chid** (int chid) throw ()
- int **Chid** (void) const throw ()
- void **Coid** (int coid) throw ()
- int **Coid** (void) const throw ()
- uint8_t **MinCode** (void) const throw ()

  Obtain the minimum pulse code value for use with channel.

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::Channel< T, V, R >

A **Channel** is a connection point through which IPC messages flow.

Definition at line 786 of file ipc.svn-base.

Constructor & Destructor Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::Channel< T, V, R >::Channel (MessageVector< T, V, R > & msg, sigc::slot< int, Reception & > cbMsg, sigc::slot< void, Reception & > cbPulse) [inline]

Construct an instance of a **Channel** class.

Parameters:

msg message vector into which messages will be received.
cbMsg message handler callback

See also:

QNXAPI::NameChannel::m_cbMsg.

Parameters:

cbPulse pulse handler callback

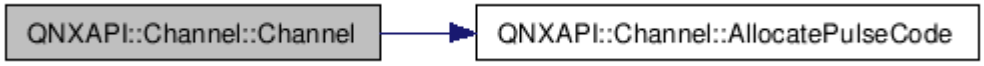
See also:

QNXAPI::NameChannel::m_cbPulse.

Definition at line 796 of file ipc.svn-base.

References QNXAPI::Channel< T, V, R >::AllocatePulseCode().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual QNXAPI::Channel< T, V, R >::~Channel (void) [inline, virtual]

Destroy an instance of a **Channel** class.

Definition at line 809 of file ipc.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::Channel< T, V, R >::Channel (MessageVector< T, V, R > & msg, sigc::slot< int, Reception & > cbMsg, sigc::slot< void, Reception & > cbPulse) [inline]

Construct an instance of a **Channel** class.

Parameters:

msg message vector into which messages will be received.
cbMsg message handler callback

See also:

QNXAPI::NameChannel::m_cbMsg.

Parameters:

cbPulse pulse handler callback

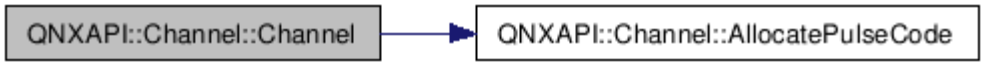
See also:

QNXAPI::NameChannel::m_cbPulse.

Definition at line 796 of file ipc.

References QNXAPI::Channel< T, V, R >::AllocatePulseCode().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual QNXAPI::Channel< T, V, R >::~Channel (void) [inline, virtual]

Destroy an instance of a **Channel** class.

Definition at line 809 of file ipc.

Member Function Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::Attach (uint32_t* flags = *0*, *int* wakeupPri = *9*) [inline]

Attach a **Channel** (creates a channel, and connects to itself).

Parameters:

flags channel flags

See also:

ChannelCreate.

Parameters:

wakeupPri priority at which Wakeup method will drive the channel.

Exceptions:

ChannelCreateFailure
ChannelConnectAttachFailure

Definition at line 825 of file ipc.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual int QNXAPI::Channel< T, V, R >::Receive (Reception & rcp) [inline, virtual]

Decoupled Receive on a **Channel**.

Parameters:

rcp **Reception** to hold the reception data.

Exceptions:

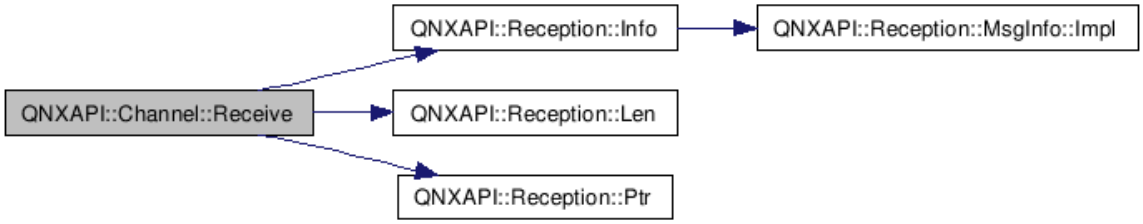
ChannelReceiveFailure

Definition at line 844 of file ipc.svn-base.

References QNXAPI::Reception::Info(), QNXAPI::Reception::Len(), and QNXAPI::Reception::Ptr().

Referenced by QNXAPI::ThreadPool< T, V, R >::Context::Receive().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Dispatch (Reception & rcp) [inline, virtual]

Decoupled Dispatch on a **Reception**.

Parameters:

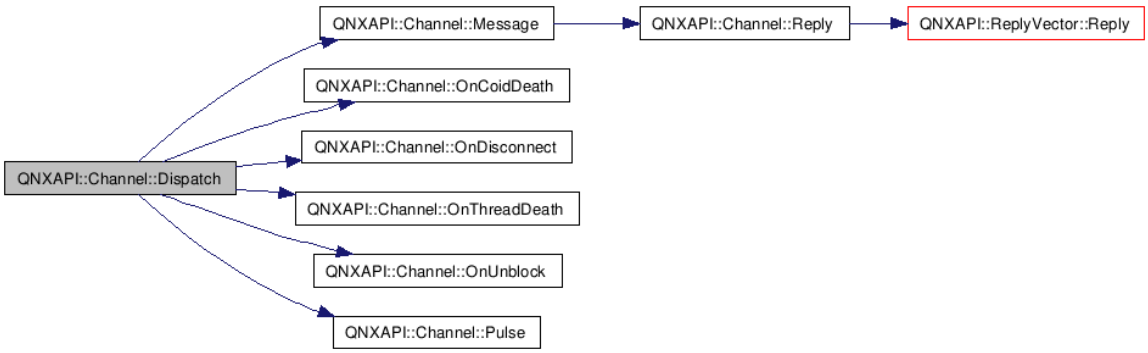
rcp **Reception** for the dispatch.

Definition at line 860 of file ipc.svn-base.

References QNXAPI::Reception::Code(), QNXAPI::Reception::Id(), QNXAPI::Channel< T, V, R >::Message(), QNXAPI::Channel< T, V, R >::OnCoidDeath(), QNXAPI::Channel< T, V, R >::OnDisconnect(), QNXAPI::Channel< T, V, R >::OnThreadDeath(), QNXAPI::Channel< T, V, R >::OnUnblock(), and QNXAPI::Channel< T, V, R >::Pulse().

Referenced by QNXAPI::ThreadPool< T, V, R >::Context::Dispatch(), and QNXAPI::Channel< T, V, R >::Receive().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Receive (void) [inline, virtual]

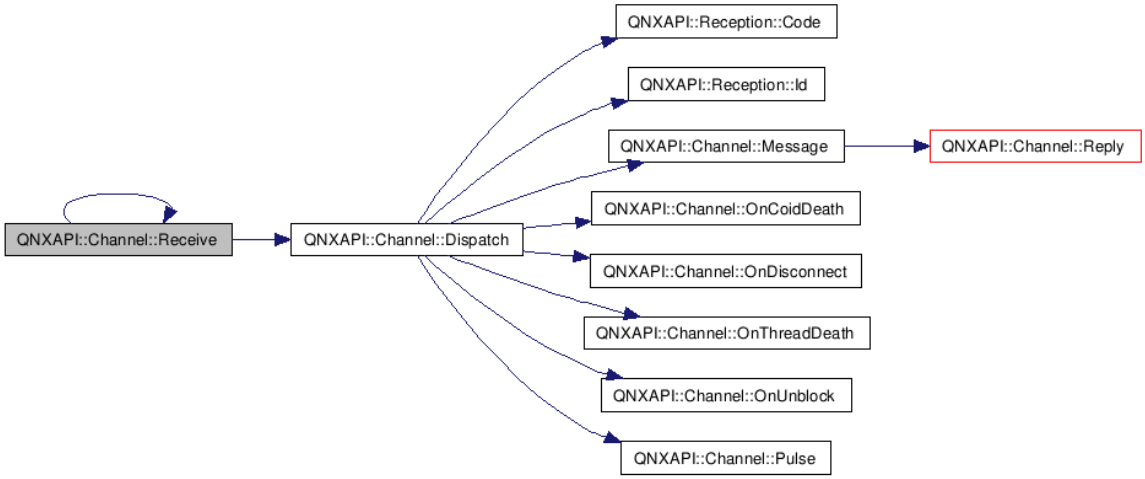
Receive and Dispatch on a channel.

Definition at line 891 of file ipc.svn-base.

References QNXAPI::Channel< T, V, R >::Dispatch(), and QNXAPI::Reception::Id().

Referenced by QNXAPI::Channel< T, V, R >::Receive().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Pulse (Reception & rcp) [inline, virtual]

Handle a Pulse on a reception.

Parameters:

rcp **Reception** as a result of a pulse.

Definition at line 905 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Dispatch().template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Message (Reception & rcp) [inline, virtual]

Handle a Message on a reception.

Parameters:

rcp **Reception** as a result of a message.

Definition at line 915 of file ipc.svn-base.

References QNXAPI::Channel< T, V, R >::Reply().

Referenced by QNXAPI::Channel< T, V, R >::Dispatch().

Here is the call graph for this function:

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Reply (Reception & rcp, int status, ReplyVector< R, V > & vect) [inline, virtual]

Reply to a Message.

Parameters:

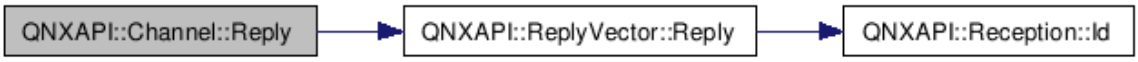
rcp **Reception** as a result of a message.
status status to report to sender.
vect specialized message vector holding reply data.

Definition at line 933 of file ipc.svn-base.

References QNXAPI::ReplyVector< T, V >::Reply().

Referenced by QNXAPI::Channel< T, V, R >::Message().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Reply (Reception & rcp, int status) [inline, virtual]

Reply to a Message.

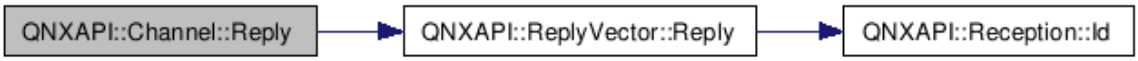
Parameters:

rcp **Reception** as a result of a message.
status status to report to sender.

Definition at line 944 of file ipc.svn-base.

References QNXAPI::ReplyVector< T, V >::Reply().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::OnUnblock (Reception & rcp) [inline, virtual]

Unblock handler.

Parameters:

rcp **Reception**.

Definition at line 956 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Dispatch().template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::OnDisconnect (Reception & rcp) [inline, virtual]

OnDisconnect handler.

Parameters:

rcp **Reception**.

Definition at line 965 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Dispatch().template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::OnThreadDeath (Reception & rcp) [inline, virtual]

OnThreadDeath handler.

Parameters:

rcp **Reception**.

Definition at line 974 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Dispatch().template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::OnCoidDeath (Reception & rcp) [inline, virtual]

OnCoidDeath handler.

Parameters:

rcp **Reception**.

Definition at line 983 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Dispatch().template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::Wakeup (void) const [inline]

Wakeup a **Channel** (deliver a wakeup pulse).

Definition at line 990 of file ipc.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::WakeupPri (int pri) throw () [inline]

Set the Wakeup priority.

Definition at line 1001 of file ipc.svn-base.

Referenced by QNXAPI::NameChannel< T, V, R >::Attach().template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::Channel< T, V, R >::AllocatePulseCode (void) throw () [inline]

Allocate a unique pulse code for the channel.

This can be added to Chid for a process unique pulse code.

Definition at line 1011 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Channel().template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::Chid (int chid) throw () [inline]

Set the channel id for the **Channel**.

Parameters:

chid channel id.

Definition at line 1018 of file ipc.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::Channel< T, V, R >::Chid (void) const throw () [inline]

Get the channel id for the **Channel**.

Returns:

channel id.

Definition at line 1025 of file ipc.svn-base.

Referenced by QNXAPI::NameChannel< T, V, R >::Attach().template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::Coid (int coid) throw () [inline]

Set the connection id for the **Channel**.

Parameters:

coid connection id.

Definition at line 1032 of file ipc.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::Channel< T, V, R >::Coid (void) const throw () [inline]

Get the connection id for the **Channel**.

Returns:

connection id.

Definition at line 1039 of file ipc.svn-base.

Referenced by QNXAPI::NameChannel< T, V, R >::Attach().template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint8_t QNXAPI::Channel< T, V, R >::MinCode (void) const throw () [inline]

Obtain the minimum pulse code value for use with channel.

Returns:

Minimum acceptable pulse code value

Definition at line 1046 of file ipc.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::Attach (uint32_t flags = 0, int wakeupPri = 9) [inline]

Attach a **Channel** (creates a channel, and connects to itself).

Parameters:

flags channel flags

See also:

ChannelCreate.

Parameters:

wakeupPri priority at which Wakeup method will drive the channel.

Exceptions:

ChannelCreateFailure
ChannelConnectAttachFailure

Definition at line 825 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual int QNXAPI::Channel< T, V, R >::Receive (Reception & rcp) [inline, virtual]

Decoupled Receive on a **Channel**.

Parameters:

rcp **Reception** to hold the reception data.

Exceptions:

ChannelReceiveFailure

Definition at line 844 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Dispatch (Reception & rcp) [inline, virtual]

Decoupled Dispatch on a **Reception**.

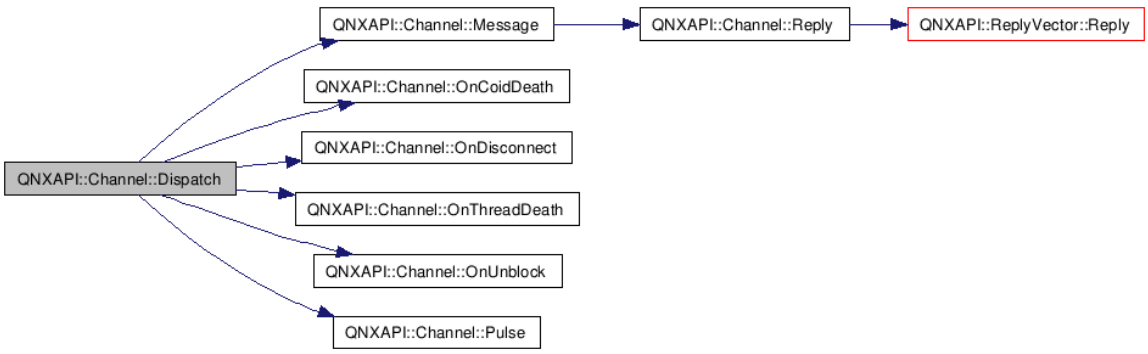
Parameters:

rcp **Reception** for the dispatch.

Definition at line 860 of file ipc.

References QNXAPI::Channel< T, V, R >::Message(), QNXAPI::Channel< T, V, R >::OnCoidDeath(), QNXAPI::Channel< T, V, R >::OnDisconnect(), QNXAPI::Channel< T, V, R >::OnThreadDeath(), QNXAPI::Channel< T, V, R >::OnUnblock(), and QNXAPI::Channel< T, V, R >::Pulse().

Here is the call graph for this function:



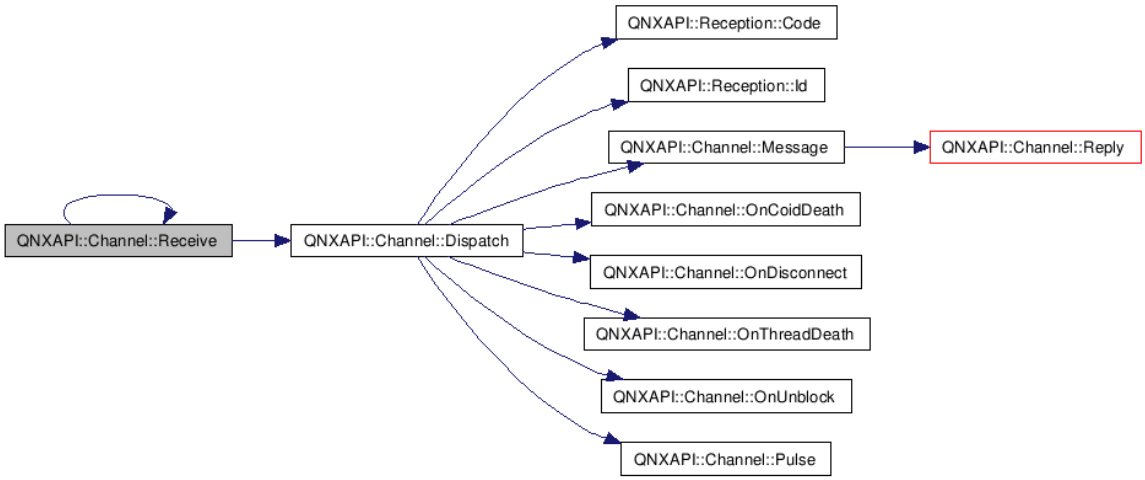
template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Receive (void) [inline, virtual]

Receive and Dispatch on a channel.

Definition at line 891 of file ipc.

References QNXAPI::Channel< T, V, R >::Dispatch(), and QNXAPI::Channel< T, V, R >::Receive().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Pulse (Reception & rcp) [inline, virtual]

Handle a Pulse on a reception.

Parameters:

rcp **Reception** as a result of a pulse.

Definition at line 905 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Message (Reception & rcp) [inline, virtual]

Handle a Message on a reception.

Parameters:

rcp **Reception** as a result of a message.

Definition at line 915 of file ipc.

References QNXAPI::Channel< T, V, R >::Reply().

Here is the call graph for this function:

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Reply (Reception & rcp, int status, ReplyVector< R, V > & vect) [inline, virtual]

Reply to a Message.

Parameters:

rcp **Reception** as a result of a message.
status status to report to sender.
vect specialized message vector holding reply data.

Definition at line 933 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::Reply (Reception & rcp, int status) [inline, virtual]

Reply to a Message.

Parameters:

rcp **Reception** as a result of a message.
status status to report to sender.

Definition at line 944 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::OnUnblock (Reception & rcp) [inline, virtual]

Unblock handler.

Parameters:

rcp **Reception**.

Definition at line 956 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::OnDisconnect (Reception & rcp) [inline, virtual]

OnDisconnect handler.

Parameters:

rcp **Reception**.

Definition at line 965 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::OnThreadDeath (Reception & rcp) [inline, virtual]

OnThreadDeath handler.

Parameters:

rcp **Reception**.

Definition at line 974 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::Channel< T, V, R >::OnCoidDeath (Reception & rcp) [inline, virtual]

OnCoidDeath handler.

Parameters:

rcp **Reception**.

Definition at line 983 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::Wakeup (void) const [inline]

Wakeup a **Channel** (deliver a wakeup pulse).

Definition at line 990 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::WakeupPri (int pri) throw () [inline]

Set the Wakeup priority.

Definition at line 1001 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::Channel< T, V, R >::AllocatePulseCode (void) throw () [inline]

Allocate a unique pulse code for the channel.

This can be added to Chid for a process unique pulse code.

Definition at line 1011 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::Chid (int chid) throw () [inline]

Set the channel id for the **Channel**.

Parameters:

chid channel id.

Definition at line 1018 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::Channel< T, V, R >::Chid (void) const throw () [inline]

Get the channel id for the **Channel**.

Returns:

channel id.

Definition at line 1025 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::Channel< T, V, R >::Coid (int coid) throw () [inline]

Set the connection id for the **Channel**.

Parameters:

coid connection id.

Definition at line 1032 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::Channel< T, V, R >::Coid (void) const throw () [inline]

Get the connection id for the **Channel**.

Returns:

connection id.

Definition at line 1039 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint8_t QNXAPI::Channel< T, V, R >::MinCode (void) const throw () [inline]

Obtain the minimum pulse code value for use with channel.

Returns:

Minimum acceptable pulse code value

Definition at line 1046 of file ipc.

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

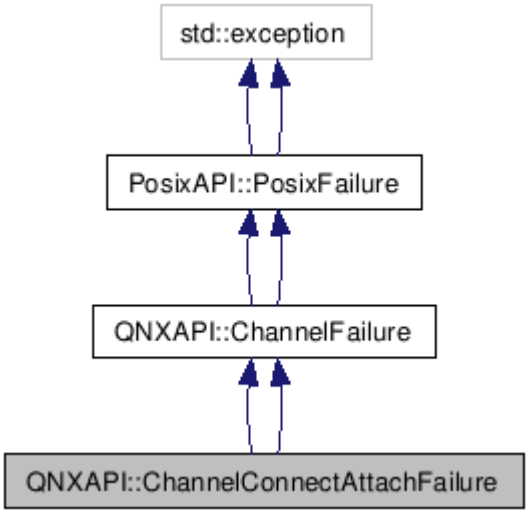
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::ChannelConnectAttachFailure Class Reference

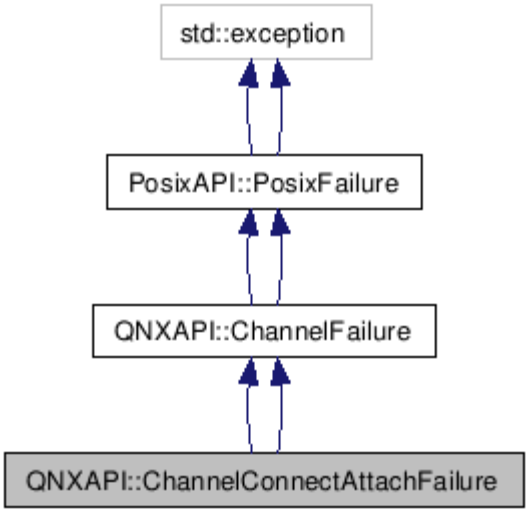
Channel could not be attached to.

Inherits **QNXAPI::ChannelFailure**, and **QNXAPI::ChannelFailure**.

Inheritance diagram for QNXAPI::ChannelConnectAttachFailure:



Collaboration diagram for QNXAPI::ChannelConnectAttachFailure:



Public Member Functions

- **ChannelConnectAttachFailure** (int err)
  *Construct an instance of a **ChannelConnectAttachFailure** exception.*

- **ChannelConnectAttachFailure** (int err)
  *Construct an instance of a **ChannelConnectAttachFailure** exception.*

Detailed Description

Channel could not be attached to.

Thrown when an error occurs as the result of an attempt to connect to a channel.

Definition at line 217 of file ipc.svn-base.

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

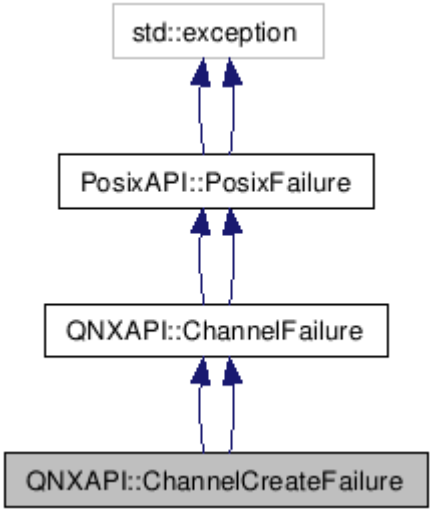
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::ChannelCreateFailure Class Reference

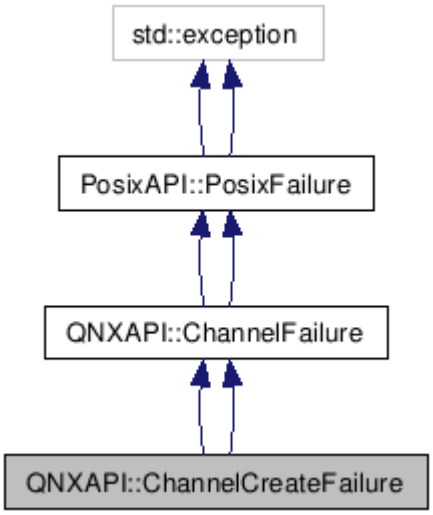
Channel could not be created.

Inherits **QNXAPI::ChannelFailure**, and **QNXAPI::ChannelFailure**.

Inheritance diagram for QNXAPI::ChannelCreateFailure:



Collaboration diagram for QNXAPI::ChannelCreateFailure:



Public Member Functions

- **ChannelCreateFailure** (int err)
  *Construct an instance of a **ChannelCreateFailure** exception.*

- **ChannelCreateFailure** (int err)
  *Construct an instance of a **ChannelCreateFailure** exception.*

Detailed Description

Channel could not be created.

Thrown when an error occurs as the result of a channel creation attempt.

Definition at line 201 of file ipc.svn-base.

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

- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

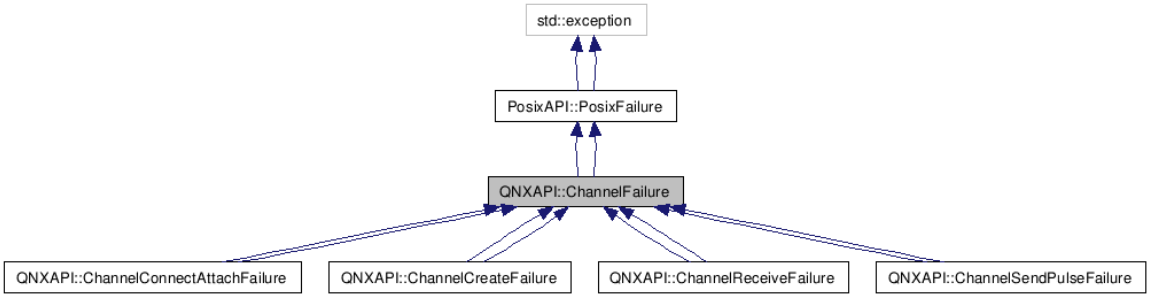
QNXAPI::ChannelFailure Class Reference

Base class for channel failures.

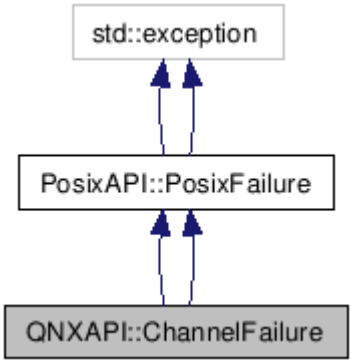
Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inherited by **QNXAPI::ChannelConnectAttachFailure**, **QNXAPI::ChannelConnectAttachFailure**, **QNXAPI::ChannelCreateFailure**, **QNXAPI::ChannelCreateFailure**, **QNXAPI::ChannelReceiveFailure**, **QNXAPI::ChannelReceiveFailure**, **QNXAPI::ChannelSendPulseFailure**, and **QNXAPI::ChannelSendPulseFailure**.

Inheritance diagram for QNXAPI::ChannelFailure:



Collaboration diagram for QNXAPI::ChannelFailure:



Public Member Functions

- **ChannelFailure** (int err)
  *Construct an instance of a **ChannelFailure** exception.*

- **ChannelFailure** (int err)
  *Construct an instance of a **ChannelFailure** exception.*

Detailed Description

Base class for channel failures.

Definition at line 185 of file ipc.svn-base.

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

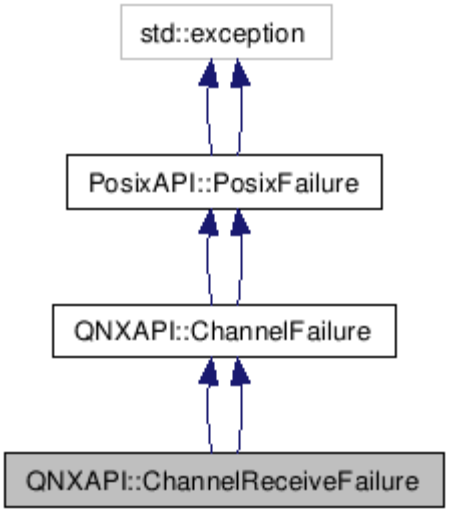
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::ChannelReceiveFailure Class Reference

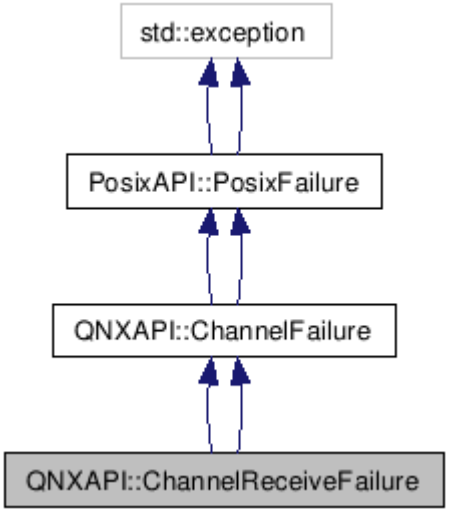
Channel could not receive a message.

Inherits **QNXAPI::ChannelFailure**, and **QNXAPI::ChannelFailure**.

Inheritance diagram for QNXAPI::ChannelReceiveFailure:



Collaboration diagram for QNXAPI::ChannelReceiveFailure:



Public Member Functions

- **ChannelReceiveFailure** (int err)
  *Construct an instance of a **ChannelReceiveFailure** exception.*

- **ChannelReceiveFailure** (int err)
  *Construct an instance of a **ChannelReceiveFailure** exception.*

Detailed Description

Channel could not receive a message.

Thrown when an error occurs while a channel is attempting a receive.

Definition at line 233 of file ipc.svn-base.

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

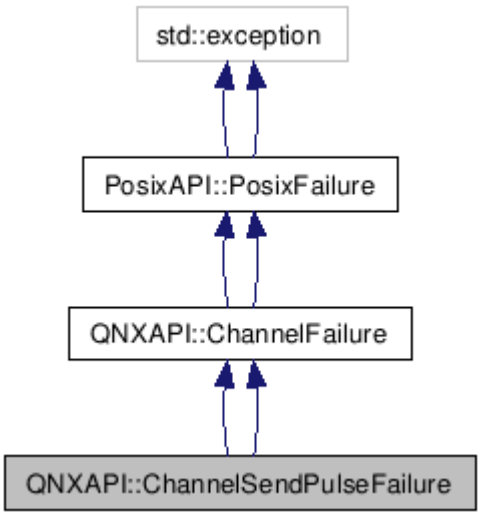
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::ChannelSendPulseFailure Class Reference

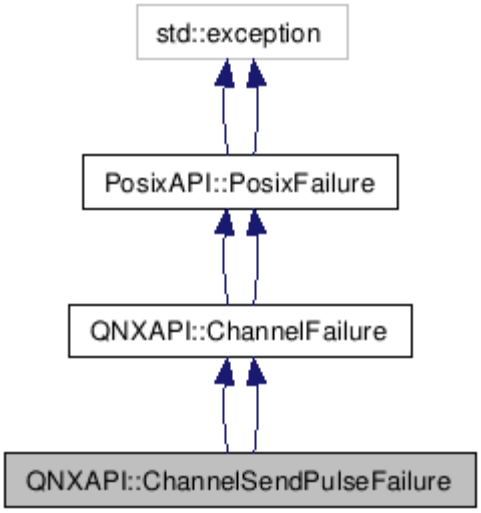
A pulse could not be sent on a channel.

Inherits **QNXAPI::ChannelFailure**, and **QNXAPI::ChannelFailure**.

Inheritance diagram for QNXAPI::ChannelSendPulseFailure:



Collaboration diagram for QNXAPI::ChannelSendPulseFailure:



Public Member Functions

- **ChannelSendPulseFailure** (int err)
  *Construct an instance of a **ChannelSendPulseFailure** exception.*

- **ChannelSendPulseFailure** (int err)
  *Construct an instance of a **ChannelSendPulseFailure** exception.*

Detailed Description

A pulse could not be sent on a channel.

Thrown when an error occurs as the result of a channel sending a pulse.

Definition at line 249 of file ipc.svn-base.

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

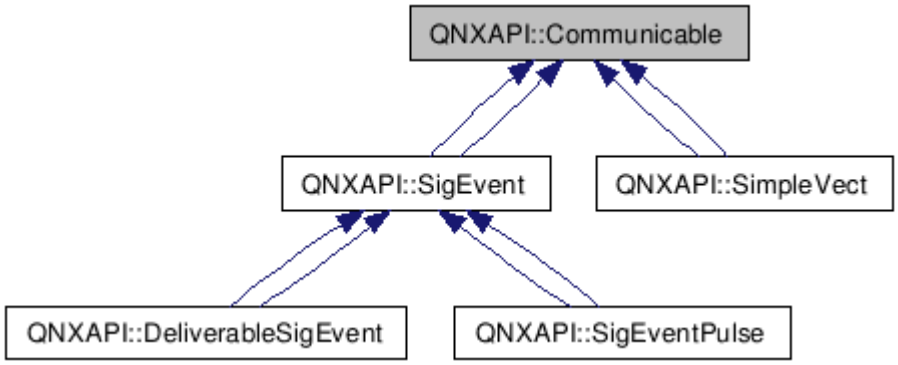
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::Communicable Class Reference

Communicable class.

Inherited by **QNXAPI::SigEvent**, **QNXAPI::SigEvent**, **QNXAPI::SimpleVect**, and **QNXAPI::SimpleVect**.

Inheritance diagram for QNXAPI::Communicable:



Detailed Description

Communicable class.

Interface for classes that are "communicable" (i.e. can be sent using IPC).

Communicable classes must be **Decomposable**.

Definition at line 108 of file interfaces.svn-base.

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

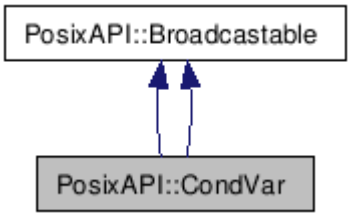
- include/qfc/.svn/text-base/interfaces.svn-base
- include/qfc/interfaces

PosixAPI::CondVar Class Reference

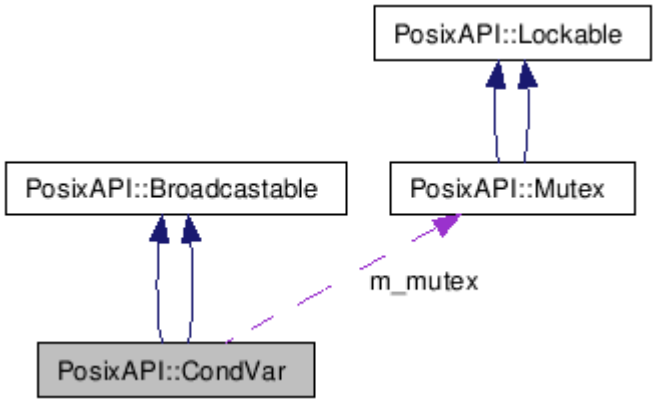
CondVar class.

Inherits **PosixAPI::Broadcastable**, and **PosixAPI::Broadcastable**.

Inheritance diagram for PosixAPI::CondVar:



Collaboration diagram for PosixAPI::CondVar:



Public Member Functions

- **CondVar** (**Mutex** &mutex)
  *Construct an instance of a **CondVar** class.*

- virtual **~CondVar** ()
  *Destroy an instance of a **CondVar** class.*

- void **Wait** ()
  Wait for change in predicate.

- void **TimedWait** (uint64_t to)
  Wait for change in predicate with timeout.

- void **Broadcast** (void) const
  Notify waiters of change in predicate.

- **CondVar** (**Mutex** &mutex)
  *Construct an instance of a **CondVar** class.*

- virtual **~CondVar** ()
  *Destroy an instance of a **CondVar** class.*

- void **Wait** ()
  Wait for change in predicate.

- void **TimedWait** (uint64_t to)
  Wait for change in predicate with timeout.

- void **Broadcast** (void) const
  Notify waiters of change in predicate.

Detailed Description

CondVar class.

Implements conditional notification of change in predicate

Author:
Rennie Allen `renniеallen@gmail.com`

Definition at line 377 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::CondVar::CondVar (Mutex &* `mutex`*)

Construct an instance of a **CondVar** class.

Parameters:
mutex **Mutex** associated with **CondVar**

PosixAPI::CondVar::CondVar (Mutex &* `mutex`*)

Construct an instance of a **CondVar** class.

Parameters:
mutex **Mutex** associated with **CondVar**

Member Function Documentation

void PosixAPI::CondVar::TimedWait (uint64_t* to*)

Wait for change in predicate with timeout.

Parameters:

to **Timeout**

void PosixAPI::CondVar::TimedWait (uint64_t* to*)

Wait for change in predicate with timeout.

Parameters:

to **Timeout**

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

- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

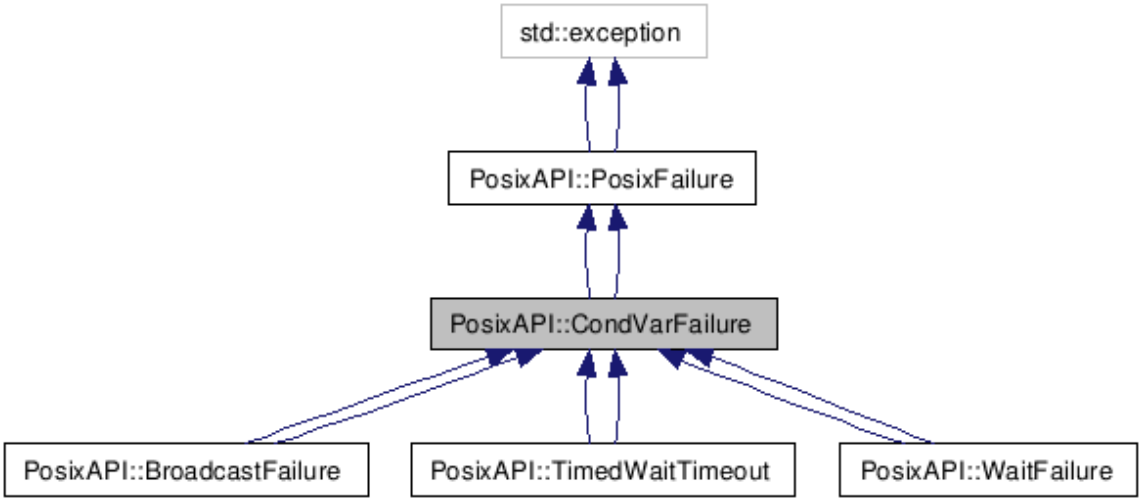
PosixAPI::CondVarFailure Class Reference

CondVarFailure base class.

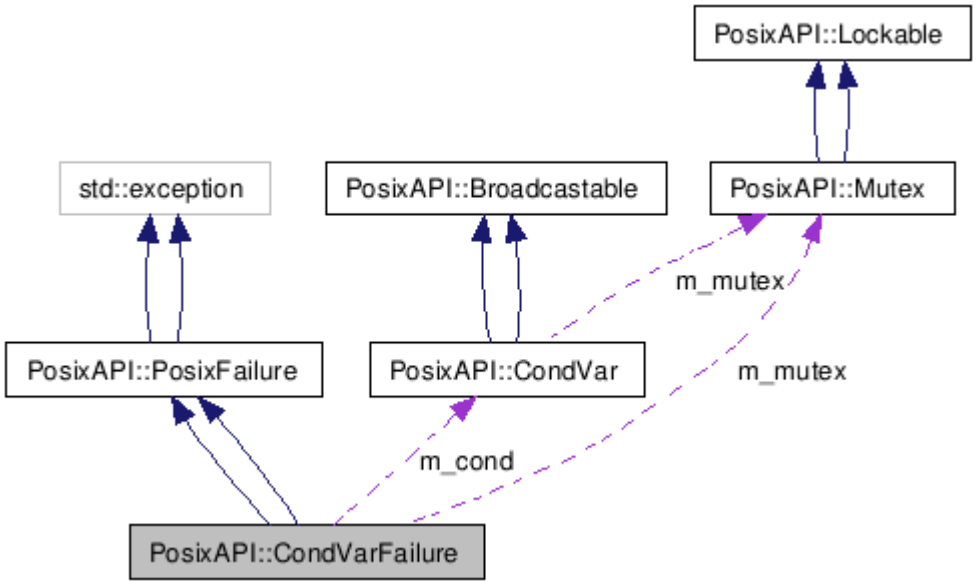
Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inherited by **PosixAPI::BroadcastFailure**, **PosixAPI::BroadcastFailure**, **PosixAPI::TimedWaitTimeout**, **PosixAPI::TimedWaitTimeout**, **PosixAPI::WaitFailure**, and **PosixAPI::WaitFailure**.

Inheritance diagram for PosixAPI::CondVarFailure:



Collaboration diagram for PosixAPI::CondVarFailure:



Public Member Functions

- **CondVarFailure** (const **CondVar** &cond, const **Mutex** &mutex, int err)
  *Construct an instance of a **CondVarFailure** class.*

- **CondVarFailure** (const **CondVar** &cond, const **Mutex** &mutex, int err)
  *Construct an instance of a **CondVarFailure** class.*

Detailed Description

CondVarFailure base class.

Author:

Rennie Allen rennieallen@gmail.com

Definition at line 154 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::CondVarFailure::CondVarFailure (const CondVar &* cond*, const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **CondVarFailure** class.

Parameters:

cond **CondVar** that experienced exception
mutex **Mutex** associated with **CondVar**
err Posix error code of failure

Definition at line 164 of file synchron.svn-base.PosixAPI::CondVarFailure::CondVarFailure (const CondVar &* cond*, const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **CondVarFailure** class.

Parameters:

cond **CondVar** that experienced exception
mutex **Mutex** associated with **CondVar**
err Posix error code of failure

Definition at line 164 of file synchron.

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

- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

QNXAPI::Connection Class Reference

Connection class. Represents a connection to a channel.

Public Member Functions

- **Connection** (void)
  *Construct an instance of a **Connection** class.*

- **Connection** (int id)
  *Construct an instance of a **Connection** class.*

- void **Id** (int id)
  Set the low-level connection id.

- int **Id** (void)
  Get the low-level connection id.

- void **Attach** (int chid)
  Attach to a channel.

- **Connection** (void)
  *Construct an instance of a **Connection** class.*

- **Connection** (int id)
  *Construct an instance of a **Connection** class.*

- void **Id** (int id)
  Set the low-level connection id.

- int **Id** (void)
  Get the low-level connection id.

- void **Attach** (int chid)
  Attach to a channel.

Detailed Description

Connection class. Represents a connection to a channel.

The connection class allows a client to send messages on a channel.

Definition at line 466 of file ipc.svn-base.

Constructor & Destructor Documentation

QNXAPI::Connection::Connection (int id) [inline]

Construct an instance of a **Connection** class.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

id instantiate connection class around an existing connection.

Definition at line 481 of file ipc.svn-base.QNXAPI::Connection::Connection (int id) [inline]

Construct an instance of a **Connection** class.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

id instantiate connection class around an existing connection.

Definition at line 481 of file ipc.

Member Function Documentation

void QNXAPI::Connection::Id (int id) [inline]

Set the low-level connection id.

Parameters:

id set this instance of connection to id.

Definition at line 490 of file ipc.svn-base.

Referenced by QNXAPI::NameConnection< T, V, R >::Connect(), QNXAPI::MessageVector< T, V, R >::Send(), QNXAPI::NameConnection< T, V, R >::SendPulse(), and QNXAPI::NameConnection< T, V, R >::~NameConnection().int QNXAPI::Connection::Id (void) [inline]

Get the low-level connection id.

Returns:

this connections id.

Definition at line 496 of file ipc.svn-base.void QNXAPI::Connection::Attach (int* **chid) [inline]***

Attach to a channel.

Parameters:

chid channel id to attach to.

Definition at line 503 of file ipc.svn-base.void QNXAPI::Connection::Id (int* **id) [inline]***

Set the low-level connection id.

Parameters:

id set this instance of connection to id.

Definition at line 490 of file ipc.int QNXAPI::Connection::Id (void) [inline]

Get the low-level connection id.

Returns:

this connections id.

Definition at line 496 of file ipc.void QNXAPI::Connection::Attach (int* **chid) [inline]***

Attach to a channel.

Parameters:

chid channel id to attach to.

Definition at line 503 of file ipc.

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

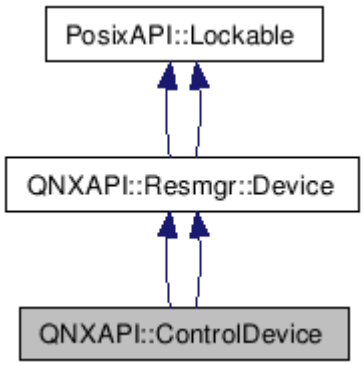
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::ControlDevice Class Reference

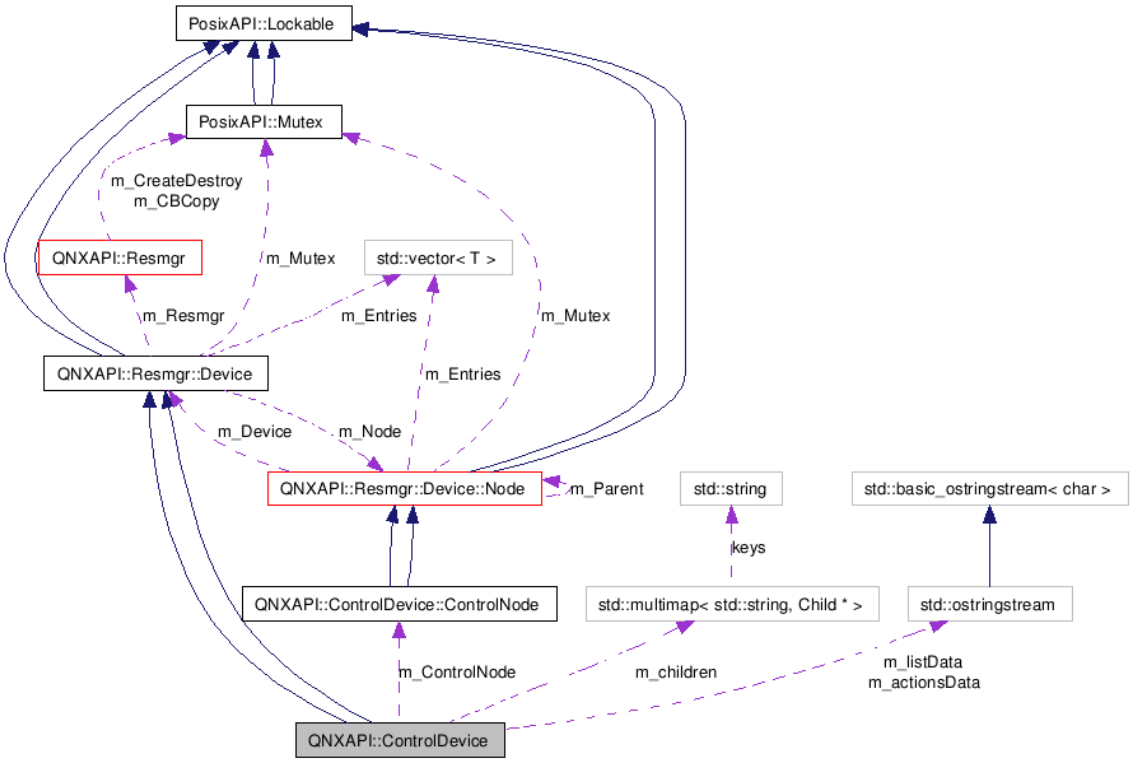
The **Sysmon ControlDevice** class.

Inherits **QNXAPI::Resmgr::Device**, and **QNXAPI::Resmgr::Device**.

Inheritance diagram for QNXAPI::ControlDevice:



Collaboration diagram for QNXAPI::ControlDevice:



Public Types

- enum **ReadMode**
  Read modes for this control device.

- enum **ReadMode**
  Read modes for this control device.

Public Member Functions

- virtual Node & **RootNode** (void)
  Return devices node.

- virtual Node & **RootNode** (void)
  Return devices node.

Classes

- class **ControlNode**
  *The **Sysmon ControlNode** class.*

Detailed Description

The **Sysmon ControlDevice** class.

Specializes the **Resmgr::Device** class, provide a control device.

Author:

rennieallen@gmail.com

Definition at line 47 of file sysmon.svn-base.

Member Function Documentation

virtual Node& QNXAPI::ControlDevice::RootNode (void) [virtual]

Return devices node.

Returns:

the root Node for the device.

Reimplemented from *QNXAPI::Resmgr::Device* **(p.250).*virtual Node& QNXAPI::ControlDevice::RootNode (void) [virtual]***

Return devices node.

Returns:

the root Node for the device.

Reimplemented from **QNXAPI::Resmgr::Device** (*p.250*).

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

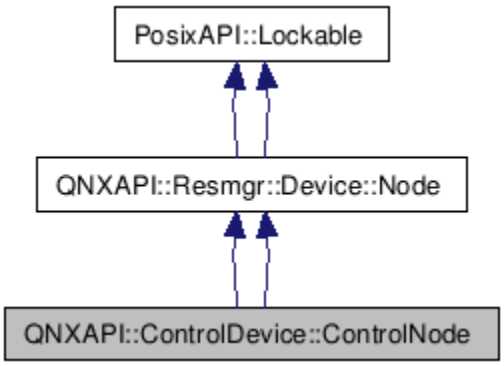
- include/qfc/.svn/text-base/sysmon.svn-base
- include/qfc/sysmon

QNXAPI::ControlDevice::ControlNode Class Reference

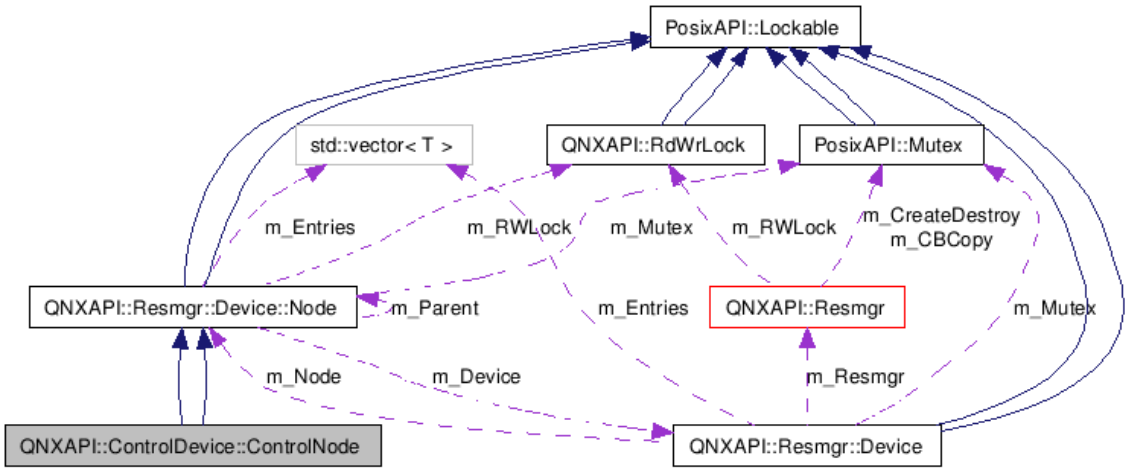
The **Sysmon ControlNode** class.

Inherits **QNXAPI::Resmgr::Device::Node**, and **QNXAPI::Resmgr::Device::Node**.

Inheritance diagram for QNXAPI::ControlDevice::ControlNode:



Collaboration diagram for QNXAPI::ControlDevice::ControlNode:



Public Member Functions

- **ControlNode** (**QNXAPI::Resmgr::Device** ***device**, **QNXAPI::Resmgr::Device::Node** &parent, dev_t type, mode_t mode)
- virtual int **Write** (resmgr_context_t *ctp, io_write_t *msg, **QNXAPI::Resmgr::Ocb** &**ocb**) throw ()
  Perform write() operation on node.

- virtual int **Read** (resmgr_context_t *ctp, io_read_t *msg, **QNXAPI::Resmgr::Ocb** &**ocb**) throw ()
  Perform read() operation on node.

- **ControlNode** (**QNXAPI::Resmgr::Device** ***device**, **QNXAPI::Resmgr::Device::Node** &parent, dev_t type, mode_t mode)
- virtual int **Write** (resmgr_context_t *ctp, io_write_t *msg, **QNXAPI::Resmgr::Ocb** &**ocb**) throw ()
  Perform write() operation on node.

- virtual int **Read** (resmgr_context_t *ctp, io_read_t *msg, **QNXAPI::Resmgr::Ocb** &**ocb**) throw ()
  Perform read() operation on node.

Detailed Description

The **Sysmon ControlNode** class.

Specializes the **Resmgr::Device::Node** class, to snoop node data for commands, and to insert output into node data.

Author:
rennieallen@gmail.com

Definition at line 61 of file sysmon.svn-base.

Constructor & Destructor Documentation

QNXAPI::ControlDevice::ControlNode::ControlNode (QNXAPI::Resmgr::Device * device, QNXAPI::Resmgr::Device::Node & parent, dev_t type, mode_t mode)

Create a contol node.

Parameters:

device pointer to **ControlDevice** parent.
parent parent node.
type device type.
mode device nodes mode.

QNXAPI::ControlDevice::ControlNode::ControlNode (QNXAPI::Resmgr::Device * device, QNXAPI::Resmgr::Device::Node & parent, dev_t type, mode_t mode)

Create a contol node.

Parameters:

device pointer to **ControlDevice** parent.
parent parent node.
type device type.
mode device nodes mode.

Member Function Documentation

virtual int QNXAPI::ControlDevice::ControlNode::Write (resmgr_context_t * ctp, io_write_t * msg, QNXAPI::Resmgr::Ocb & ocb) throw () [virtual]

Perform write() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library write message.
ocb open control block to control write.

Returns:

small positive integer from errno.h

Reimplemented from QNXAPI::Resmgr::Device::Node (p.280).virtual int QNXAPI::ControlDevice::ControlNode::Read (resmgr_context_t * ctp, io_read_t * msg, QNXAPI::Resmgr::Ocb & ocb) throw () [virtual]

Perform read() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb open control block to control read.

Returns:

small positive integer from errno.h

Reimplemented from *QNXAPI::Resmgr::Device::Node* ***(p.279).virtual int QNXAPI::ControlDevice::ControlNode::Write (resmgr_context_t * ctp, io_write_t * msg, QNXAPI::Resmgr::Ocb & ocb) throw () [virtual]***

Perform write() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library write message.
ocb open control block to control write.

Returns:

small positive integer from errno.h

Reimplemented from *QNXAPI::Resmgr::Device::Node* ***(p.280).virtual int QNXAPI::ControlDevice::ControlNode::Read (resmgr_context_t * ctp, io_read_t * msg, QNXAPI::Resmgr::Ocb & ocb) throw () [virtual]***

Perform read() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb open control block to control read.

Returns:

small positive integer from errno.h

Reimplemented from **QNXAPI::Resmgr::Device::Node** (*p.279*).

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

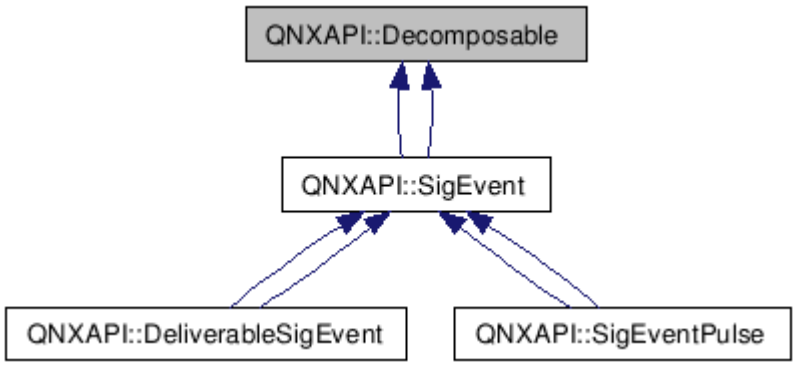
- include/qfc/.svn/text-base/sysmon.svn-base
- include/qfc/sysmon

QNXAPI::Decomposable Class Reference

Decomposable class.

Inherited by **QNXAPI::SigEvent**, and **QNXAPI::SigEvent**.

Inheritance diagram for QNXAPI::Decomposable:



Detailed Description

Decomposable class.

Interface for classes that are "decomposable" (i.e. can be converted into basic element).

Decomposable classes must implement a single linear area of Plain Old Data (Pod).

This interfaces defines the access to the POD.

Definition at line 93 of file interfaces.svn-base.

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

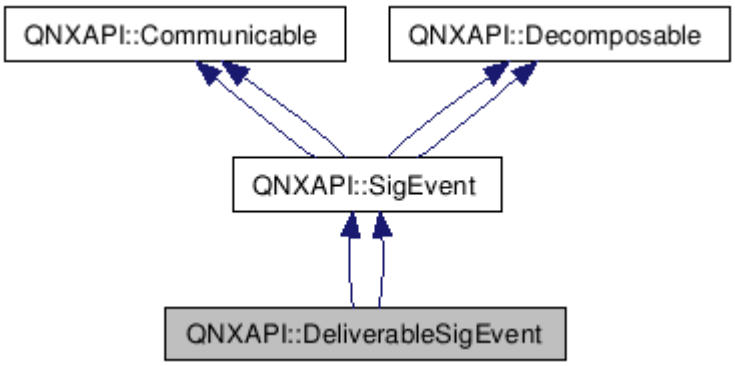
- include/qfc/.svn/text-base/interfaces.svn-base
- include/qfc/interfaces

QNXAPI::DeliverableSigEvent Class Reference

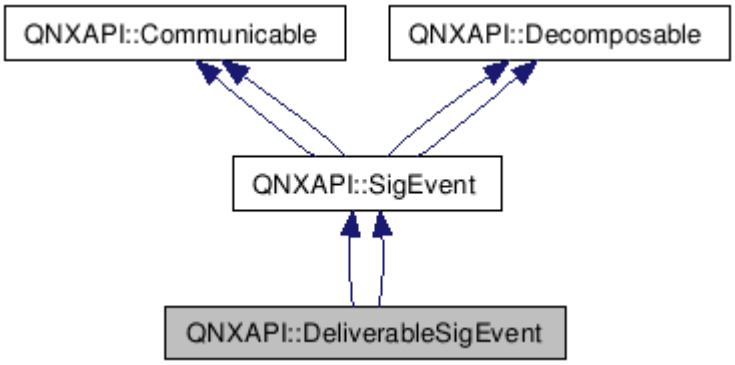
Deliverable **SigEvent**.

Inherits **QNXAPI::SigEvent**, and **QNXAPI::SigEvent**.

Inheritance diagram for QNXAPI::DeliverableSigEvent:



Collaboration diagram for QNXAPI::DeliverableSigEvent:



Public Member Functions

- **DeliverableSigEvent** (void)
  *Construct an instance of a **DeliverableSigEvent** class.*

- **DeliverableSigEvent** (int rcvId)
  *Construct an instance of a **DeliverableSigEvent** class.*

- void **Id** (int rcvId)
  *Set the rcvId associated with this **SigEvent**.*

- int **Id** (void)
  *Obtain the rcvId associated with this **SigEvent**.*

- void **Deliver** (void)
  *Deliver this **SigEvent** to originator.*

- **DeliverableSigEvent** (void)
  *Construct an instance of a **DeliverableSigEvent** class.*

- **DeliverableSigEvent** (int rcvId)
  *Construct an instance of a **DeliverableSigEvent** class.*

- void **Id** (int rcvId)
  *Set the rcvId associated with this **SigEvent**.*

- int **Id** (void)
  *Obtain the rcvId associated with this **SigEvent**.*

- void **Deliver** (void)
  *Deliver this **SigEvent** to originator.*

Detailed Description

Deliverable **SigEvent**.

SigEvent that can be delivered over a channel asynchronously

Author:

Rennie Allen `rennieallen@gmail.com`

Examples:

Serv/Serv.h.

Definition at line 261 of file sigevent.svn-base.

Constructor & Destructor Documentation

QNXAPI::DeliverableSigEvent::DeliverableSigEvent (int* `rcvId`*) [inline]

Construct an instance of a **DeliverableSigEvent** class.

Parameters:

rcvId Receive ID provided at the time this **SigEvent** was received

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 277 of file sigevent.svn-base.QNXAPI::DeliverableSigEvent::DeliverableSigEvent (int* `rcvId`*) [inline]

Construct an instance of a **DeliverableSigEvent** class.

Parameters:

rcvId Receive ID provided at the time this **SigEvent** was received

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 277 of file sigevent.

Member Function Documentation

void QNXAPI::DeliverableSigEvent::Id (int* `rcvId`*) [inline]

Set the rcvId associated with this **SigEvent**.

Parameters:

rcvId Receive ID provided at the time this **SigEvent** was received

Definition at line 286 of file sigevent.svn-base.int QNXAPI::DeliverableSigEvent::Id (void) [inline]

Obtain the rcvId associated with this **SigEvent**.

Returns:

Receive ID provided at the time this **SigEvent** was received

Definition at line 296 of file sigevent.svn-base.void QNXAPI::DeliverableSigEvent::Id (int* `rcvId`*) [inline]

Set the rcvId associated with this **SigEvent**.

Parameters:

rcvId Receive ID provided at the time this **SigEvent** was received

Definition at line 286 of file sigevent.int QNXAPI::DeliverableSigEvent::Id (void) [inline]

Obtain the rcvId associated with this **SigEvent**.

Returns:

Receive ID provided at the time this **SigEvent** was received

Definition at line 296 of file sigevent.

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

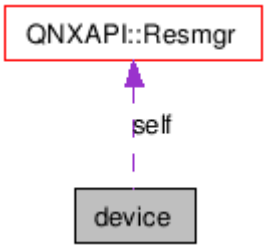
- include/qfc/.svn/text-base/sigevent.svn-base
- include/qfc/sigevent

---

device Struct Reference

'C' representation of a resource manager device

Collaboration diagram for device:



Public Attributes

- iofunc_attr_t **hdr**
  storage for header

- **QNXAPI::Resmgr** * **self**
  storage for pointer to self

- **QNXAPI::Resmgr** * **self**
  storage for pointer to self

Detailed Description

'C' representation of a resource manager device

Extension of iofunc_attr struct to allow hooking of resmgr class.

Definition at line 59 of file resmgr.svn-base.

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

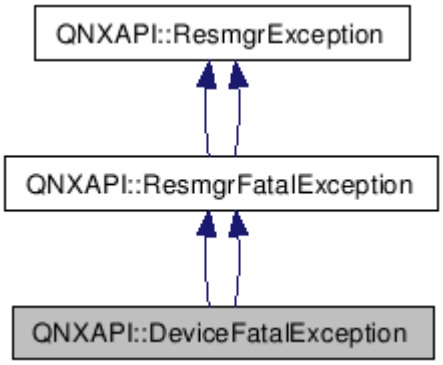
- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

QNXAPI::DeviceFatalException Class Reference

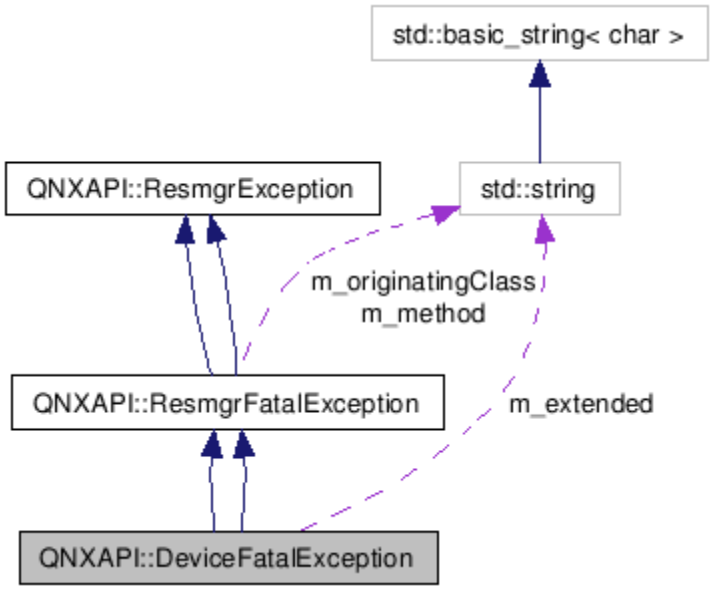
Resource Manager Device fatal exception base class.

Inherits **QNXAPI::ResmgrFatalException**, and **QNXAPI::ResmgrFatalException**.

Inheritance diagram for QNXAPI::DeviceFatalException:



Collaboration diagram for QNXAPI::DeviceFatalException:



Public Member Functions

- **DeviceFatalException** (const char *func, int lineNum, int err, const char *extendedError="None available")
  *Build a new **DeviceFatalException**.*

- const std::string & **ExtendedErrorDescription** (void)
  Retrieve extended information regarding exception.

- **DeviceFatalException** (const char *func, int lineNum, int err, const char *extendedError="None available")
  *Build a new **DeviceFatalException**.*

- const std::string & **ExtendedErrorDescription** (void)
  Retrieve extended information regarding exception.

Detailed Description

Resource Manager Device fatal exception base class.

This class provides information that travels with Device fatal exceptions.

Definition at line 193 of file resmgr.svn-base.

Constructor & Destructor Documentation

QNXAPI::DeviceFatalException::DeviceFatalException (const char * func, int lineNum, int err, const char * extendedError = "None available") [inline]

Build a new **DeviceFatalException**.

Parameters:

func method that is calling constructor.
lineNum line number of file.
err standard error code.
extendedError extended information regarding exception (optional)

Definition at line 208 of file resmgr.svn-base.QNXAPI::DeviceFatalException::DeviceFatalException (const char * func, int lineNum, int err, const char * extendedError = "None available") [inline]

Build a new **DeviceFatalException**.

Parameters:

func method that is calling constructor.
lineNum line number of file.
err standard error code.
extendedError extended information regarding exception (optional)

Definition at line 208 of file resmgr.

Member Function Documentation

const std::string& QNXAPI::DeviceFatalException::ExtendedErrorDescription (void) [inline]

Retrieve extended information regarding exception.

Returns:

std::string with extended information about exception or "None Available".

Definition at line 219 of file resmgr.svn-base.const std::string& QNXAPI::DeviceFatalException::ExtendedErrorDescription (void) [inline]

Retrieve extended information regarding exception.

Returns:

std::string with extended information about exception or "None Available".

Definition at line 219 of file resmgr.

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

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

PosixAPI::GetTimeFailure Class Reference

GetTimeFailure exception.

Inherits std::exception, and std::exception.

Inheritance diagram for PosixAPI::GetTimeFailure:

Collaboration diagram for PosixAPI::GetTimeFailure:

Public Member Functions

- **GetTimeFailure** (int err)
  *Construct a **GetTimeFailure** class.*

- virtual const char * **what** (void) throw ()
  *Get the textual description of the error, using the standard **what()** method.*

- **GetTimeFailure** (int err)
  *Construct a **GetTimeFailure** class.*

- virtual const char * **what** (void) throw ()
  *Get the textual description of the error, using the standard **what()** method.*

Detailed Description

GetTimeFailure exception.

Exception encountered while attempting to obtain time.

Definition at line 104 of file except.svn-base.

Constructor & Destructor Documentation

PosixAPI::GetTimeFailure::GetTimeFailure (int* err*) [inline]

Construct a **GetTimeFailure** class.

Parameters:

err standard error code.

Definition at line 112 of file except.svn-base.PosixAPI::GetTimeFailure::GetTimeFailure (int* err*) [inline]

Construct a **GetTimeFailure** class.

Parameters:

err standard error code.

Definition at line 112 of file except.

Member Function Documentation

virtual const char* PosixAPI::GetTimeFailure::what (void) throw () [inline, virtual]

Get the textual description of the error, using the standard **what()** method.

Returns:

textual representation of the error.

Definition at line 121 of file except.svn-base.virtual const char* PosixAPI::GetTimeFailure::what (void) throw () [inline, virtual]

Get the textual description of the error, using the standard **what()** method.

Returns:

textual representation of the error.

Definition at line 121 of file except.

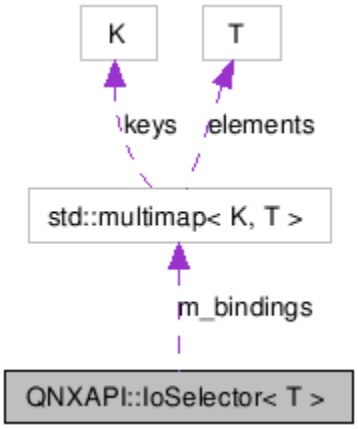
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/except.svn-base
- include/qfc/except

QNXAPI::IoSelector< T > Class Template Reference

IoSelector class.

Collaboration diagram for QNXAPI::IoSelector< T >:



Public Types

- typedef sigc::slot< T *, dispatch_t * > **ContextAlloc**
- typedef sigc::slot< void, T * > **ContextFree**
- typedef sigc::slot< void, T * > **ContextBlock**
- typedef std::multimap< int, **Binding** * > **BindingMap**
  A binding map.

- typedef sigc::slot< T *, dispatch_t * > **ContextAlloc**
- typedef sigc::slot< void, T * > **ContextFree**
- typedef sigc::slot< void, T * > **ContextBlock**
- typedef std::multimap< int, **Binding** * > **BindingMap**
  A binding map.

Public Member Functions

- **IoSelector** (**BindingMap** &bindings)
  *Construct an instance of **IoSelector**.*

- **IoSelector** (**BindingMap** &bindings, dispatch_t *dispatch, **ContextAlloc** contextAlloc, **ContextFree** contextFree, **ContextBlock** contextBlock)
  *Construct an instance of **IoSelector**.*

- **BindingMap** & **Bindings** (void) throw ()
  Obtain reference to underlying bindings.

- int **Connect** (void) throw ()
  Obtain a connection to the IoSelector's underlying channel.

- void **Block** (void)
  *Block the **IoSelector** waiting for events (as described by the bindings) to occur.*

- void **Dispatch** (dispatch_t *dispatch) throw ()
  *Set the dispatch for the **IoSelector**.*

- virtual **~IoSelector** ()
  *Destroy an instance of **IoSelector**.*

- **IoSelector** (**BindingMap** &bindings)
  *Construct an instance of **IoSelector**.*

- **IoSelector** (**BindingMap** &bindings, dispatch_t *dispatch, **ContextAlloc** contextAlloc, **ContextFree** contextFree, **ContextBlock** contextBlock)
  *Construct an instance of **IoSelector**.*

- **BindingMap** & **Bindings** (void) throw ()
  Obtain reference to underlying bindings.

- int **Connect** (void) throw ()
  Obtain a connection to the IoSelector's underlying channel.

- void **Block** (void)
  *Block the **IoSelector** waiting for events (as described by the bindings) to occur.*

- void **Dispatch** (dispatch_t *dispatch) throw ()
  *Set the dispatch for the **IoSelector**.*

- virtual **~IoSelector** ()
  *Destroy an instance of **IoSelector**.*

Classes

- class **Binding**
  ***Binding** class.*

- class **ContextAllocateFailure**
  ContextAllocFailure class.

- class **DispatchCreateFailure**
  ***DispatchCreateFailure** class.*

- class **DispatchNotAllocatedFailure**
  ***DispatchNotAllocatedFailure** class.*

- class **MessageBindFailure**
  MessageBindFailure *class.*

- class **MessageBinding**
  MessageBinding *class.*

- class **PulseBindFailure**
  PulseBindFailure *class.*

- class **PulseBinding**
  PulseBinding *class.*

- class **ResmgrAttachFailure**
  ResmgrAttachFailure *class.*

- class **SelectBindFailure**
  SelectBindFailure *class.*

- class **SelectBinding**
  SelectBinding *class.*

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >

IoSelector class.

Implements a single point selector for input/output. Client registers callbacks for different variants of I/O:

- Messages (i.e. incoming messages)
-

- Select notifications (i.e. readability or writeability on a socket)
-
- Pulses (i.e. incoming generic notifications)

Author:

Rennie Allen `rennieallen@gmail.com`

Definition at line 45 of file ioselector.svn-base.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::IoSelector (BindingMap & bindings) [inline]

Construct an instance of **IoSelector**.

Parameters:

bindings reference to a BindingMap of bindings to a particular **IoSelector** event.

Definition at line 370 of file ioselector.svn-base.template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::IoSelector (BindingMap & bindings, dispatch_t * dispatch, ContextAlloc contextAlloc, ContextFree contextFree, ContextBlock contextBlock) [inline]

Construct an instance of **IoSelector**.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

dispatch pointer to an allocated dispatch context (can supply NULL and later set with Dispatch method)

Definition at line 398 of file ioselector.svn-base.template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::IoSelector (BindingMap & bindings) [inline]

Construct an instance of **IoSelector**.

Parameters:

bindings reference to a BindingMap of bindings to a particular **IoSelector** event.

Definition at line 370 of file ioselector.template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::IoSelector (BindingMap & bindings, dispatch_t * dispatch, ContextAlloc contextAlloc, ContextFree contextFree, ContextBlock contextBlock) [inline]

Construct an instance of **IoSelector**.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters:

dispatch pointer to an allocated dispatch context (can supply NULL and later set with Dispatch method)

Definition at line 398 of file ioselector.

Member Function Documentation

template<typename T = dispatch_context_t> BindingMap& QNXAPI::IoSelector< T >::Bindings (void) throw () [inline]

Obtain reference to underlying bindings.

Returns:

reference to a BindingMap of bindings to this **IoSelector**.

Definition at line 411 of file ioselector.svn-base.template<typename T = dispatch_context_t> int QNXAPI::IoSelector< T >::Connect (void) throw () [inline]

Obtain a connection to the IoSelector's underlying channel.

Returns:

connection id (coid) attached to the IoSelector's channel

Definition at line 420 of file ioselector.svn-base.template<typename T = dispatch_context_t> BindingMap& QNXAPI::IoSelector< T >::Bindings (void) throw () [inline]

Obtain reference to underlying bindings.

Returns:

reference to a BindingMap of bindings to this **IoSelector**.

Definition at line 411 of file ioselector.template<typename T = dispatch_context_t> int QNXAPI::IoSelector< T >::Connect (void) throw () [inline]

Obtain a connection to the IoSelector's underlying channel.

Returns:

connection id (coid) attached to the IoSelector's channel

Definition at line 420 of file ioselector.

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

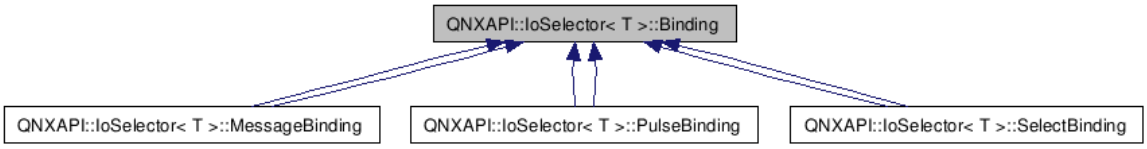
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::Binding Class Reference

Binding class.

Inherited by **QNXAPI::IoSelector< T >::MessageBinding**, **QNXAPI::IoSelector< T >::MessageBinding**, **QNXAPI::IoSelector< T >::PulseBinding**, **QNXAPI::IoSelector< T >::PulseBinding**, **QNXAPI::IoSelector< T >::SelectBinding**, and **QNXAPI::IoSelector< T >::SelectBinding**.

Inheritance diagram for QNXAPI::IoSelector< T >::Binding:



Public Member Functions

- **Binding** (int cookie, sigc::slot< void, int, void * > callback)
  *Construct an instance of **Binding**.*

- virtual void **Dispatch** (int cookie, void *data)
  Dispatch the bindings callback.

- virtual void **Bind** (dispatch_t *dispatch)=0
  Bind this binding to the dispatch.

- virtual **~Binding** (void)
  *Destroy an instance of **Binding**.*

- **Binding** (int cookie, sigc::slot< void, int, void * > callback)
  *Construct an instance of **Binding**.*

- virtual void **Dispatch** (int cookie, void *data)
  Dispatch the bindings callback.

- virtual void **Bind** (dispatch_t *dispatch)=0
  Bind this binding to the dispatch.

- virtual **~Binding** (void)
  *Destroy an instance of **Binding**.*

Protected Attributes

- int **m_cookie**
- sigc::signal< void, int, void * > **m_callback**
- sigc::signal< void, int, void * > **m_callback**

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::Binding

Binding class.

This abstract class implements a binding to a generic **IoSelector** event.

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 183 of file ioselector.svn-base.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::Binding::Binding (int* **cookie, *sigc::slot< void, int, void * >* **callback*) *[inline]

Construct an instance of **Binding**.

Parameters:

cookie id of binding
callback Callback to invoke when binding event occurs

Definition at line 191 of file ioselector.svn-base.

References QNXAPI::IoSelector< T >::Binding::m_callback.template<typename T = dispatch_context_t>
QNXAPI::IoSelector< T >::Binding::Binding (int* **cookie**, *sigc::slot< void, int, void * >* **callback**) [inline]

Construct an instance of **Binding**.

Parameters:

cookie id of binding
callback Callback to invoke when binding event occurs

Definition at line 191 of file ioselector.
References QNXAPI::IoSelector< T >::Binding::m_callback.

Member Function Documentation

template<typename T = dispatch_context_t> virtual void QNXAPI::IoSelector< T >::Binding::Dispatch (int* **cookie**, *void * **data**) [inline, virtual]

Dispatch the bindings callback.

Parameters:

cookie token associated with binding
data pointer to data (i.e. pointer binding instance)

Definition at line 201 of file ioselector.svn-base.

References QNXAPI::IoSelector< T >::Binding::m_callback.template<typename T = dispatch_context_t>
virtual void QNXAPI::IoSelector< T >::Binding::Bind (dispatch_t * **dispatch**) [pure virtual]

Bind this binding to the dispatch.

Parameters:

dispatch Dispatch point to bind to

Implemented in** QNXAPI::IoSelector< T >::SelectBinding **(p.98)**, QNXAPI::IoSelector< T >::MessageBinding **(p.87)**, QNXAPI::IoSelector< T >::PulseBinding **(p.91)**, QNXAPI::IoSelector< T >::SelectBinding **(p.98)**, QNXAPI::IoSelector< T >::MessageBinding **(p.87)**, **and** QNXAPI::IoSelector< T >::PulseBinding **(p.91)**.template<typename T = dispatch_context_t> virtual void QNXAPI::IoSelector< T >::Binding::Dispatch (int* **cookie**, *void * **data**) [inline, virtual]

Dispatch the bindings callback.

Parameters:

cookie token associated with binding

data pointer to data (i.e. pointer binding instance)

Definition at line 201 of file ioselector.

References QNXAPI::IoSelector< T >::Binding::m_callback.template<typename T = dispatch_context_t>
virtual void QNXAPI::IoSelector< T >::Binding::Bind (dispatch_t * dispatch) [pure virtual]

Bind this binding to the dispatch.

Parameters:

dispatch Dispatch point to bind to

Implemented in **QNXAPI::IoSelector< T >::SelectBinding** (*p.98*), **QNXAPI::IoSelector< T >::MessageBinding** (*p.87*), **QNXAPI::IoSelector< T >::PulseBinding** (*p.91*), **QNXAPI::IoSelector< T >::SelectBinding** (*p.98*), **QNXAPI::IoSelector< T >::MessageBinding** (*p.87*), and **QNXAPI::IoSelector< T >::PulseBinding** (*p.91*).

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

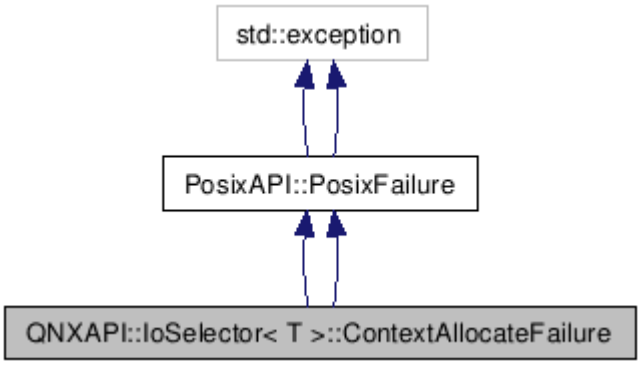
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::ContextAllocateFailure Class Reference

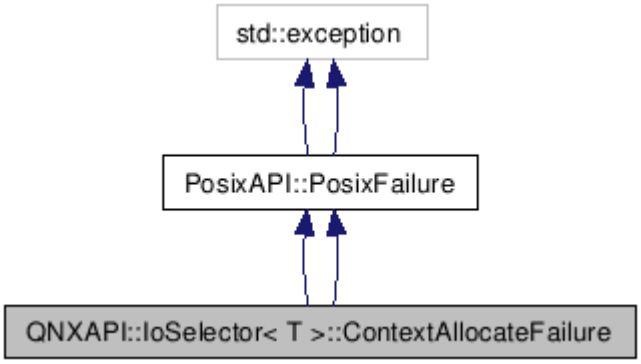
ContextAllocFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::IoSelector< T >::ContextAllocateFailure:



Collaboration diagram for QNXAPI::IoSelector< T >::ContextAllocateFailure:



Public Member Functions

- **ContextAllocateFailure** (int err)
  *Construct a **ContextAllocateFailure** class.*

- **ContextAllocateFailure** (int err)
  *Construct a **ContextAllocateFailure** class.*

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::ContextAllocateFailure

ContextAllocFailure class.

Thrown on a failure to allocate a blocking context.

Definition at line 109 of file ioselector.svn-base.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::ContextAllocateFailure::ContextAllocateFailure (int err) [inline]

Construct a **ContextAllocateFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 117 of file ioselector.svn-base.template<typename T = dispatch_context_t>
QNXAPI::IoSelector< T >::ContextAllocateFailure::ContextAllocateFailure (int err*) [inline]*

Construct a **ContextAllocateFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 117 of file ioselector.

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

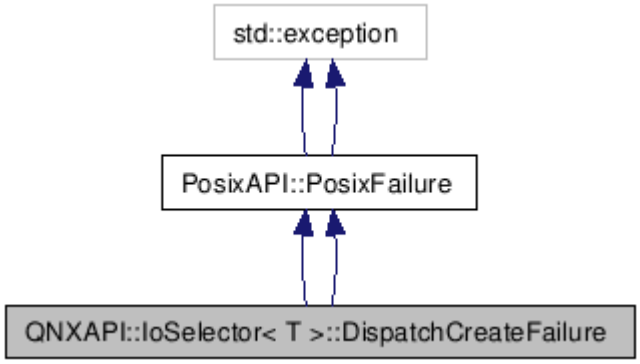
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::DispatchCreateFailure Class Reference

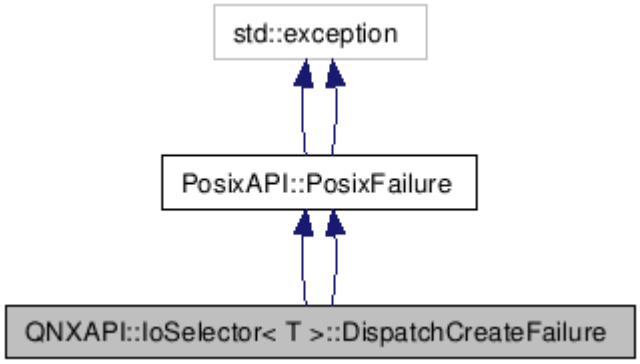
DispatchCreateFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::IoSelector< T >::DispatchCreateFailure:



Collaboration diagram for QNXAPI::IoSelector< T >::DispatchCreateFailure:



Public Member Functions

- **DispatchCreateFailure** (int err)
  *Construct a **DispatchCreateFailure** class.*

- **DispatchCreateFailure** (int err)
  *Construct a **DispatchCreateFailure** class.*

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::DispatchCreateFailure

DispatchCreateFailure class.

Thrown on a failure to allocate a dispatch point.

Definition at line 56 of file ioselector.svn-base.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::DispatchCreateFailure::DispatchCreateFailure (int err) [inline]

Construct a **DispatchCreateFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 64 of file ioselector.svn-base.template<typename T = dispatch_context_t>
QNXAPI::IoSelector< T >::DispatchCreateFailure::DispatchCreateFailure (int err*) [inline]*

Construct a **DispatchCreateFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 64 of file ioselector.

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

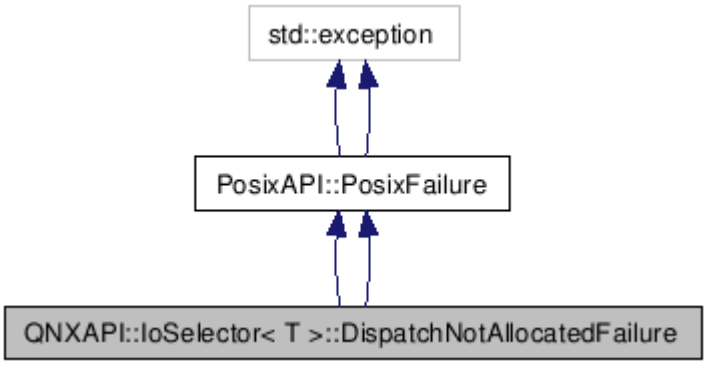
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::DispatchNotAllocatedFailure Class Reference

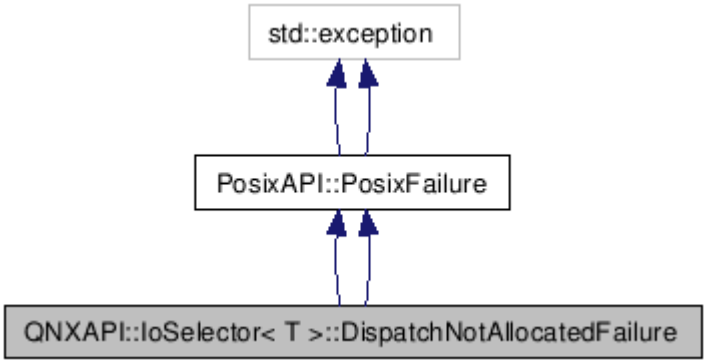
DispatchNotAllocatedFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::IoSelector< T >::DispatchNotAllocatedFailure:



Collaboration diagram for QNXAPI::IoSelector< T >::DispatchNotAllocatedFailure:



Public Member Functions

- **DispatchNotAllocatedFailure** (void)
  *Construct a **DispatchCreateFailure** class.*

- **DispatchNotAllocatedFailure** (void)
  *Construct a **DispatchCreateFailure** class.*

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::DispatchNotAllocatedFailure

DispatchNotAllocatedFailure class.

Thrown on a failure to allocate a dispatch point.

Definition at line 74 of file ioselector.svn-base.

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

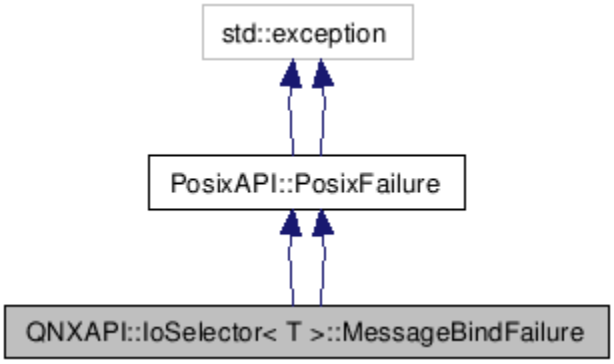
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::MessageBindFailure Class Reference

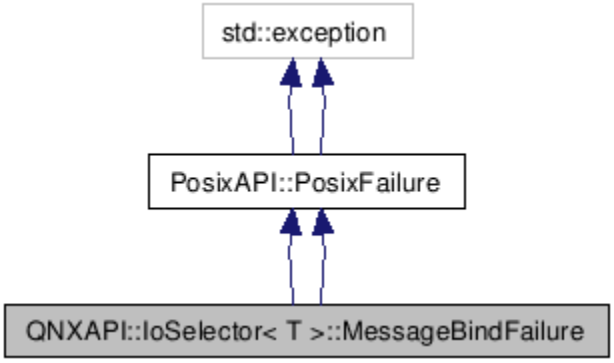
MessageBindFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::IoSelector< T >::MessageBindFailure:



Collaboration diagram for QNXAPI::IoSelector< T >::MessageBindFailure:



Public Member Functions

- **MessageBindFailure** (int err)
  *Construct a **MessageBindFailure** class.*

- **MessageBindFailure** (int err)
  *Construct a **MessageBindFailure** class.*

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::MessageBindFailure

MessageBindFailure class.

Thrown on a failure to bind to the dispatches message callback.

Definition at line 145 of file ioselector.svn-base.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::MessageBindFailure::MessageBindFailure (int err) [inline]

Construct a **MessageBindFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 153 of file ioselector.svn-base.template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::MessageBindFailure::MessageBindFailure (int err) [inline]

Construct a **MessageBindFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 153 of file ioselector.

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

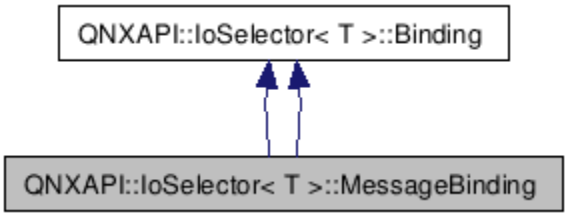
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::MessageBinding Class Reference

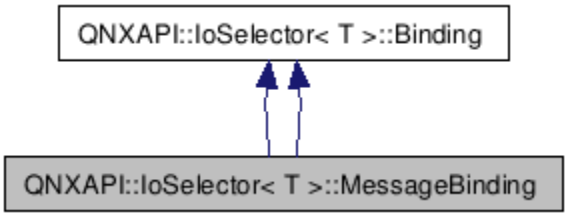
MessageBinding class.

Inherits **QNXAPI::IoSelector< T >::Binding**, and **QNXAPI::IoSelector< T >::Binding**.

Inheritance diagram for QNXAPI::IoSelector< T >::MessageBinding:



Collaboration diagram for QNXAPI::IoSelector< T >::MessageBinding:



Public Member Functions

- **MessageBinding** (int start, sigc::slot< void, int, void * > callback, uint16_t end)
  *Construct an instance of **MessageBinding**.*

- virtual void **Bind** (dispatch_t *dp)
- **MessageBinding** (int start, sigc::slot< void, int, void * > callback, uint16_t end)
  *Construct an instance of **MessageBinding**.*

- virtual void **Bind** (dispatch_t *dp)

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::MessageBinding

MessageBinding class.

This concrete class implements a binding to a **IoSelector** message event.

Definition at line 286 of file ioselector.svn-base.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::MessageBinding::MessageBinding (int start, sigc::slot< void, int, void * > callback, uint16_t end) [inline]

Construct an instance of **MessageBinding**.

Parameters:

start first message number of message id range
callback Callback to invoke when message (within range) received
end last message number of message id range

Definition at line 296 of file ioselector.svn-base.template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::MessageBinding::MessageBinding (int* start, *sigc::slot< void, int, void * > callback, *uint16_t* end) [inline]

Construct an instance of **MessageBinding**.

Parameters:

start first message number of message id range
callback Callback to invoke when message (within range) received
end last message number of message id range

Definition at line 296 of file ioselector.

Member Function Documentation

template<typename T = dispatch_context_t> virtual void QNXAPI::IoSelector< T >::MessageBinding::Bind (dispatch_t * dp) [inline, virtual]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Implements **QNXAPI::IoSelector< T >::Binding** (*p.78*).

Definition at line 302 of file ioselector.svn-base.

References QNXAPI::IoSelector< T >::Binding::m_cookie.template<typename T = dispatch_context_t> virtual void QNXAPI::IoSelector< T >::MessageBinding::Bind (dispatch_t * dp) [inline, virtual]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Implements **QNXAPI::IoSelector< T >::Binding** (*p.78*).

Definition at line 302 of file ioselector.

References QNXAPI::IoSelector< T >::Binding::m_cookie.

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

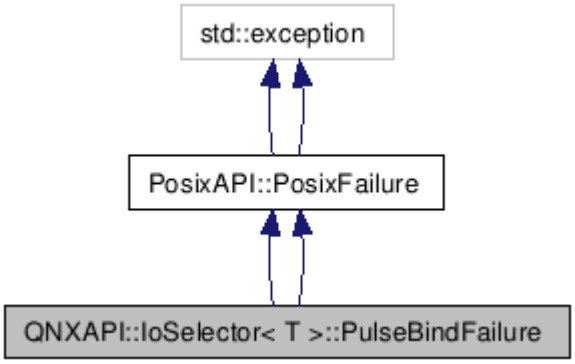
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::PulseBindFailure Class Reference

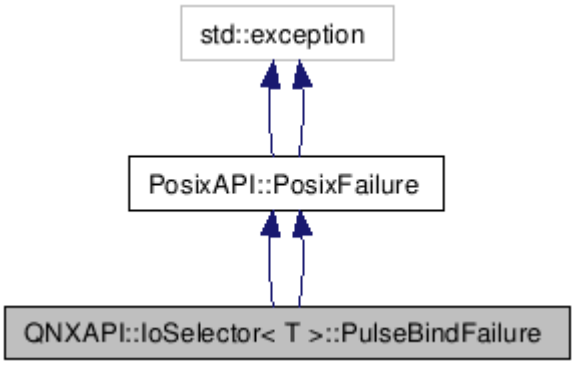
PulseBindFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::IoSelector< T >::PulseBindFailure:



Collaboration diagram for QNXAPI::IoSelector< T >::PulseBindFailure:



Public Member Functions

- **PulseBindFailure** (int err)
  *Construct a **PulseBindFailure** class.*

- **PulseBindFailure** (int err)
  *Construct a **PulseBindFailure** class.*

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::PulseBindFailure

PulseBindFailure class.

Thrown on a failure to bind to the dispatches pulse callback.

Definition at line 163 of file ioselector.svn-base.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::PulseBindFailure::PulseBindFailure (int err) [inline]

Construct a **PulseBindFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 171 of file ioselector.svn-base.template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::PulseBindFailure::PulseBindFailure (int err) [inline]

Construct a **PulseBindFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 171 of file ioselector.

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

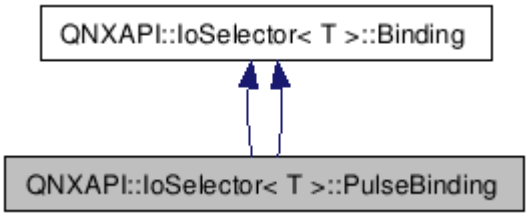
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::PulseBinding Class Reference

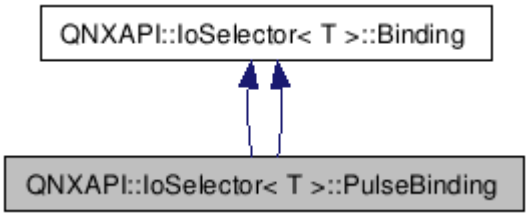
PulseBinding class.

Inherits **QNXAPI::IoSelector< T >::Binding**, and **QNXAPI::IoSelector< T >::Binding**.

Inheritance diagram for QNXAPI::IoSelector< T >::PulseBinding:



Collaboration diagram for QNXAPI::IoSelector< T >::PulseBinding:



Public Member Functions

- **PulseBinding** (int code, sigc::slot< void, int, void * > callback)
  *Construct an instance of **PulseBinding**.*

- virtual void **Bind** (dispatch_t *dp)
- **PulseBinding** (int code, sigc::slot< void, int, void * > callback)
  *Construct an instance of **PulseBinding**.*

- virtual void **Bind** (dispatch_t *dp)

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::PulseBinding

PulseBinding class.

This concrete class implements a binding to a **IoSelector** pulse event.

Definition at line 327 of file ioselector.svn-base.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::PulseBinding::PulseBinding (int* **code, *sigc::slot< void, int, void * >* **callback) *[inline]

Construct an instance of **PulseBinding**.

Parameters:

code code of pulse
callback Callback to invoke when pulse received

Definition at line 336 of file ioselector.svn-base.template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::PulseBinding::PulseBinding (int code, sigc::slot< void, int, void * > callback) [inline]

Construct an instance of **PulseBinding**.

Parameters:

code code of pulse
callback Callback to invoke when pulse received

Definition at line 336 of file ioselector.

Member Function Documentation

template<typename T = dispatch_context_t> virtual void QNXAPI::IoSelector< T >::PulseBinding::Bind (dispatch_t * dp) [inline, virtual]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Implements **QNXAPI::IoSelector< T >::Binding** (*p.78*).

Definition at line 342 of file ioselector.svn-base.template<typename T = dispatch_context_t> virtual void QNXAPI::IoSelector< T >::PulseBinding::Bind (dispatch_t * dp) [inline, virtual]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Implements **QNXAPI::IoSelector< T >::Binding** (*p.78*).

Definition at line 342 of file ioselector.

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

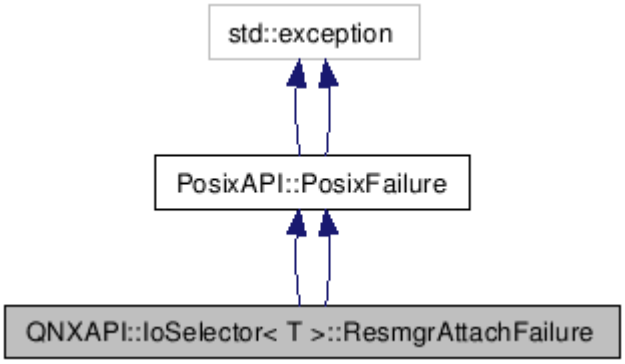
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::ResmgrAttachFailure Class Reference

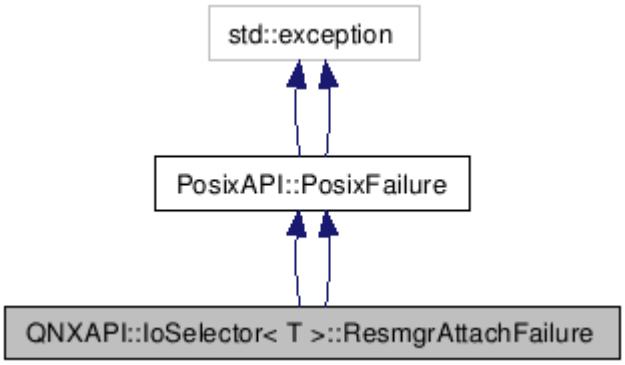
ResmgrAttachFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::IoSelector< T >::ResmgrAttachFailure:



Collaboration diagram for QNXAPI::IoSelector< T >::ResmgrAttachFailure:



Public Member Functions

- **ResmgrAttachFailure** (int err)
  *Construct a **ResmgrAttachFailure** class.*

- **ResmgrAttachFailure** (int err)
  *Construct a **ResmgrAttachFailure** class.*

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::ResmgrAttachFailure

ResmgrAttachFailure class.

Thrown on a failure to create a channel for the dispatch point.

Definition at line 91 of file ioselector.svn-base.

Constructor & Destructor Documentation

```
template<typename T = dispatch_context_t> QNXAPI::IoSelector< T
>::ResmgrAttachFailure::ResmgrAttachFailure (int err) [inline]
```

Construct a **ResmgrAttachFailure** class.

Parameters:

err Posix error code (from errno.h)

```
Definition at line 99 of file ioselector.svn-base.template<typename T = dispatch_context_t>
QNXAPI::IoSelector< T >::ResmgrAttachFailure::ResmgrAttachFailure (int err) [inline]
```

Construct a **ResmgrAttachFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 99 of file ioselector.

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

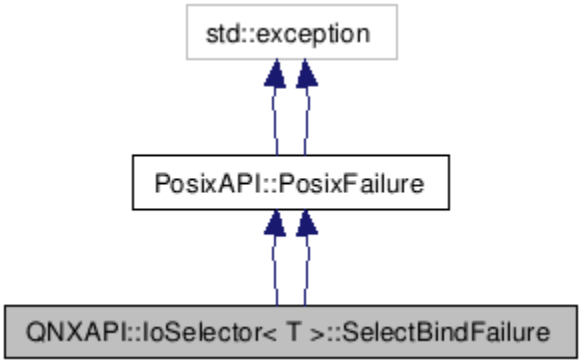
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::SelectBindFailure Class Reference

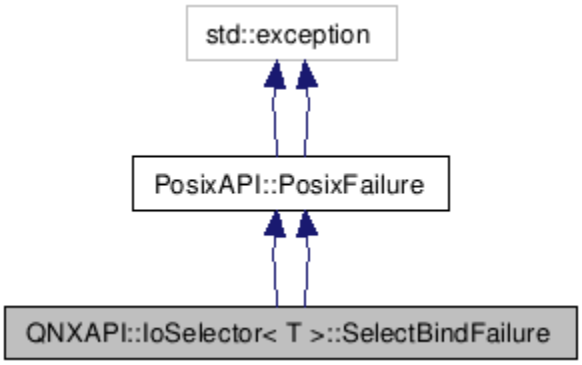
SelectBindFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::IoSelector< T >::SelectBindFailure:



Collaboration diagram for QNXAPI::IoSelector< T >::SelectBindFailure:



Public Member Functions

- **SelectBindFailure** (int err)
  *Construct a **SelectBindFailure** class.*

- **SelectBindFailure** (int err)
  *Construct a **SelectBindFailure** class.*

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::SelectBindFailure

SelectBindFailure class.

Thrown on a failure to bind to the dispatches select callback.

Definition at line 127 of file ioselector.svn-base.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::SelectBindFailure::SelectBindFailure (int err) [inline]

Construct a **SelectBindFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 135 of file ioselector.svn-base.template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::SelectBindFailure::SelectBindFailure (int err) [inline]

Construct a **SelectBindFailure** class.

Parameters:

err Posix error code (from errno.h)

Definition at line 135 of file ioselector.

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

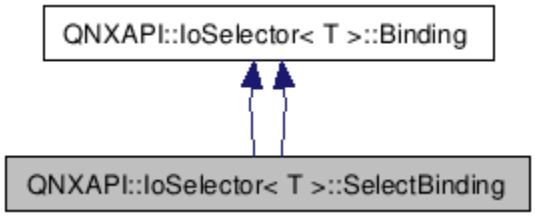
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

QNXAPI::IoSelector< T >::SelectBinding Class Reference

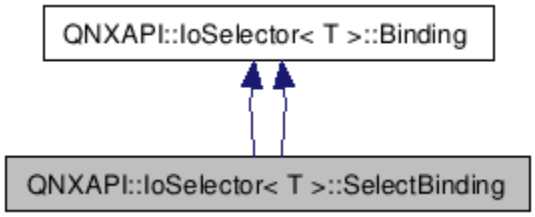
SelectBinding class.

Inherits **QNXAPI::IoSelector< T >::Binding**, and **QNXAPI::IoSelector< T >::Binding**.

Inheritance diagram for QNXAPI::IoSelector< T >::SelectBinding:



Collaboration diagram for QNXAPI::IoSelector< T >::SelectBinding:



Public Types

- enum **Triggers**
  Triggers for a select binding.

- enum **Triggers**
  Triggers for a select binding.

Public Member Functions

- **SelectBinding** (int fd, sigc::slot< void, int, void * > callback, **Triggers** flags=(**Triggers**)(Read|ReArm))
  *Construct an instance of **SelectBinding**.*

- virtual void **Bind** (dispatch_t *dp)
- **SelectBinding** (int fd, sigc::slot< void, int, void * > callback, **Triggers** flags=(**Triggers**)(Read|ReArm))
  *Construct an instance of **SelectBinding**.*

- virtual void **Bind** (dispatch_t *dp)

Detailed Description

template<typename T = dispatch_context_t> class QNXAPI::IoSelector< T >::SelectBinding

SelectBinding class.

This concrete class implements a binding to a **IoSelector** select event.

Definition at line 230 of file ioselector.svn-base.

Member Enumeration Documentation

template<typename T = dispatch_context_t> enum QNXAPI::IoSelector::SelectBinding::Triggers

Triggers for a select binding.

- Write call binding when descriptor is writable
-
- read call binding when descriptor is readable
-
- ReArm automatically ReArm following a trigger
-

Definition at line 240 of file ioselector.svn-base.template<typename T = dispatch_context_t> enum QNXAPI::IoSelector::SelectBinding::Triggers

Triggers for a select binding.

- Write call binding when descriptor is writable
-
- read call binding when descriptor is readable
-
- ReArm automatically ReArm following a trigger
-

Definition at line 240 of file ioselector.

Constructor & Destructor Documentation

template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::SelectBinding::SelectBinding (int* fd*, *sigc::slot< void, int, void * >* callback, *Triggers* flags = *(Triggers)(Read|ReArm)) [inline]

Construct an instance of **SelectBinding**.

Parameters:

fd file descriptor for select binding
callback Callback to invoke when select event received
flags Triggers for the select event

Definition at line 253 of file ioselector.svn-base.template<typename T = dispatch_context_t> QNXAPI::IoSelector< T >::SelectBinding::SelectBinding (int* fd*, *sigc::slot< void, int, void * >* callback, *Triggers* flags = *(Triggers)(Read|ReArm)) [inline]

Construct an instance of **SelectBinding**.

Parameters:

fd file descriptor for select binding
callback Callback to invoke when select event received
flags Triggers for the select event

Definition at line 253 of file ioselector.

Member Function Documentation

template<typename T = dispatch_context_t> virtual void QNXAPI::IoSelector< T >::SelectBinding::Bind (dispatch_t * dp) [inline, virtual]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Implements **QNXAPI::IoSelector< T >::Binding** (*p.78*).

Definition at line 259 of file ioselector.svn-base.

References QNXAPI::IoSelector< T >::Binding::m_cookie.template<typename T = dispatch_context_t> virtual void QNXAPI::IoSelector< T >::SelectBinding::Bind (dispatch_t * dp) [inline, virtual]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Implements **QNXAPI::IoSelector< T >::Binding** (*p.78*).

Definition at line 259 of file ioselector.

References QNXAPI::IoSelector< T >::Binding::m_cookie.

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

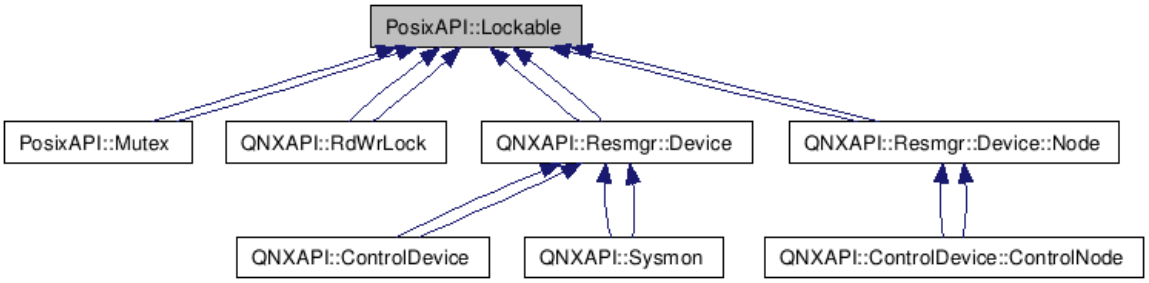
- include/qfc/.svn/text-base/ioselector.svn-base
- include/qfc/ioselector

PosixAPI::Lockable Class Reference

Interface: **Lockable**.

Inherited by **PosixAPI::Mutex**, **PosixAPI::Mutex**, **QNXAPI::RdWrLock**, **QNXAPI::RdWrLock**, **QNXAPI::Resmgr::Device**, **QNXAPI::Resmgr::Device**, **QNXAPI::Resmgr::Device::Node**, and **QNXAPI::Resmgr::Device::Node**.

Inheritance diagram for PosixAPI::Lockable:



Public Types

- enum **Mode**
  Lock mode.

- enum **Mode**
  Lock mode.

Detailed Description

Interface: **Lockable**.

Definition at line 52 of file interfaces.svn-base.

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

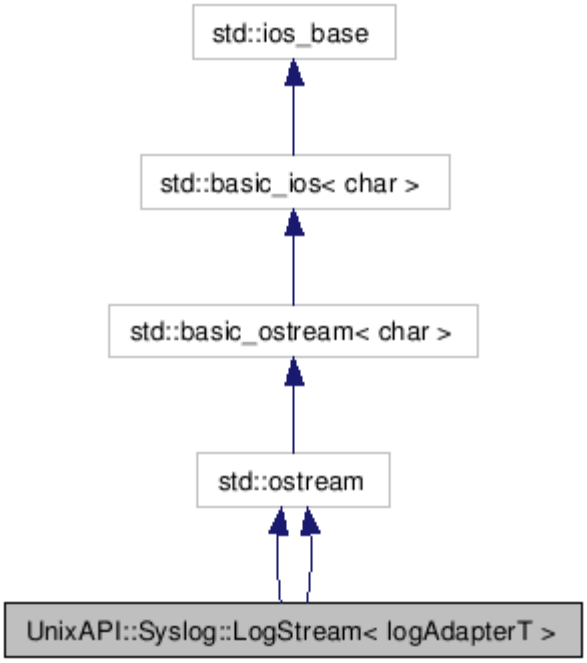
- include/qfc/.svn/text-base/interfaces.svn-base
- include/qfc/interfaces

UnixAPI::Syslog::LogStream< logAdapterT > Class Template Reference

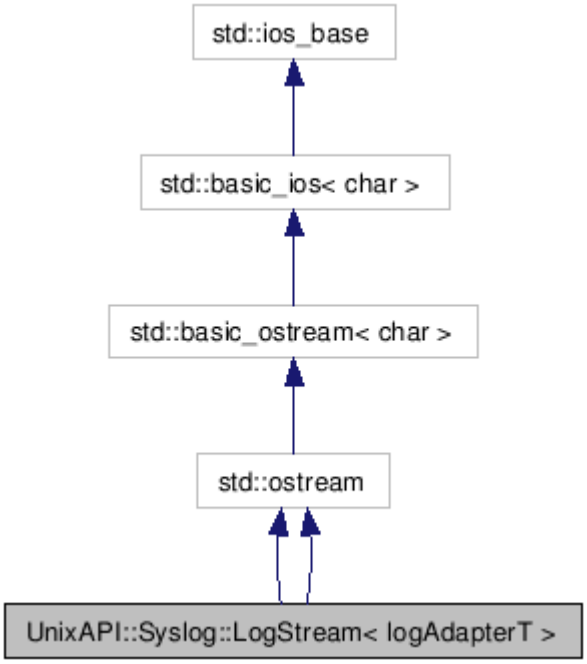
The **Syslog LogStream** class a logging adapter. Data from the stream buffer is supplied to the logging system adapter. By default to logging system adapters are provided:.

Inherits std::ostream, and std::ostream.

Inheritance diagram for UnixAPI::Syslog::LogStream< logAdapterT >:



Collaboration diagram for UnixAPI::Syslog::LogStream< logAdapterT >:



Public Member Functions

- **LogStream** ()
- **~LogStream** ()
- void **open** (const char *ident, **LogOption** opt=NoDelay, **Facility** fac=User)
- **LogStream** ()

- **~LogStream** ()
- void **open** (const char *ident, **LogOption** opt=NoDelay, **Facility** fac=User)

Detailed Description

template<class logAdapterT = SyslogAdapter> class UnixAPI::Syslog::LogStream< logAdapterT >

The **Syslog LogStream** class a logging adapter. Data from the stream buffer is supplied to the logging system adapter. By default to logging system adapters are provided:.

SyslogAdapter (syslogd). **SlogAdapter** (slogger).

Author:

rennieallen@gmail.com

Definition at line 155 of file syslog.svn-base.

Constructor & Destructor Documentation

template<class logAdapterT = SyslogAdapter> UnixAPI::Syslog::LogStream< logAdapterT >::LogStream () [inline]

Construct a **Syslog::LogStream**.

This is a template and if the log adapter is not supplied then the default is set to UnixAPI::Syslog::SyslogAdapter.

Definition at line 164 of file syslog.svn-base.template<class logAdapterT = SyslogAdapter> UnixAPI::Syslog::LogStream< logAdapterT >::~LogStream () [inline]

Destroy a **Syslog::LogStream**.

Definition at line 171 of file syslog.svn-base.template<class logAdapterT = SyslogAdapter> UnixAPI::Syslog::LogStream< logAdapterT >::LogStream () [inline]

Construct a **Syslog::LogStream**.

This is a template and if the log adapter is not supplied then the default is set to UnixAPI::Syslog::SyslogAdapter.

Definition at line 164 of file syslog.template<class logAdapterT = SyslogAdapter> UnixAPI::Syslog::LogStream< logAdapterT >::~LogStream () [inline]

Destroy a **Syslog::LogStream**.

Definition at line 171 of file syslog.

Member Function Documentation

template<class logAdapterT = SyslogAdapter> void UnixAPI::Syslog::LogStream< logAdapterT >::open (const char * ident, LogOption opt = NoDelay, Facility fac = User) [inline]

Open the **LogStream**.

Parameters:

ident 'C' style string identifying for this instance.
opt Logging option (see **UnixAPI::Syslog::LogOption**).
fac (see **UnixAPI::Syslog::Facility**).

Definition at line 184 of file syslog.svn-base.template<class logAdapterT = SyslogAdapter> void UnixAPI::Syslog::LogStream< logAdapterT >::open (const char * ident, LogOption opt = NoDelay, Facility fac = User) [inline]

Open the **LogStream**.

Parameters:

ident 'C' style string identifying for this instance.
opt Logging option (see **UnixAPI::Syslog::LogOption**).
fac (see **UnixAPI::Syslog::Facility**).

Definition at line 184 of file syslog.

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

- include/qfc/.svn/text-base/syslog.svn-base
- include/qfc/syslog

QNXAPI::MessageVector< T, V, R > Class Template Reference

Message vector class.

Public Member Functions

- **MessageVector** ()
  *Construct an instance of a **MessageVector** class.*

- virtual **~MessageVector** ()
  *Destroy an instance of a **MessageVector** class.*

- void **Add** (T item)
  Add a vector.

- void **Add** (void *ptr, size_t len)

Add a 'C' style vector entry.

- void **Clear** (void)
  Clear (erase) a message vector.

- virtual void **Read** (**Reception** &rcp, int offset=0)
  Read from a reception into the message vector.

- virtual void **Send** (**Connection** &conn)
  *Send from the message vector via a **Connection**.*

- virtual void **Send** (**Connection** &conn, **ReplyVector**< R, V > &rvect)
  *Send from the message vector via a **Connection**, and obtain a reply.*

- **MessageVector** ()
  *Construct an instance of a **MessageVector** class.*

- virtual **~MessageVector** ()
  *Destroy an instance of a **MessageVector** class.*

- void **Add** (T item)
  Add a vector.

- void **Add** (void *ptr, size_t len)
  Add a 'C' style vector entry.

- void **Clear** (void)
  Clear (erase) a message vector.

- virtual void **Read** (**Reception** &rcp, int offset=0)
  Read from a reception into the message vector.

- virtual void **Send** (**Connection** &conn)
  *Send from the message vector via a **Connection**.*

- virtual void **Send** (**Connection** &conn, **ReplyVector**< R, V > &rvect)
  *Send from the message vector via a **Connection**, and obtain a reply.*

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::MessageVector< T, V, R >

Message vector class.

Examples:

Serv/Serv.h.

Definition at line 633 of file ipc.svn-base.

Member Function Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::MessageVector< T, V, R >::Add (T* item*) *[inline]

Add a vector.

Parameters:

item a variant communicable type to add to vector.

Definition at line 655 of file ipc.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::MessageVector< T, V, R >::Add (void * ptr, size_t len) [inline]

Add a 'C' style vector entry.

Parameters:

ptr 'C' style void pointer to data
len Length of data pointed to by ptr

Definition at line 666 of file ipc.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::MessageVector< T, V, R >::Read (Reception & rcp, int offset = 0) [inline, virtual]

Read from a reception into the message vector.

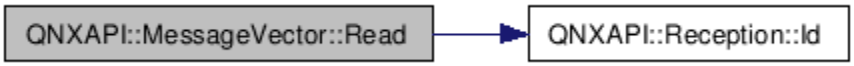
Parameters:

rcp reference to **Reception** to receive from.
offset offset to read from (**Reception** size automatically accounted for).

Definition at line 686 of file ipc.svn-base.

References QNXAPI::Reception::Id().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::MessageVector< T, V, R >::Send (Connection & conn) [inline, virtual]

Send from the message vector via a **Connection**.

Parameters:

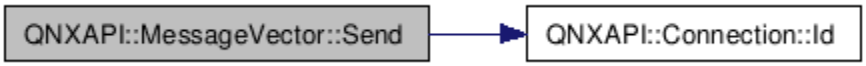
conn **Connection** to send through.

Definition at line 715 of file ipc.svn-base.

References QNXAPI::Connection::Id().

Referenced by QNXAPI::NameConnection< T, V, R >::Send().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::MessageVector< T, V, R >::Send (Connection & conn, ReplyVector< R, V > & rvect) [inline, virtual]

Send from the message vector via a **Connection**, and obtain a reply.

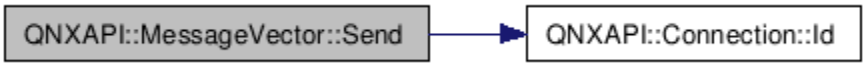
Parameters:

conn **Connection** to send through.
rvect other **MessageVector** to receive reply.

Definition at line 746 of file ipc.svn-base.

References QNXAPI::Connection::Id(), and QNXAPI::ReplyVector< T, V >::m_vect.

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::MessageVector< T, V, R >::Add (T* item) [inline]

Add a vector.

Parameters:

item a variant communicable type to add to vector.

Definition at line 655 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::MessageVector< T, V, R >::Add (void * ptr, size_t len) [inline]

Add a 'C' style vector entry.

Parameters:

ptr 'C' style void pointer to data
len Length of data pointed to by ptr

Definition at line 666 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::MessageVector< T, V, R >::Read (Reception & rcp, int offset = 0) [inline, virtual]

Read from a reception into the message vector.

Parameters:

rcp reference to **Reception** to receive from.
offset offset to read from (**Reception** size automatically accounted for).

Definition at line 686 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::MessageVector< T, V, R >::Send (Connection & conn) [inline, virtual]

Send from the message vector via a **Connection**.

Parameters:

conn **Connection** to send through.

Definition at line 715 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void QNXAPI::MessageVector< T, V, R >::Send (Connection & conn, ReplyVector< R, V > & rvect) [inline, virtual]

Send from the message vector via a **Connection**, and obtain a reply.

Parameters:

conn **Connection** to send through.
rvect other **MessageVector** to receive reply.

Definition at line 746 of file ipc.

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

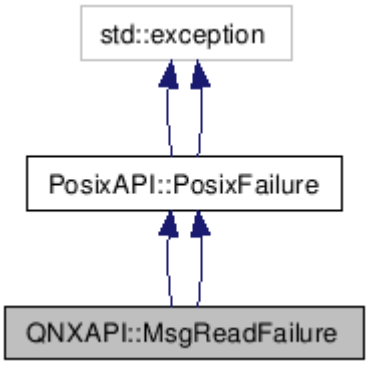
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::MsgReadFailure Class Reference

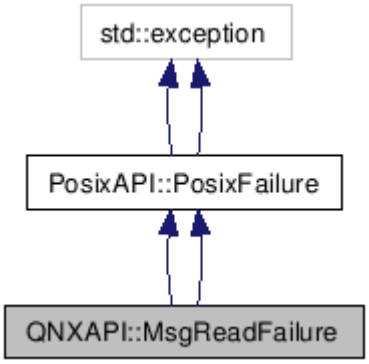
Message could not be read.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::MsgReadFailure:



Collaboration diagram for QNXAPI::MsgReadFailure:



Public Member Functions

- **MsgReadFailure** (int err)
  *Construct an instance of a **MsgReadFailure** exception.*

- **MsgReadFailure** (int err)
  *Construct an instance of a **MsgReadFailure** exception.*

Detailed Description

Message could not be read.

Thrown when an error occurs as the result of a message read.

Definition at line 123 of file ipc.svn-base.

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

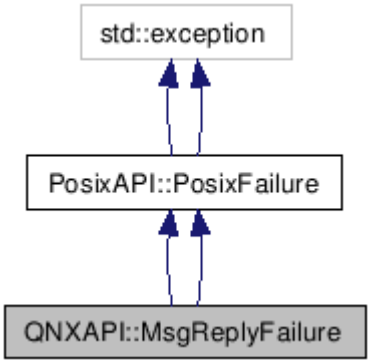
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::MsgReplyFailure Class Reference

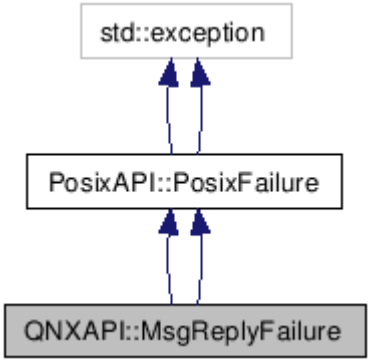
Message could not be replied to.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::MsgReplyFailure:



Collaboration diagram for QNXAPI::MsgReplyFailure:



Public Member Functions

- **MsgReplyFailure** (int err)
  *Construct an instance of a **MsgReplyFailure** exception.*

- **MsgReplyFailure** (int err)
  *Construct an instance of a **MsgReplyFailure** exception.*

Detailed Description

Message could not be replied to.

Thrown when an error occurs as the result of a message reply.

Definition at line 171 of file ipc.svn-base.

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

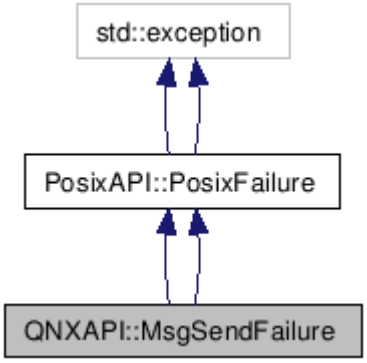
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::MsgSendFailure Class Reference

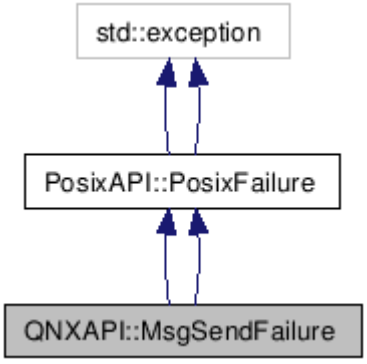
Message could not be sent.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::MsgSendFailure:



Collaboration diagram for QNXAPI::MsgSendFailure:



Public Member Functions

- **MsgSendFailure** (int err)
  *Construct an instance of a **MsgSendFailure** exception.*

- **MsgSendFailure** (int err)
  *Construct an instance of a **MsgSendFailure** exception.*

Detailed Description

Message could not be sent.

Thrown when an error occurs as the result of a message send.

Examples:

Cli/Cli.cpp.

Definition at line 139 of file ipc.svn-base.

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

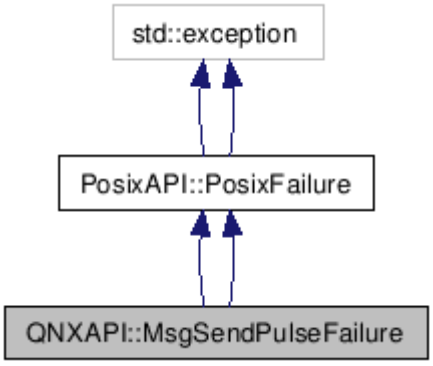
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::MsgSendPulseFailure Class Reference

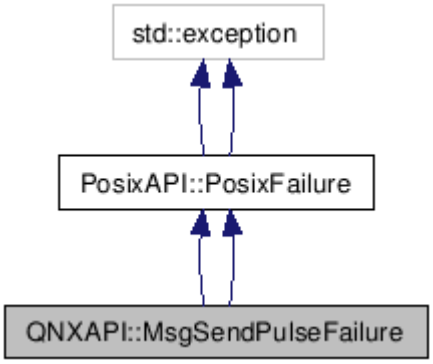
Pulse could not be sent.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::MsgSendPulseFailure:



Collaboration diagram for QNXAPI::MsgSendPulseFailure:



Public Member Functions

- **MsgSendPulseFailure** (int err)
  *Construct an instance of a **MsgSendPulseFailure** exception.*

- **MsgSendPulseFailure** (int err)
  *Construct an instance of a **MsgSendPulseFailure** exception.*

Detailed Description

Pulse could not be sent.

Thrown when an error occurs as the result of a pulse send.

Definition at line 155 of file ipc.svn-base.

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

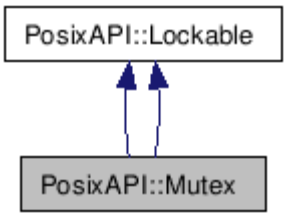
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

PosixAPI::Mutex Class Reference

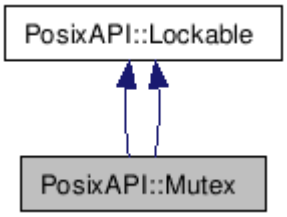
Mutex class.

Inherits **PosixAPI::Lockable**, and **PosixAPI::Lockable**.

Inheritance diagram for PosixAPI::Mutex:



Collaboration diagram for PosixAPI::Mutex:



Public Member Functions

- **Mutex** (void)
  *Construct an instance of a **Mutex** class.*

- **Mutex** (**MutexAttr** &attrib)
  *Construct an instance of a **Mutex** class.*

- virtual **~Mutex** (void)
  *Destroy an instance of a **Mutex** class.*

- virtual void **Lock** (**Lockable::Mode** exclusive=Lockable::Exclusive) const
  Lock mutex.

- virtual void **TryLock** (**Lockable::Mode** exclusive=Lockable::Exclusive) const
  Try to lock mutex.

- virtual void **Unlock** (void) const
  Unlock mutex.

- void **TimedLock** (uint64_t to)

Lock mutex with timeout.

- pthread_mutex_t & **Impl** (void) const throw ()
  Obtain underlying 'C' representation of mutex.

- **Mutex** (void)
  *Construct an instance of a **Mutex** class.*

- **Mutex** (**MutexAttr** &attrib)
  *Construct an instance of a **Mutex** class.*

- virtual **~Mutex** (void)
  *Destroy an instance of a **Mutex** class.*

- virtual void **Lock** (**Lockable::Mode** exclusive=Lockable::Exclusive) const
  Lock mutex.

- virtual void **TryLock** (**Lockable::Mode** exclusive=Lockable::Exclusive) const
  Try to lock mutex.

- virtual void **Unlock** (void) const
  Unlock mutex.

- void **TimedLock** (uint64_t to)
  Lock mutex with timeout.

- pthread_mutex_t & **Impl** (void) const throw ()
  Obtain underlying 'C' representation of mutex.

Detailed Description

Mutex class.

Implements mutex synchronization

Author:

Rennie Allen `rennieallen@gmail.com`

Examples:

Serv/Serv.h.

Definition at line 312 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::Mutex::Mutex (MutexAttr &* attrib*)

Construct an instance of a **Mutex** class.

Parameters:

attrib **Mutex** attributes

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

PosixAPI::Mutex::Mutex (MutexAttr &* attrib*)

Construct an instance of a **Mutex** class.

Parameters:

attrib **Mutex** attributes

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Member Function Documentation

virtual void PosixAPI::Mutex::Lock (Lockable::Mode* exclusive = *Lockable::Exclusive) const [virtual]

Lock mutex.

Parameters:

exclusive Exclusivity with which to lock

Implements **PosixAPI::Lockable** (*p.98*).

Referenced by UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::Lock().virtual void PosixAPI::Mutex::TryLock (Lockable::Mode* **exclusive =** ***Lockable::Exclusive) const [virtual]

Try to lock mutex.

Parameters:

exclusive Exclusivity with which to lock

Implements *PosixAPI::Lockable* ***(*****p.98*****).void PosixAPI::Mutex::TimedLock (uint64_t*** **to*****)***

Lock mutex with timeout.

Parameters:

to **Timeout**

pthread_mutex_t& PosixAPI::Mutex::Impl (void) const throw () [inline]

Obtain underlying 'C' representation of mutex.

Returns:

Reference to the underlying 'C' pthread_mutex_t representation

Definition at line 364 of file synchron.svn-base.virtual void PosixAPI::Mutex::Lock (Lockable::Mode* **exclusive =** ***Lockable::Exclusive) const [virtual]

Lock mutex.

Parameters:

exclusive Exclusivity with which to lock

Implements *PosixAPI::Lockable* ***(*****p.98*****).virtual void PosixAPI::Mutex::TryLock (Lockable::Mode*** **exclusive = *Lockable::Exclusive) const [virtual]***

Try to lock mutex.

Parameters:

exclusive Exclusivity with which to lock

Implements** PosixAPI::Lockable **(**p.98**).void PosixAPI::Mutex::TimedLock (uint64_t to)

Lock mutex with timeout.

Parameters:

to **Timeout**

pthread_mutex_t& PosixAPI::Mutex::Impl (void) const throw () [inline]

Obtain underlying 'C' representation of mutex.

Returns:

Reference to the underlying 'C' pthread_mutex_t representation

Definition at line 364 of file synchron.

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

- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

PosixAPI::MutexAttr Class Reference

Posix **Mutex** attributes.

Public Member Functions

- **MutexAttr** ()
- **MutexAttr** (**MutexAttrProtocol** protocol, bool recursive=false)
- void **Protocol** (**MutexAttrProtocol** protocol)
- void **Recursive** (bool recurse)
- **MutexAttrProtocol Protocol** (void) const
- bool **Recursive** (void) const
- pthread_mutexattr_t * **operator &** (void)
- **~MutexAttr** ()
- **MutexAttr** ()
- **MutexAttr** (**MutexAttrProtocol** protocol, bool recursive=false)
- void **Protocol** (**MutexAttrProtocol** protocol)
- void **Recursive** (bool recurse)
- **MutexAttrProtocol Protocol** (void) const
- bool **Recursive** (void) const
- pthread_mutexattr_t * **operator &** (void)
- **~MutexAttr** ()

Detailed Description

Posix **Mutex** attributes.

Definition at line 243 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::MutexAttr::MutexAttr ()

Construct a mutex attribute with defaults.

PosixAPI::MutexAttr::MutexAttr (MutexAttrProtocol* **protocol**, *bool* **recursive** = *false)

Construct a mutex attribute, supplying protocol and (optionally recursive) settings.

Parameters:

protocol mutex priority protocol.
recursive boolean indicating whether mutex should be able to be used recursively (optional, default: false)

PosixAPI::MutexAttr::~MutexAttr ()

Destroy the mutex attribute.

PosixAPI::MutexAttr::MutexAttr ()

Construct a mutex attribute with defaults.

PosixAPI::MutexAttr::MutexAttr (MutexAttrProtocol* **protocol**, *bool* **recursive** = *false)

Construct a mutex attribute, supplying protocol and (optionally recursive) settings.

Parameters:

protocol mutex priority protocol.
recursive boolean indicating whether mutex should be able to be used recursively (optional, default: false)

PosixAPI::MutexAttr::~MutexAttr ()

Destroy the mutex attribute.

Member Function Documentation

void PosixAPI::MutexAttr::Protocol (MutexAttrProtocol* **protocol**)

Set attributes priority protocol member.

Parameters:

protocol mutex priority protocol.

void PosixAPI::MutexAttr::Recursive (bool* **recurse**)

Set attributes recursion enable member.

Parameters:

recurse boolean; true - enable recursive locks, false - disable.

MutexAttrProtocol PosixAPI::MutexAttr::Protocol (void) const

Retrieve mutex attributes protocol setting.

Returns:
current protocol setting.

bool PosixAPI::MutexAttr::Recursive (void) const

Retrieve mutex attributes recursion setting.

Returns:
current recursion setting (true - recursion allowed; false - disallowed).

pthread_mutexattr_t* PosixAPI::MutexAttr::operator & (void)

Retrieve a pointer to the encapsulated pthread_mutexattr_t structure.

Returns:
pointer to pthread_mutexattr_t structure.

void PosixAPI::MutexAttr::Protocol (MutexAttrProtocol* **protocol)***

Set attributes priority protocol member.

Parameters:
protocol mutex priority protocol.

void PosixAPI::MutexAttr::Recursive (bool* **recurse)***

Set attributes recursion enable member.

Parameters:
recurse boolean; true - enable recursive locks, false - disable.

MutexAttrProtocol PosixAPI::MutexAttr::Protocol (void) const

Retrieve mutex attributes protocol setting.

Returns:
current protocol setting.

bool PosixAPI::MutexAttr::Recursive (void) const

Retrieve mutex attributes recursion setting.

Returns:
current recursion setting (true - recursion allowed; false - disallowed).

pthread_mutexattr_t* PosixAPI::MutexAttr::operator & (void)

Retrieve a pointer to the encapsulated pthread_mutexattr_t structure.

Returns:
pointer to pthread_mutexattr_t structure.

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

- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

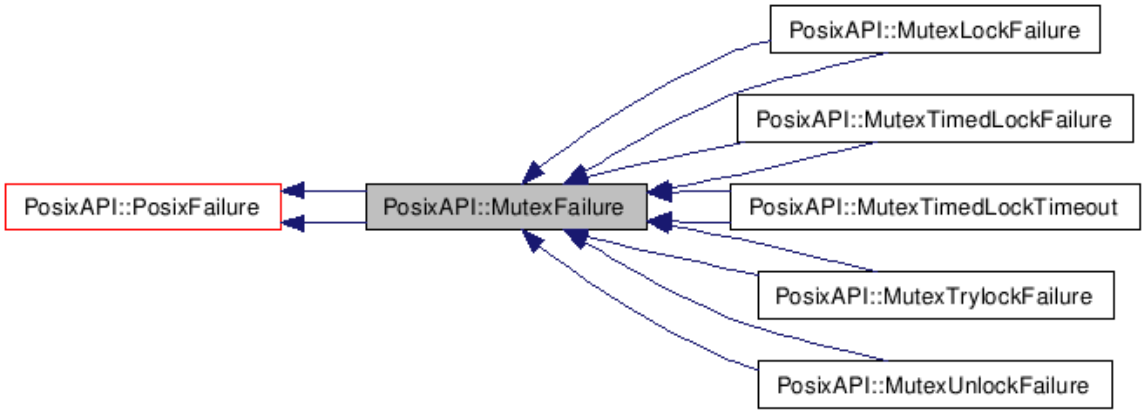
PosixAPI::MutexFailure Class Reference

MutexFailure base class.

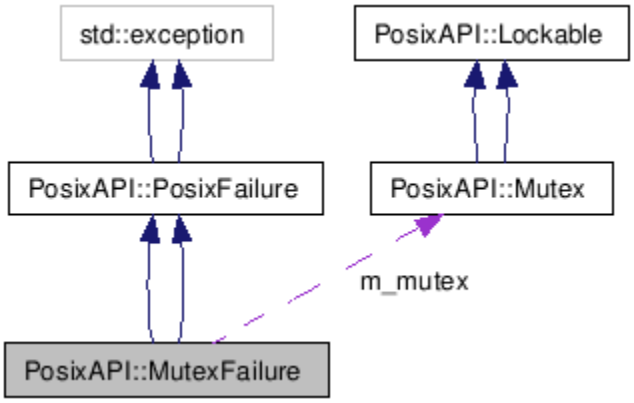
Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inherited by **PosixAPI::MutexLockFailure**, **PosixAPI::MutexLockFailure**, **PosixAPI::MutexTimedLockFailure**, **PosixAPI::MutexTimedLockFailure**, **PosixAPI::MutexTimedLockTimeout**, **PosixAPI::MutexTimedLockTimeout**, **PosixAPI::MutexTrylockFailure**, **PosixAPI::MutexTrylockFailure**, **PosixAPI::MutexUnlockFailure**, and **PosixAPI::MutexUnlockFailure**.

Inheritance diagram for PosixAPI::MutexFailure:



Collaboration diagram for PosixAPI::MutexFailure:



Public Member Functions

- **MutexFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexFailure** class.*

- **MutexFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexFailure** class.*

Detailed Description

MutexFailure base class.

Author:

Rennie Allen `rennieallen@gmail.com`

Definition at line 37 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::MutexFailure::MutexFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexFailure** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 46 of file synchron.svn-base.PosixAPI::MutexFailure::MutexFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexFailure** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 46 of file synchron.

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

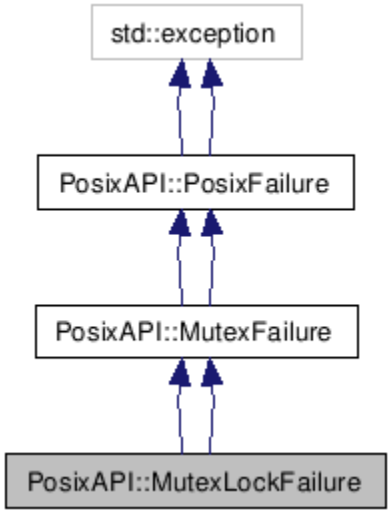
- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

PosixAPI::MutexLockFailure Class Reference

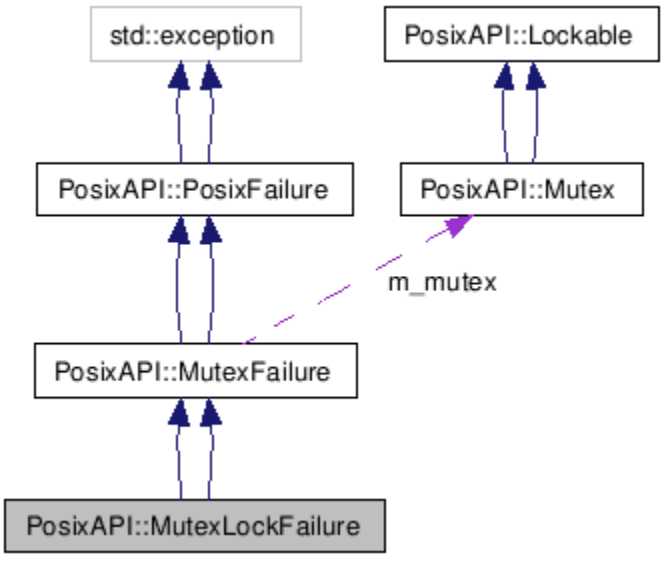
MutexLockFailure class.

Inherits **PosixAPI::MutexFailure**, and **PosixAPI::MutexFailure**.

Inheritance diagram for PosixAPI::MutexLockFailure:



Collaboration diagram for PosixAPI::MutexLockFailure:



Public Member Functions

- **MutexLockFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexLockFailure** class.*

- **MutexLockFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexLockFailure** class.*

Detailed Description

MutexLockFailure class.

Author:

Rennie Allen `rennieallen@gmail.com`

Definition at line 59 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::MutexLockFailure::MutexLockFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexLockFailure** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 68 of file synchron.svn-base.PosixAPI::MutexLockFailure::MutexLockFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexLockFailure** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 68 of file synchron.

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

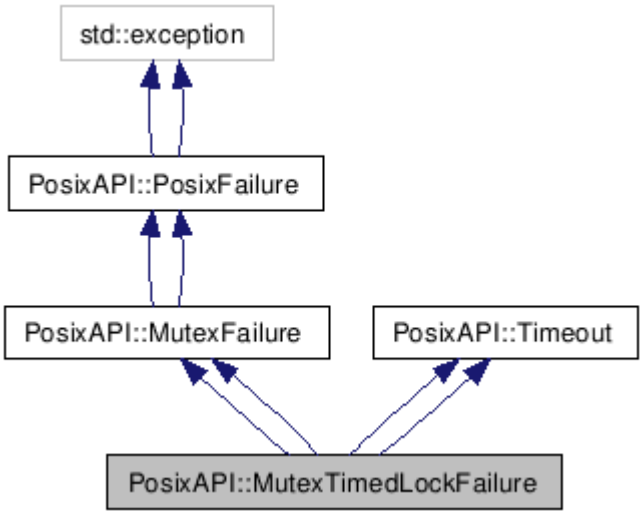
- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

PosixAPI::MutexTimedLockFailure Class Reference

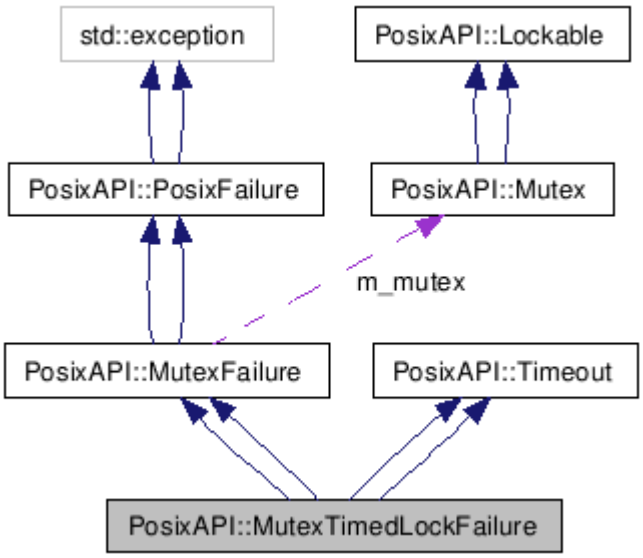
MutexTimedLockFailure class.

Inherits **PosixAPI::MutexFailure**, **PosixAPI::Timeout**, **PosixAPI::MutexFailure**, and **PosixAPI::Timeout**.

Inheritance diagram for PosixAPI::MutexTimedLockFailure:



Collaboration diagram for PosixAPI::MutexTimedLockFailure:



Public Member Functions

- **MutexTimedLockFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexTimedLockFailure** class.*

- **MutexTimedLockFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexTimedLockFailure** class.*

Detailed Description

MutexTimedLockFailure class.

Author:
Rennie Allen `rennieallen@gmail.com`

Definition at line 116 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::MutexTimedLockFailure::MutexTimedLockFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexTimedLockFailure** class.

Parameters:
mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 125 of file synchron.svn-base.PosixAPI::MutexTimedLockFailure::MutexTimedLockFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexTimedLockFailure** class.

Parameters:
mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 125 of file synchron.

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

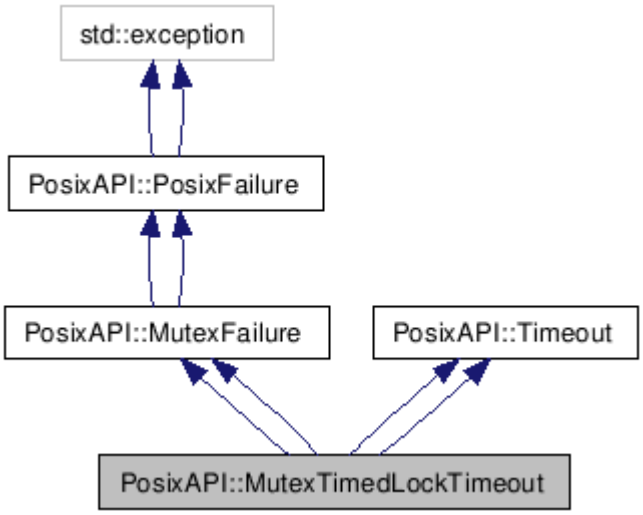
- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

PosixAPI::MutexTimedLockTimeout Class Reference

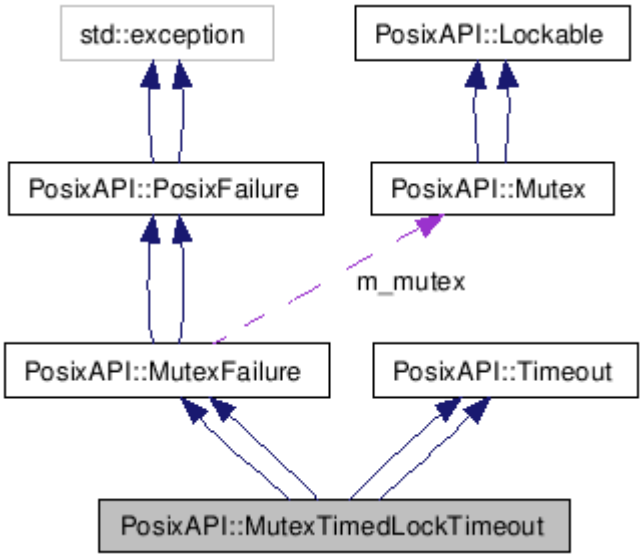
MutexTimedLockTimeout class.

Inherits **PosixAPI::MutexFailure**, **PosixAPI::Timeout**, **PosixAPI::MutexFailure**, and **PosixAPI::Timeout**.

Inheritance diagram for PosixAPI::MutexTimedLockTimeout:



Collaboration diagram for PosixAPI::MutexTimedLockTimeout:



Public Member Functions

- **MutexTimedLockTimeout** (const **Mutex** &mutex, int err=EOK)
  *Construct an instance of a **MutexTimedLockTimeout** class.*

- **MutexTimedLockTimeout** (const **Mutex** &mutex, int err=EOK)
  *Construct an instance of a **MutexTimedLockTimeout** class.*

Detailed Description

MutexTimedLockTimeout class.

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 135 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::MutexTimedLockTimeout::MutexTimedLockTimeout (const Mutex &* mutex, *int* err = *EOK) [inline]

Construct an instance of a **MutexTimedLockTimeout** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 144 of file synchron.svn-base.PosixAPI::MutexTimedLockTimeout::MutexTimedLockTimeout (const Mutex & mutex, int err = EOK) [inline]

Construct an instance of a **MutexTimedLockTimeout** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 144 of file synchron.

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

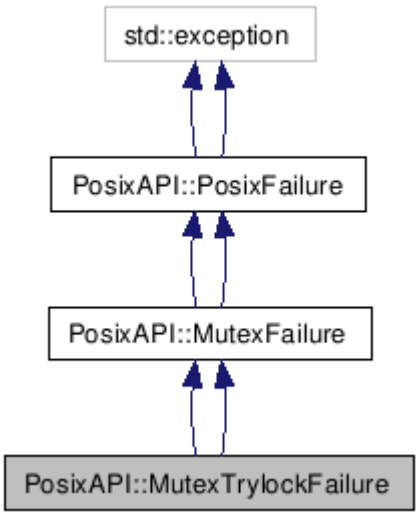
- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

PosixAPI::MutexTrylockFailure Class Reference

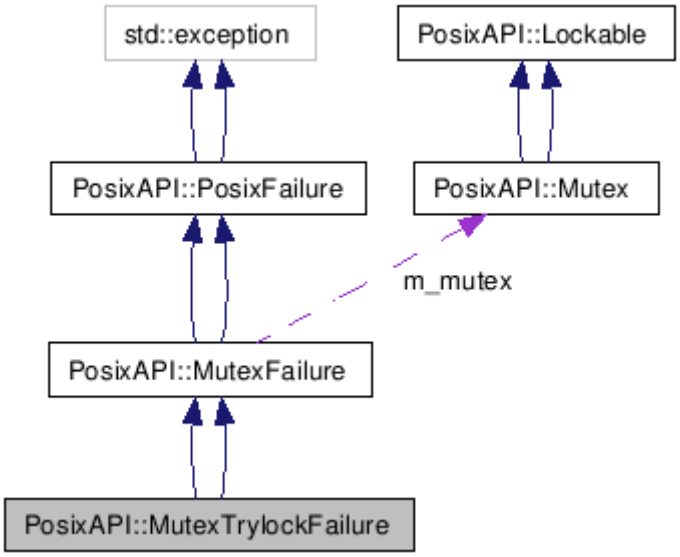
MutexTrylockFailure class.

Inherits **PosixAPI::MutexFailure**, and **PosixAPI::MutexFailure**.

Inheritance diagram for PosixAPI::MutexTrylockFailure:



Collaboration diagram for PosixAPI::MutexTrylockFailure:



Public Member Functions

- **MutexTrylockFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexTrylockFailure** class.*

- **MutexTrylockFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexTrylockFailure** class.*

Detailed Description

MutexTrylockFailure class.

Author:

Rennie Allen `rennieallen@gmail.com`

Definition at line 97 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::MutexTrylockFailure::MutexTrylockFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexTrylockFailure** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 106 of file synchron.svn-base.PosixAPI::MutexTrylockFailure::MutexTrylockFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexTrylockFailure** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 106 of file synchron.

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

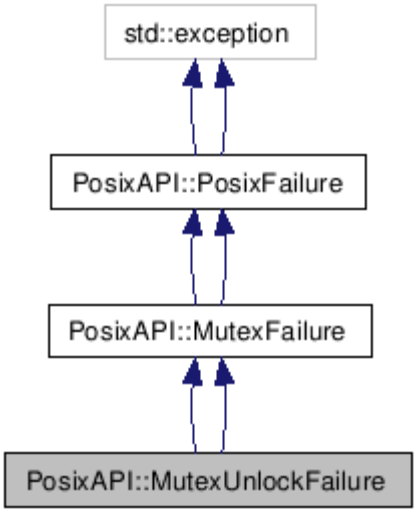
- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

PosixAPI::MutexUnlockFailure Class Reference

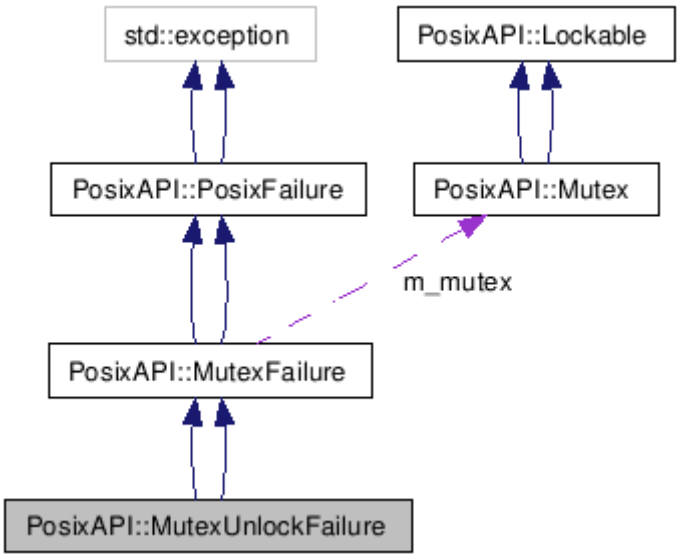
MutexUnlockFailure class.

Inherits **PosixAPI::MutexFailure**, and **PosixAPI::MutexFailure**.

Inheritance diagram for PosixAPI::MutexUnlockFailure:



Collaboration diagram for PosixAPI::MutexUnlockFailure:



Public Member Functions

- **MutexUnlockFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexUnlockFailure** class.*

- **MutexUnlockFailure** (const **Mutex** &mutex, int err)
  *Construct an instance of a **MutexUnlockFailure** class.*

Detailed Description

MutexUnlockFailure class.

Author:

Rennie Allen rennieallen@gmail.com

Definition at line 78 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::MutexUnlockFailure::MutexUnlockFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexUnlockFailure** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 87 of file synchron.svn-base.PosixAPI::MutexUnlockFailure::MutexUnlockFailure (const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **MutexUnlockFailure** class.

Parameters:

mutex **Mutex** that experienced exception
err Posix error code of failure

Definition at line 87 of file synchron.

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

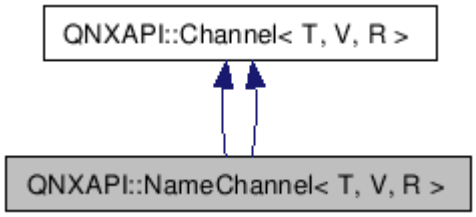
- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

QNXAPI::NameChannel< T, V, R > Class Template Reference

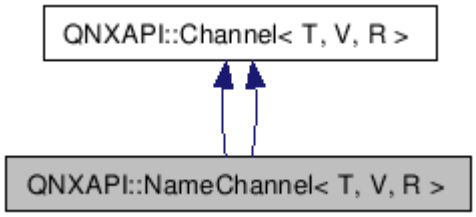
NameChannel class.

Inherits **QNXAPI::Channel< T, V, R >< T, V, R >**, and **QNXAPI::Channel< T, V, R >< T, V, R >**.

Inheritance diagram for QNXAPI::NameChannel< T, V, R >:



Collaboration diagram for QNXAPI::NameChannel< T, V, R >:



Public Member Functions

- **NameChannel** (**MessageVector**< T, V, R > &msg, sigc::slot< int, **Reception** & > cbMsg, sigc::slot< void, **Reception** & > cbPulse)
  *Construct an instance of a **NameChannel** class.*

- **NameChannel** (const char *name, **MessageVector**< T, V, R > &msg, sigc::slot< int, **Reception** & > cbMsg, sigc::slot< void, **Reception** & > cbPulse, uint32_t flags=0, int wakeupPri=10)
  *Construct an instance of a **NameChannel** class, and attach it immediately.*

- virtual **~NameChannel** (void)
  *Destroy an instance of a **NameChannel** class.*

- void **Attach** (const char *name, uint32_t flags=0, int wakeupPri=10)
  *Attach an instance of a **NameChannel** class, and attach it immediately.*

- **NameChannel** (**MessageVector**< T, V, R > &msg, sigc::slot< int, **Reception** & > cbMsg, sigc::slot< void, **Reception** & > cbPulse)
  *Construct an instance of a **NameChannel** class.*

- **NameChannel** (const char *name, **MessageVector**< T, V, R > &msg, sigc::slot< int, **Reception** & > cbMsg, sigc::slot< void, **Reception** & > cbPulse, uint32_t flags=0, int wakeupPri=10)

*Construct an instance of a **NameChannel** class, and attach it immediately.*

- virtual **~NameChannel** (void)
  *Destroy an instance of a **NameChannel** class.*

- void **Attach** (const char *name, uint32_t flags=0, int wakeupPri=10)
  *Attach an instance of a **NameChannel** class, and attach it immediately.*

Protected Attributes

- name_attach_t * **m_attach**
  Underlying 'C' name_attach representation.

- name_attach_t * **m_attach**
  Underlying 'C' name_attach representation.

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::NameChannel< T, V, R >

NameChannel class.

A **NameChannel** extends a channel, by allowing it to have a name. A named channel can be located outside of the process, while a plain channel is only useful within a process.

Examples:

Serv/Serv.h.

Definition at line 1076 of file ipc.svn-base.

Constructor & Destructor Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::NameChannel< T, V, R >::NameChannel (MessageVector< T, V, R > & msg, sigc::slot< int, Reception & > cbMsg, sigc::slot< void, Reception & > cbPulse) [inline]

Construct an instance of a **NameChannel** class.

Parameters:

msg message vector into which messages will be received.
cbMsg message handler callback

See also:

QNXAPI::NameChannel::m_cbMsg.

Parameters:

cbPulse pulse handler callback

See also:

QNXAPI::NameChannel::m_cbPulse.

Definition at line 1086 of file ipc.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::NameChannel< T, V, R >::NameChannel (const char * name, MessageVector< T, V, R > & msg, sigc::slot< int, Reception & > cbMsg, sigc::slot< void, Reception & > cbPulse, uint32_t flags = 0, int wakeupPri = 10) [inline]

Construct an instance of a **NameChannel** class, and attach it immediately.

Parameters:

name prefix to be registered in pathname space.
msg message vector into which messages will be received.
cbMsg message handler callback

See also:

QNXAPI::NameChannel::m_cbMsg.

Parameters:

cbPulse pulse handler callback

See also:

QNXAPI::NameChannel::m_cbPulse.

Parameters:

flags channel flags

See also:

ChannelCreate.

Parameters:

wakeupPri priority at which Wakeup method will drive the channel.

Definition at line 1101 of file ipc.svn-base.

References QNXAPI::NameChannel< T, V, R >::Attach().

Here is the call graph for this function:

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::NameChannel< T, V, R >::NameChannel (MessageVector< T, V, R > & msg, sigc::slot< int, Reception & > cbMsg, sigc::slot< void, Reception & > cbPulse) [inline]

Construct an instance of a **NameChannel** class.

Parameters:

msg message vector into which messages will be received.
cbMsg message handler callback

See also:

QNXAPI::NameChannel::m_cbMsg.

Parameters:

cbPulse pulse handler callback

See also:

QNXAPI::NameChannel::m_cbPulse.

Definition at line 1086 of file ipc.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::NameChannel< T, V, R >::NameChannel (const char * name, MessageVector< T, V, R > & msg, sigc::slot< int, Reception & > cbMsg, sigc::slot< void, Reception & > cbPulse, uint32_t flags = 0, int wakeupPri = 10) [inline]

Construct an instance of a **NameChannel** class, and attach it immediately.

Parameters:

name prefix to be registered in pathname space.
msg message vector into which messages will be received.
cbMsg message handler callback

See also:

QNXAPI::NameChannel::m_cbMsg.

Parameters:

cbPulse pulse handler callback

See also:

QNXAPI::NameChannel::m_cbPulse.

Parameters:

flags channel flags

See also:

ChannelCreate.

Parameters:

wakeupPri priority at which Wakeup method will drive the channel.

Definition at line 1101 of file ipc.

References QNXAPI::NameChannel< T, V, R >::Attach().

Here is the call graph for this function:

Member Function Documentation

```
template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void
QNXAPI::NameChannel< T, V, R >::Attach (const char * name, uint32_t flags = 0, int wakeupPri = 10)
[inline]
```

Attach an instance of a **NameChannel** class, and attach it immediately.

Parameters:

name prefix to be registered in pathname space.
flags channel flags

See also:

ChannelCreate.

Parameters:

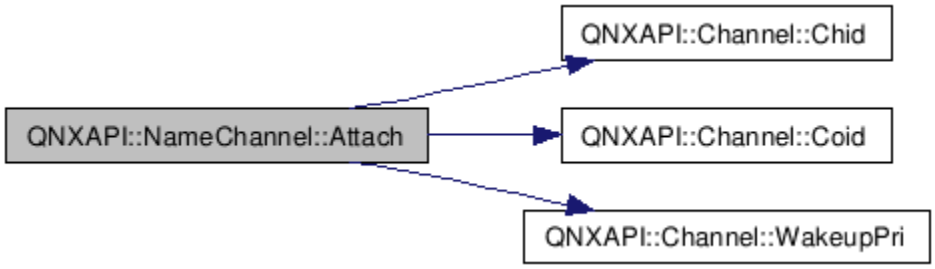
wakeupPri priority at which Wakeup method will drive the channel.

Definition at line 1124 of file ipc.svn-base.

References QNXAPI::Channel< T, V, R >::Chid(), QNXAPI::Channel< T, V, R >::Coid(), QNXAPI::NameChannel< T, V, R >::m_attach, and QNXAPI::Channel< T, V, R >::WakeupPri().

Referenced by QNXAPI::NameChannel< T, V, R >::NameChannel().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::NameChannel< T, V, R >::Attach (const char * name, uint32_t flags = 0, int wakeupPri = 10) [inline]

Attach an instance of a **NameChannel** class, and attach it immediately.

Parameters:

name prefix to be registered in pathname space.
flags channel flags

See also:

ChannelCreate.

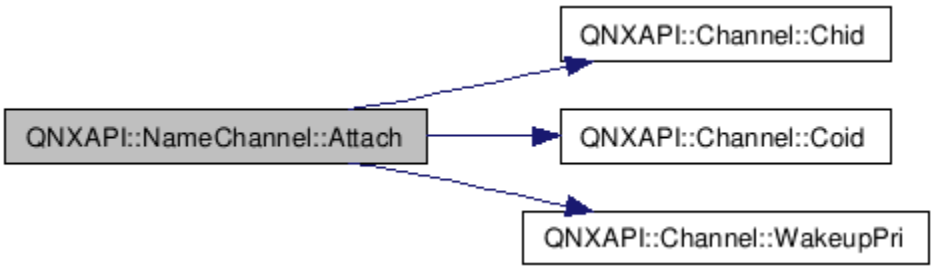
Parameters:

wakeupPri priority at which Wakeup method will drive the channel.

Definition at line 1124 of file ipc.

References QNXAPI::Channel< T, V, R >::Chid(), QNXAPI::Channel< T, V, R >::Coid(), QNXAPI::NameChannel< T, V, R >::m_attach, and QNXAPI::Channel< T, V, R >::WakeupPri().

Here is the call graph for this function:



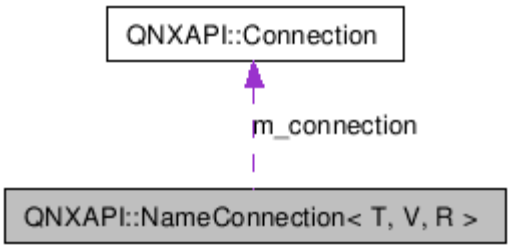
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::NameConnection< T, V, R > Class Template Reference

NameConnection class.

Collaboration diagram for QNXAPI::NameConnection< T, V, R >:



Public Member Functions

- **NameConnection** (void)
  *Construct an instance of a **NameConnection** class.*

- **NameConnection** (const char *name, uint32_t flags=0)
  *Construct an instance of a **NameConnection** class, and connect to the named channel immediately.*

- virtual **~NameConnection** (void)
  *Destroy an instance of a **NameConnection** class.*

- void **Connect** (const char *name, uint32_t flags=0)
  *Connect an instance of a **NameConnection** class to the named channel.*

- void **Send** (**MessageVector**< T, V, R > &svect)
  *Send a message on an instance of a **NameConnection** class to the named channel.*

- void **Send** (**MessageVector**< T, V, R > &svect, **ReplyVector**< R, V > &rvect)
  *Send a message on an instance of a **NameConnection** class to the named channel, and receive the reply.*

- void **SendPulse** (int pri, int code, int val)
  *Send a pulse on an instance of a **NameConnection** class to the named channel.*

- **NameConnection** (void)
  *Construct an instance of a **NameConnection** class.*

- **NameConnection** (const char *name, uint32_t flags=0)
  *Construct an instance of a **NameConnection** class, and connect to the named channel immediately.*

- virtual **~NameConnection** (void)
  *Destroy an instance of a **NameConnection** class.*

- void **Connect** (const char *name, uint32_t flags=0)
  *Connect an instance of a **NameConnection** class to the named channel.*

- void **Send** (**MessageVector**< T, V, R > &svect)
  *Send a message on an instance of a **NameConnection** class to the named channel.*

- void **Send** (**MessageVector**< T, V, R > &svect, **ReplyVector**< R, V > &rvect)
  *Send a message on an instance of a **NameConnection** class to the named channel, and receive the reply.*

- void **SendPulse** (int pri, int code, int val)
  *Send a pulse on an instance of a **NameConnection** class to the named channel.*

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::NameConnection< T, V, R >

NameConnection class.

A **NameConnection** implements a connection to a name channel (typically, although not necessarily, from a different process than the process that created the **NameChannel**.

Examples:

Cli/Cli.cpp.

Definition at line 1150 of file ipc.svn-base.

Constructor & Destructor Documentation

```
template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem>
QNXAPI::NameConnection< T, V, R >::NameConnection (const char * name, uint32_t flags = 0) [inline]
```

Construct an instance of a **NameConnection** class, and connect to the named channel immediately.

Parameters:

name name to be attached to (opened).
flags connection flags

See also:

ConnectAttach.

Definition at line 1167 of file ipc.svn-base.

References QNXAPI::NameConnection< T, V, R >::Connect().

Here is the call graph for this function:


```
template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem>
QNXAPI::NameConnection< T, V, R >::NameConnection (const char * name, uint32_t flags = 0) [inline]
```

Construct an instance of a **NameConnection** class, and connect to the named channel immediately.

Parameters:

name name to be attached to (opened).
flags connection flags

See also:

ConnectAttach.

Definition at line 1167 of file ipc.

References QNXAPI::NameConnection< T, V, R >::Connect().

Here is the call graph for this function:

Member Function Documentation

```
template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void
QNXAPI::NameConnection< T, V, R >::Connect (const char * name, uint32_t flags = 0) [inline]
```

Connect an instance of a **NameConnection** class to the named channel.

Parameters:

name name to be attached to (opened).
flags connection flags

See also:

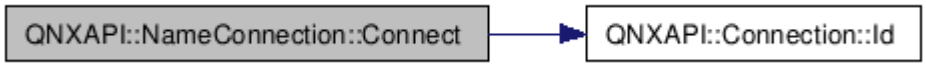
ConnectAttach.

Definition at line 1186 of file ipc.svn-base.

References QNXAPI::Connection::Id().

Referenced by QNXAPI::NameConnection< T, V, R >::NameConnection().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::NameConnection< T, V, R >::Send (MessageVector< T, V, R > & svect) [inline]

Send a message on an instance of a **NameConnection** class to the named channel.

Parameters:

svect message vector containing message to send.

Definition at line 1200 of file ipc.svn-base.

References QNXAPI::MessageVector< T, V, R >::Send().

Here is the call graph for this function:

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::NameConnection< T, V, R >::Send (MessageVector< T, V, R > & svect, ReplyVector< R, V > & rvect) [inline]

Send a message on an instance of a **NameConnection** class to the named channel, and receive the reply.

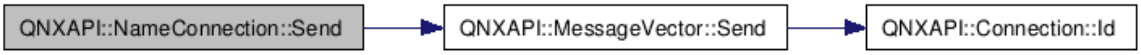
Parameters:

svect message vector containing message to send.
rvect specialized message vector to hold the reply.

Definition at line 1212 of file ipc.svn-base.

References QNXAPI::MessageVector< T, V, R >::Send().

Here is the call graph for this function:




```
template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void
QNXAPI::NameConnection< T, V, R >::SendPulse (int pri, int code, int val) [inline]
```

Send a pulse on an instance of a **NameConnection** class to the named channel.

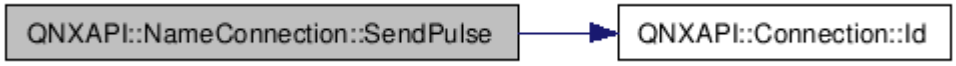
Parameters:

pri priority of pulse.
code pulse code.
val pulse value.

Definition at line 1225 of file ipc.svn-base.

References QNXAPI::Connection::Id().

Here is the call graph for this function:




```
template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void
QNXAPI::NameConnection< T, V, R >::Connect (const char * name, uint32_t flags = 0) [inline]
```

Connect an instance of a **NameConnection** class to the named channel.

Parameters:

name name to be attached to (opened).
flags connection flags

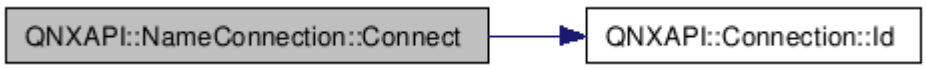
See also:

ConnectAttach.

Definition at line 1186 of file ipc.

References QNXAPI::Connection::Id().

Here is the call graph for this function:




```
template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void
QNXAPI::NameConnection< T, V, R >::Send (MessageVector< T, V, R > & svect) [inline]
```

Send a message on an instance of a **NameConnection** class to the named channel.

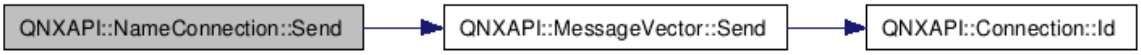
Parameters:

svect message vector containing message to send.

Definition at line 1200 of file ipc.

References QNXAPI::MessageVector< T, V, R >::Send().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::NameConnection< T, V, R >::Send (MessageVector< T, V, R > & svect, ReplyVector< R, V > & rvect) [inline]

Send a message on an instance of a **NameConnection** class to the named channel, and receive the reply.

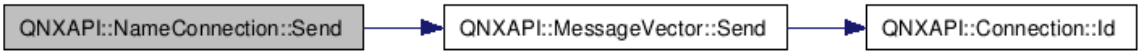
Parameters:

svect message vector containing message to send.
rvect specialized message vector to hold the reply.

Definition at line 1212 of file ipc.

References QNXAPI::MessageVector< T, V, R >::Send().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::NameConnection< T, V, R >::SendPulse (int pri, int code, int val) [inline]

Send a pulse on an instance of a **NameConnection** class to the named channel.

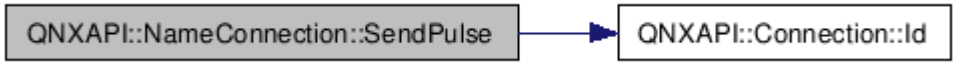
Parameters:

pri priority of pulse.
code pulse code.
val pulse value.

Definition at line 1225 of file ipc.

References QNXAPI::Connection::Id().

Here is the call graph for this function:



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

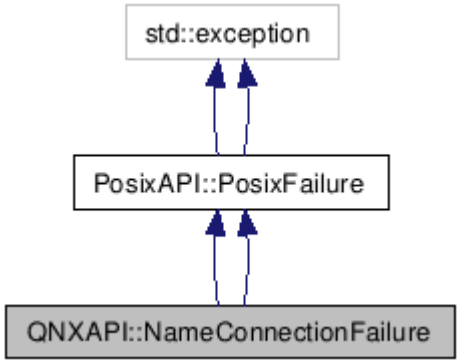
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::NameConnectionFailure Class Reference

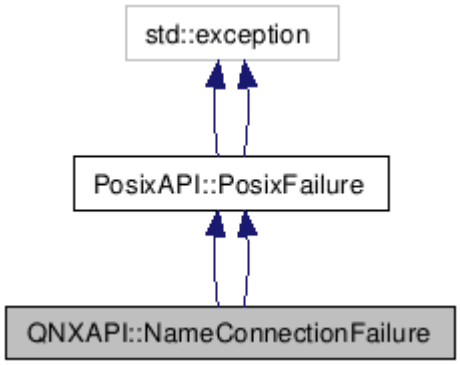
A **NameConnection** failure occured.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::NameConnectionFailure:



Collaboration diagram for QNXAPI::NameConnectionFailure:



Public Member Functions

- **NameConnectionFailure** (int err)
  *Construct an instance of a **NameConnectionFailure** exception.*

- **NameConnectionFailure** (int err)
  *Construct an instance of a **NameConnectionFailure** exception.*

Detailed Description

A **NameConnection** failure occured.

Thrown when an error occurs as the result of an operation on a name connection.

Examples:

Cli/Cli.cpp.

Definition at line 298 of file ipc.svn-base.

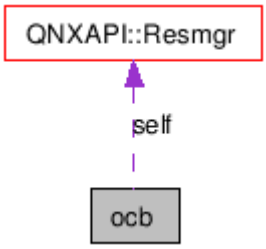
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

ocb Struct Reference

'C' representation of a resource manager ocb (Open Control Block)

Collaboration diagram for ocb:



Public Attributes

- iofunc_ocb_t **hdr**
  storage for header

- **QNXAPI::Resmgr** * **self**
  storage for pointer to self

- **QNXAPI::Resmgr** * **self**

storage for pointer to self

Detailed Description

'C' representation of a resource manager ocb (Open Control Block)

Extension of iofunc_ocb struct to allow hooking of resmgr class.

Definition at line 72 of file resmgr.svn-base.

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

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

QNXAPI::PoolContext< T, V, R > Class Template Reference

PoolContext class.

Public Member Functions

- **PoolContext** (**ThreadPool**< T, V, R > &pool, typename **ThreadPool**< T, V, R >::Context &context)
  *Construct an instance of a **PoolContext** class.*

- **~PoolContext** ()
  *Destroy an instance of a **PoolContext** class.*

- **PoolContext** (**ThreadPool**< T, V, R > &pool, typename **ThreadPool**< T, V, R >::Context &context)
  *Construct an instance of a **PoolContext** class.*

- **~PoolContext** ()
  *Destroy an instance of a **PoolContext** class.*

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::PoolContext< T, V, R >

PoolContext class.

Author:
rennieallen@gmail.com

Definition at line 505 of file threadpool.svn-base.

Constructor & Destructor Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::PoolContext< T, V, R >::PoolContext (ThreadPool< T, V, R > & pool, typename ThreadPool< T, V, R >::Context & context) [inline]

Construct an instance of a **PoolContext** class.

Parameters:
pool Reference to pool
context Reference to context

Definition at line 514 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::PoolContext< T, V, R >::PoolContext (ThreadPool< T, V, R > & pool, typename ThreadPool< T, V, R >::Context & context) [inline]

Construct an instance of a **PoolContext** class.

Parameters:
pool Reference to pool
context Reference to context

Definition at line 514 of file threadpool.

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

- include/qfc/.svn/text-base/threadpool.svn-base
- include/qfc/threadpool

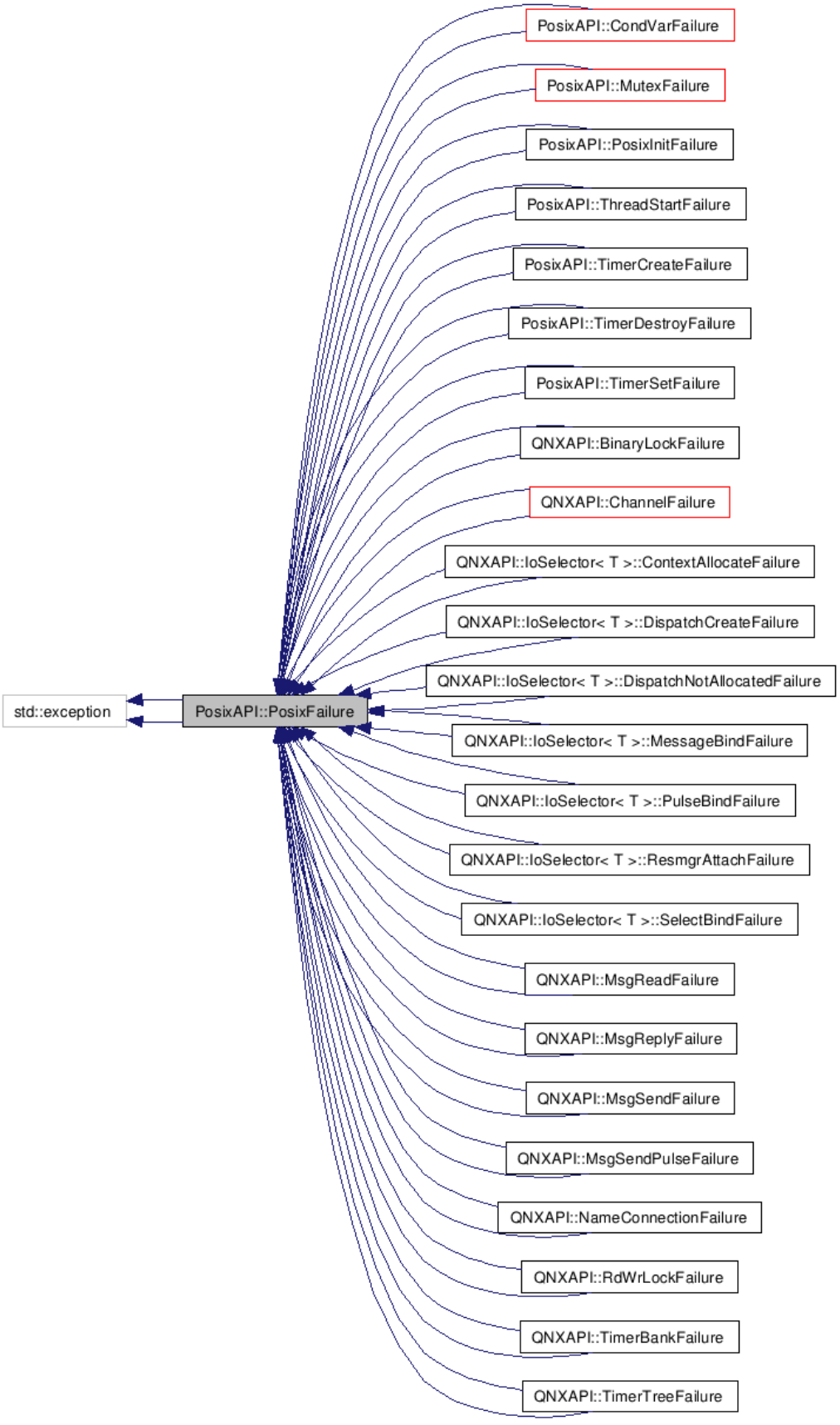
PosixAPI::PosixFailure Class Reference

PosixFailure base class.

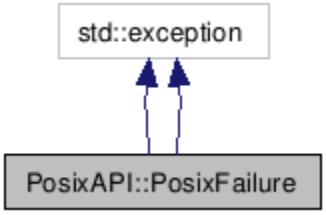
Inherits std::exception, and std::exception.

Inherited by **PosixAPI::CondVarFailure**, **PosixAPI::CondVarFailure**, **PosixAPI::MutexFailure**, **PosixAPI::MutexFailure**, **PosixAPI::PosixInitFailure**, **PosixAPI::PosixInitFailure**, **PosixAPI::ThreadStartFailure**, **PosixAPI::ThreadStartFailure**, **PosixAPI::TimerCreateFailure**, **PosixAPI::TimerCreateFailure**, **PosixAPI::TimerDestroyFailure**, **PosixAPI::TimerDestroyFailure**, **PosixAPI::TimerSetFailure**, **PosixAPI::TimerSetFailure**, **QNXAPI::BinaryLockFailure**, **QNXAPI::BinaryLockFailure**, **QNXAPI::ChannelFailure**, **QNXAPI::ChannelFailure**, **QNXAPI::IoSelector< T >::ContextAllocateFailure**, **QNXAPI::IoSelector< T >::ContextAllocateFailure**, **QNXAPI::IoSelector< T >::DispatchCreateFailure**, **QNXAPI::IoSelector< T >::DispatchCreateFailure**, **QNXAPI::IoSelector< T >::DispatchNotAllocatedFailure**, **QNXAPI::IoSelector< T >::DispatchNotAllocatedFailure**, **QNXAPI::IoSelector< T >::MessageBindFailure**, **QNXAPI::IoSelector< T >::MessageBindFailure**, **QNXAPI::IoSelector< T >::PulseBindFailure**, **QNXAPI::IoSelector< T >::PulseBindFailure**, **QNXAPI::IoSelector< T >::ResmgrAttachFailure**, **QNXAPI::IoSelector< T >::ResmgrAttachFailure**, **QNXAPI::IoSelector< T >::SelectBindFailure**, **QNXAPI::IoSelector< T >::SelectBindFailure**, **QNXAPI::MsgReadFailure**, **QNXAPI::MsgReadFailure**, **QNXAPI::MsgReplyFailure**, **QNXAPI::MsgReplyFailure**, **QNXAPI::MsgSendFailure**, **QNXAPI::MsgSendFailure**, **QNXAPI::MsgSendPulseFailure**, **QNXAPI::MsgSendPulseFailure**, **QNXAPI::NameConnectionFailure**, **QNXAPI::NameConnectionFailure**, **QNXAPI::RdWrLockFailure**, **QNXAPI::RdWrLockFailure**, **QNXAPI::TimerBankFailure**, **QNXAPI::TimerBankFailure**, **QNXAPI::TimerTreeFailure**, and **QNXAPI::TimerTreeFailure**.

Inheritance diagram for PosixAPI::PosixFailure:


std::exception
PosixAPI::PosixFailure
PosixAPI::CondVarFailure
PosixAPI::MutexFailure
PosixAPI::PosixInitFailure
PosixAPI::ThreadStartFailure
PosixAPI::TimerCreateFailure
PosixAPI::TimerDestroyFailure
PosixAPI::TimerSetFailure
QNXAPI::BinaryLockFailure
QNXAPI::ChannelFailure
QNXAPI::IoSelector< T >::ContextAllocateFailure
QNXAPI::IoSelector< T >::DispatchCreateFailure
QNXAPI::IoSelector< T >::DispatchNotAllocatedFailure
QNXAPI::IoSelector< T >::MessageBindFailure
QNXAPI::IoSelector< T >::PulseBindFailure
QNXAPI::IoSelector< T >::ResmgrAttachFailure
QNXAPI::IoSelector< T >::SelectBindFailure
QNXAPI::MsgReadFailure
QNXAPI::MsgReplyFailure
QNXAPI::MsgSendFailure
QNXAPI::MsgSendPulseFailure
QNXAPI::NameConnectionFailure
QNXAPI::RdWrLockFailure
QNXAPI::TimerBankFailure
QNXAPI::TimerTreeFailure

Collaboration diagram for PosixAPI::PosixFailure:



Public Member Functions

- **PosixFailure** (int err)
  *Construct a **PosixFailure** base class.*

- int **Err** (void)
  Get the posix error number.

- virtual const char * **ErrMsg** (void)
  Get the textual description of the posix error.

- virtual const char * **what** (void) throw ()
  *Get the textual description of the posix error, using the standard **what()** method.*

- **PosixFailure** (int err)
  *Construct a **PosixFailure** base class.*

- int **Err** (void)
  Get the posix error number.

- virtual const char * **ErrMsg** (void)
  Get the textual description of the posix error.

- virtual const char * **what** (void) throw ()
  *Get the textual description of the posix error, using the standard **what()** method.*

Detailed Description

PosixFailure base class.

Base for posix exceptions. Encapsulates error number, and method to decode error number.

Examples:

Cli/Cli.cpp.

Definition at line 42 of file except.svn-base.

Constructor & Destructor Documentation

PosixAPI::PosixFailure::PosixFailure (int* err*) [inline]

Construct a **PosixFailure** base class.

Parameters:

err standard error code.

Definition at line 50 of file except.svn-base.PosixAPI::PosixFailure::PosixFailure (int* err*) [inline]

Construct a **PosixFailure** base class.

Parameters:

err standard error code.

Definition at line 50 of file except.

Member Function Documentation

int PosixAPI::PosixFailure::Err (void) [inline]

Get the posix error number.

Returns:

err standard error code.

Definition at line 59 of file except.svn-base.virtual const char* PosixAPI::PosixFailure::ErrMsg (void) [inline, virtual]

Get the textual description of the posix error.

Returns:

textual representation of the error.

Examples:

Cli/Cli.cpp.

Definition at line 66 of file except.svn-base.

Referenced by what().virtual const char* PosixAPI::PosixFailure::what (void) throw () [inline, virtual]

Get the textual description of the posix error, using the standard **what()** method.

Returns:

textual representation of the error.

Definition at line 73 of file except.svn-base.

References ErrMsg().

Here is the call graph for this function:

int PosixAPI::PosixFailure::Err (void) [inline]

Get the posix error number.

Returns:

err standard error code.

Definition at line 59 of file except.virtual const char* PosixAPI::PosixFailure::ErrMsg (void) [inline, virtual]

Get the textual description of the posix error.

Returns:

textual representation of the error.

Definition at line 66 of file except.virtual const char* PosixAPI::PosixFailure::what (void) throw () [inline, virtual]

Get the textual description of the posix error, using the standard **what()** method.

Returns:

textual representation of the error.

Definition at line 73 of file except.

References ErrMsg().

Here is the call graph for this function:

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

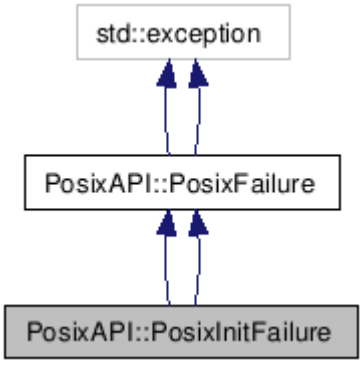
- include/qfc/.svn/text-base/except.svn-base
- include/qfc/except

PosixAPI::PosixInitFailure Class Reference

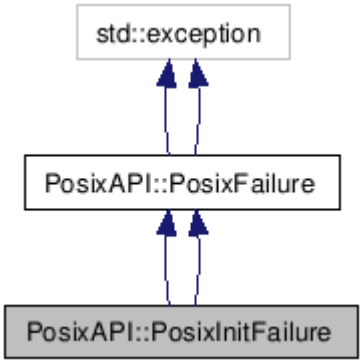
PosixInitFailure exception.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for PosixAPI::PosixInitFailure:



Collaboration diagram for PosixAPI::PosixInitFailure:



Public Member Functions

- **PosixInitFailure** (int err)
- **PosixInitFailure** (int err)

Detailed Description

PosixInitFailure exception.

Exception encountered while attempting to initialize a posix facility.

Definition at line 85 of file except.svn-base.

Constructor & Destructor Documentation

PosixAPI::PosixInitFailure::PosixInitFailure (int err) [inline]

Construct a **PosixInitFailure** base class.

Parameters:

err standard error code.

Definition at line 93 of file except.svn-base.PosixAPI::PosixInitFailure::PosixInitFailure (int err) [inline]

Construct a **PosixInitFailure** base class.

Parameters:

err standard error code.

Definition at line 93 of file except.

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

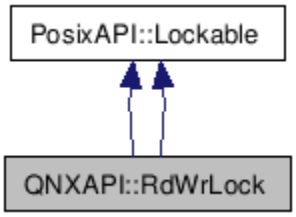
- include/qfc/.svn/text-base/except.svn-base
- include/qfc/except

QNXAPI::RdWrLock Class Reference

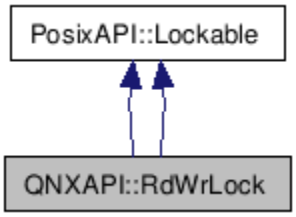
Read/Write lock.

Inherits **PosixAPI::Lockable**, and **PosixAPI::Lockable**.

Inheritance diagram for QNXAPI::RdWrLock:



Collaboration diagram for QNXAPI::RdWrLock:



Public Member Functions

- void **Lock** (**PosixAPI::Lockable::Mode** exclusive=PosixAPI::Lockable::Exclusive) const
- void **TryLock** (**PosixAPI::Lockable::Mode** mode=PosixAPI::Lockable::Exclusive) const
- void **Unlock** (void) const
- void **Lock** (**PosixAPI::Lockable::Mode** exclusive=PosixAPI::Lockable::Exclusive) const
- void **TryLock** (**PosixAPI::Lockable::Mode** mode=PosixAPI::Lockable::Exclusive) const
- void **Unlock** (void) const

Detailed Description

Read/Write lock.

Definition at line 183 of file lock.svn-base.

Member Function Documentation

void QNXAPI::RdWrLock::Lock (PosixAPI::Lockable::Mode* exclusive = *PosixAPI::Lockable::Exclusive) const [virtual]

Lock the read/write lock for write (if exclusive, or read if not exclusive). Since the default for a Lockable interface, is exclusive, calling this method with no arguments results in a call for a write lock.

Parameters:

exclusive true to attempt a write lock, false to obtain a read lock.

Implements** PosixAPI::Lockable* **(p.98)*.void QNXAPI::RdWrLock::TryLock (PosixAPI::Lockable::Mode* mode = *PosixAPI::Lockable::Exclusive) const [virtual]

Try to get an exclusive lock.

Implements** PosixAPI::Lockable* **(p.98)*.void QNXAPI::RdWrLock::Unlock (void) const [virtual]

Unlock the mutex.

Implements** PosixAPI::Lockable* **(p.98)*.void QNXAPI::RdWrLock::Lock (PosixAPI::Lockable::Mode* exclusive = *PosixAPI::Lockable::Exclusive) const [virtual]

Lock the read/write lock for write (if exclusive, or read if not exclusive). Since the default for a Lockable interface, is exclusive, calling this method with no arguments results in a call for a write lock.

Parameters:

exclusive true to attempt a write lock, false to obtain a read lock.

Implements** PosixAPI::Lockable* **(p.98)*.void QNXAPI::RdWrLock::TryLock (PosixAPI::Lockable::Mode* mode = *PosixAPI::Lockable::Exclusive) const [virtual]

Try to get an exclusive lock.

Implements** PosixAPI::Lockable* **(p.98)*.void QNXAPI::RdWrLock::Unlock (void) const [virtual]

Unlock the mutex.

Implements **PosixAPI::Lockable** (*p.98*).

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

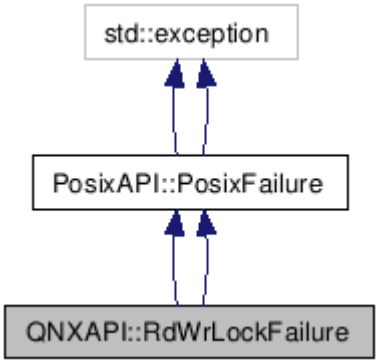
- include/qfc/.svn/text-base/lock.svn-base
- include/qfc/lock

QNXAPI::RdWrLockFailure Class Reference

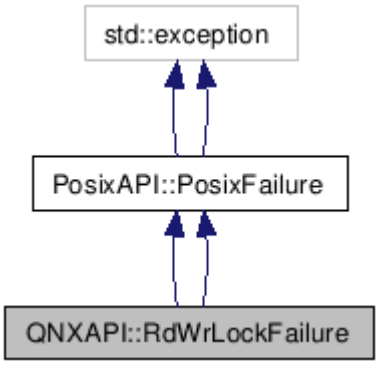
RdWrLockFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::RdWrLockFailure:



Collaboration diagram for QNXAPI::RdWrLockFailure:



Public Member Functions

- **RdWrLockFailure** (int err)
- **RdWrLockFailure** (int err)

Detailed Description

RdWrLockFailure class.

Exception encountered during read/write lock operations.

Definition at line 51 of file lock.svn-base.

Constructor & Destructor Documentation

QNXAPI::RdWrLockFailure::RdWrLockFailure (int err) [inline]

Construct a **RdWrLockFailure** class.

Parameters:

err standard error code.

Definition at line 59 of file lock.svn-base.QNXAPI::RdWrLockFailure::RdWrLockFailure (int err) [inline]

Construct a **RdWrLockFailure** class.

Parameters:

err standard error code.

Definition at line 59 of file lock.

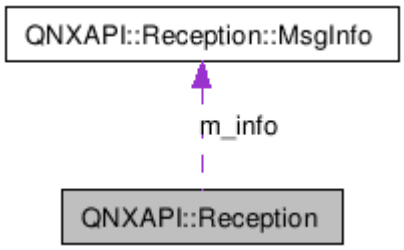
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/lock.svn-base
- include/qfc/lock

QNXAPI::Reception Class Reference

Reception class. Holds a receive context.

Collaboration diagram for QNXAPI::Reception:



Public Types

- enum **PulseCode**
- enum **PulseCode**

Public Member Functions

- **Reception** ()
  *Construct an instance of a **Reception** class.*

- **~Reception** (void)
  *Destroy an instance of a **Reception** class.*

- void **Id** (int id)

Set the receive id.

- int **Id** (void)
  Get the receive id.

- int **Code** (void)
  Get the pulse code.

- sigval **Value** (void)
  Get the pulse value.

- void * **Ptr** (void)
  Get a pointer to the start of the message.

- int **Len** (void)
  Get the length of a reception.

- _msg_info & **Info** (void)
  Get a reference to the message info.

- **Reception** ()
  *Construct an instance of a **Reception** class.*

- **~Reception** (void)
  *Destroy an instance of a **Reception** class.*

- void **Id** (int id)
  Set the receive id.

- int **Id** (void)
  Get the receive id.

- int **Code** (void)
  Get the pulse code.

- sigval **Value** (void)
  Get the pulse value.

- void * **Ptr** (void)
  Get a pointer to the start of the message.

- int **Len** (void)
  Get the length of a reception.

- _msg_info & **Info** (void)
  Get a reference to the message info.

Classes

- class **MsgInfo**
  Message info class, holds extra information pertinent to a received message.

Detailed Description

Reception class. Holds a receive context.

A Receive context is essentially the receive id, a space large enough to hold a pulse, and the message info data obtained during a receive.

Examples:

Cli/Cli.cpp, and **Serv/Serv.h**.

Definition at line 315 of file ipc.svn-base.

Member Enumeration Documentation

enum QNXAPI::Reception::PulseCode

Predefined pulse codes that can be received

Definition at line 321 of file ipc.svn-base.enum QNXAPI::Reception::PulseCode

Predefined pulse codes that can be received

Definition at line 321 of file ipc.

Member Function Documentation

void QNXAPI::Reception::Id (int id) [inline]

Set the receive id.

Parameters:

id the receive id to set the reception to.

Definition at line 390 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Dispatch(), QNXAPI::ThreadPool< T, V, R >::Context::Id(), QNXAPI::MessageVector< T, V, R >::Read(), QNXAPI::ThreadPool< T, V, R >::Context::Receive(), QNXAPI::Channel< T, V, R >::Receive(), and QNXAPI::ReplyVector< T, V >::Reply().int QNXAPI::Reception::Id (void) [inline]

Get the receive id.

Returns:

the receive id of the reception.

Definition at line 400 of file ipc.svn-base.int QNXAPI::Reception::Code (void) [inline]

Get the pulse code.

Returns:

the pulse code.

Examples:

Cli/Cli.cpp.

Definition at line 410 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Dispatch().union sigval QNXAPI::Reception::Value (void) [inline]

Get the pulse value.

Returns:

the pulse value.

Definition at line 420 of file ipc.svn-base.void* QNXAPI::Reception::Ptr (void) [inline]

Get a pointer to the start of the message.

Returns:

pointer to start of message.

Definition at line 430 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Receive().int QNXAPI::Reception::Len (void) [inline]

Get the length of a reception.

Returns:

size of reception.

Definition at line 440 of file ipc.svn-base.

Referenced by QNXAPI::Channel< T, V, R >::Receive().struct _msg_info& QNXAPI::Reception::Info (void) [inline]

Get a reference to the message info.

Returns:

reference to the message info (POD)

Definition at line 450 of file ipc.svn-base.

References QNXAPI::Reception::MsgInfo::Impl().

Referenced by QNXAPI::Channel< T, V, R >::Receive().

Here is the call graph for this function:

void QNXAPI::Reception::Id (int id) [inline]

Set the receive id.

Parameters:

id the receive id to set the reception to.

Definition at line 390 of file ipc.int QNXAPI::Reception::Id (void) [inline]

Get the receive id.

Returns:

the receive id of the reception.

Definition at line 400 of file ipc.int QNXAPI::Reception::Code (void) [inline]

Get the pulse code.

Returns:

the pulse code.

Definition at line 410 of file ipc.union sigval QNXAPI::Reception::Value (void) [inline]

Get the pulse value.

Returns:

the pulse value.

Definition at line 420 of file ipc.void* QNXAPI::Reception::Ptr (void) [inline]

Get a pointer to the start of the message.

Returns:

pointer to start of message.

Definition at line 430 of file ipc.int QNXAPI::Reception::Len (void) [inline]

Get the length of a reception.

Returns:
size of reception.

Definition at line 440 of file ipc.struct _msg_info& QNXAPI::Reception::Info (void) [inline]

Get a reference to the message info.

Returns:
reference to the message info (POD)

Definition at line 450 of file ipc.

References QNXAPI::Reception::MsgInfo::Impl().

Here is the call graph for this function:

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

- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::Reception::MsgInfo Class Reference

Message info class, holds extra information pertinent to a received message.

Public Member Functions

- **MsgInfo** ()
  Construct an instance of a message info class.

- **~MsgInfo** (void)
  Destroy an instance of a message info class.

- _msg_info & **Impl** (void)
  Obtain the underlying 'C' implementation of a message info struct.

- int **Len** (void)
  Obtain the length of the message that was received.

- **MsgInfo** ()
  Construct an instance of a message info class.

- **~MsgInfo** (void)
  Destroy an instance of a message info class.

- _msg_info & **Impl** (void)
  Obtain the underlying 'C' implementation of a message info struct.

- int **Len** (void)
  Obtain the length of the message that was received.

Detailed Description

Message info class, holds extra information pertinent to a received message.

Definition at line 331 of file ipc.svn-base.

Member Function Documentation

struct _msg_info& QNXAPI::Reception::MsgInfo::Impl (void) [inline]

Obtain the underlying 'C' implementation of a message info struct.

Returns:

Reference to 'C' _msg_info structure

Definition at line 352 of file ipc.svn-base.

Referenced by QNXAPI::Reception::Info().int QNXAPI::Reception::MsgInfo::Len (void) [inline]

Obtain the length of the message that was received.

Returns:
Length of received message

Definition at line 362 of file ipc.svn-base.struct _msg_info& QNXAPI::Reception::MsgInfo::Impl (void) [inline]

Obtain the underlying 'C' implementation of a message info struct.

Returns:
Reference to 'C' _msg_info structure

Definition at line 352 of file ipc.int QNXAPI::Reception::MsgInfo::Len (void) [inline]

Obtain the length of the message that was received.

Returns:
Length of received message

Definition at line 362 of file ipc.

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

- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::ReplyVector< T, V > Class Template Reference

ReplyVector class.

Public Member Functions

- **ReplyVector** ()
  *Construct an instance of a **ReplyVector** class.*

- virtual **~ReplyVector** ()
  *Destroy an instance of a **ReplyVector** class.*

- void **Add** (T item)
  Add a vector.

- void **Add** (void *ptr, size_t len)
  Add a simple vector.

- virtual void **Reply** (**Reception** &rcp, int status)
  *Reply via a **Reception**.*

- void **Clear** (void)
  Clear (erase) a reply vector.

- **ReplyVector** ()
  *Construct an instance of a **ReplyVector** class.*

- virtual **~ReplyVector** ()
  *Destroy an instance of a **ReplyVector** class.*

- void **Add** (T item)
  Add a vector.

- void **Add** (void *ptr, size_t len)
  Add a simple vector.

- virtual void **Reply** (**Reception** &rcp, int status)
  *Reply via a **Reception**.*

- void **Clear** (void)
  Clear (erase) a reply vector.

Public Attributes

- std::vector< T > **m_vect**
  storage for a simple vector of T

- std::vector< T > **m_vect**
  storage for a simple vector of T

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader> class QNXAPI::ReplyVector< T, V >

ReplyVector class.

Definition at line 537 of file ipc.svn-base.

Member Function Documentation

template<typename T = iovItem, typename V = SimpleVectLoader> void QNXAPI::ReplyVector< T, V >::Add (T* item) [inline]

Add a vector.

Parameters:

item a variant communicable type to add to vector.

Definition at line 559 of file ipc.svn-base.

References QNXAPI::ReplyVector< T, V >::m_vect.template<typename T = iovItem, typename V = SimpleVectLoader> void QNXAPI::ReplyVector< T, V >::Add (void * ptr, size_t len) [inline]

Add a simple vector.

Parameters:

ptr pointer to the start of the vector.
len length (magnitude) of the vector.

Definition at line 570 of file ipc.svn-base.

References QNXAPI::ReplyVector< T, V >::m_vect.template<typename T = iovItem, typename V = SimpleVectLoader> virtual void QNXAPI::ReplyVector< T, V >::Reply (Reception & rcp, int status) [inline, virtual]

Reply via a **Reception**.

Parameters:

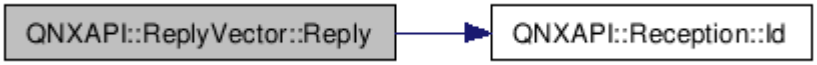
rcp reference to **Reception** to reply through.
status error status to accompany reply.

Definition at line 582 of file ipc.svn-base.

References QNXAPI::Reception::Id(), and QNXAPI::ReplyVector< T, V >::m_vect.

Referenced by QNXAPI::Channel< T, V, R >::Reply().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader> void QNXAPI::ReplyVector< T, V >::Add (T item) [inline]

Add a vector.

Parameters:

item a variant communicable type to add to vector.

Definition at line 559 of file ipc.

References QNXAPI::ReplyVector< T, V >::m_vect.template<typename T = iovItem, typename V = SimpleVectLoader> void QNXAPI::ReplyVector< T, V >::Add (void * ptr, size_t len) [inline]

Add a simple vector.

Parameters:

ptr pointer to the start of the vector.
len length (magnitude) of the vector.

Definition at line 570 of file ipc.

References QNXAPI::ReplyVector< T, V >::m_vect.template<typename T = iovItem, typename V = SimpleVectLoader> virtual void QNXAPI::ReplyVector< T, V >::Reply (Reception & rcp, int status) [inline, virtual]

Reply via a **Reception**.

Parameters:

rcp reference to **Reception** to reply through.
status error status to accompany reply.

Definition at line 582 of file ipc.

References QNXAPI::ReplyVector< T, V >::m_vect.

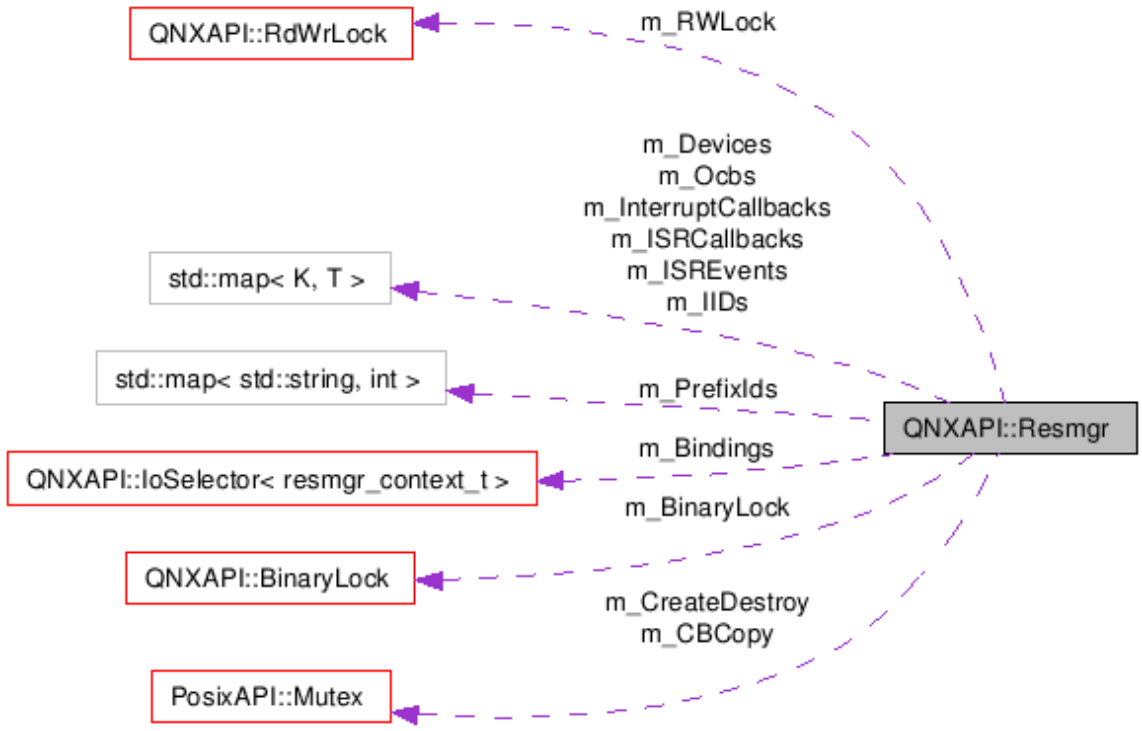
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::Resmgr Class Reference

The **Resmgr** class.

Collaboration diagram for QNXAPI::Resmgr:



Public Types

- typedef sigc::signal< const struct sigevent *, struct sigevent * > **ISRCallback**
  Callback for Interrupt Service Routine.

- typedef sigc::signal< void > **InterruptCallback**
  Callback for Interrupt handler.

- typedef sigc::signal< int, message_context_t *, int, unsigned > **PeriodicCallback**
  Callback for periodic operations.

- typedef sigc::signal< int, resmgr_context_t *, io_open_t *, **Device** *, void * > **OpenCallback**
  Callback for Open.

- typedef sigc::signal< int, resmgr_context_t *, io_unlink_t *, **Device** *, void * > **UnlinkCallback**
  Callback for Unlink.

- typedef sigc::signal< int, resmgr_context_t *, io_rename_t *, **Device** *, io_rename_extra_t * > **RenameCallback**
  Callback for Rename.

- typedef sigc::signal< int, resmgr_context_t *, io_mknod_t *, **Device** *, void * > **MknodCallback**
  Callback for Mknod.

- typedef sigc::signal< int, resmgr_context_t *, io_readlink_t *, **Device** *, void * > **ReadLinkCallback**
  Callback for Readlink.

- typedef sigc::signal< int, resmgr_context_t *, io_link_t *, **Device** *, io_link_extra_t * > **MakeLinkCallback**
  Callback for Makelink.

- typedef sigc::signal< int, resmgr_context_t *, io_pulse_t *, **Device** *, void * > **UnblockConCallback**
  Callback for Unblock connection.

- typedef sigc::signal< int, resmgr_context_t *, io_mount_t *, **Device** *, void * > **MountCallback**
  Callback for Mount.

- typedef sigc::signal< int, resmgr_context_t *, io_read_t *, **Ocb** * > **ReadCallback**
  Callback for Read.

- typedef sigc::signal< int, resmgr_context_t *, io_write_t *, **Ocb** * > **WriteCallback**
  Callback for Write.

- typedef sigc::signal< int, resmgr_context_t *, void *, **Ocb** * > **CloseOcbCallback**
  Callback for Close.

- typedef sigc::signal< int, resmgr_context_t *, io_stat_t *, **Ocb** * > **StatCallback**
  Callback for Stat.

- typedef sigc::signal< int, resmgr_context_t *, io_notify_t *, **Ocb** * > **NotifyCallback**
  Callback for IO Notify.

- typedef sigc::signal< int, resmgr_context_t *, io_devctl_t *, **Ocb** * > **DevctlCallback**
  Callback for Devctl.

- typedef sigc::signal< int, resmgr_context_t *, io_pulse_t *, **Ocb** * > **UnblockCallback**
  Callback for Unblock.

- typedef sigc::signal< int, resmgr_context_t *, io_pathconf_t *, **Ocb** * > **PathconfCallback**
  Callback for Pathconf.

- typedef sigc::signal< int, resmgr_context_t *, io_lseek_t *, **Ocb** * > **LseekCallback**
  Callback for Lseek.

- typedef sigc::signal< int, resmgr_context_t *, io_chmod_t *, **Ocb** * > **ChmodCallback**
  Callback for Chmod.

- typedef sigc::signal< int, resmgr_context_t *, io_chown_t *, **Ocb** * > **ChownCallback**
  Callback for Chown.

- typedef sigc::signal< int, resmgr_context_t *, io_utime_t *, **Ocb** * > **UtimeCallback**
  Callback for Utime.

- typedef sigc::signal< int, resmgr_context_t *, io_openfd_t *, **Ocb** * > **FdOpenCallback**
  Callback for FdOpen.

- typedef sigc::signal< int, resmgr_context_t *, io_fdinfo_t *, **Ocb** * > **FdInfoCallback**
  Callback for FdInfo.

- typedef sigc::signal< int, resmgr_context_t *, io_lock_t *, **Ocb** * > **LockCallback**
  Callback for Lock.

- typedef sigc::signal< int, resmgr_context_t *, io_space_t *, **Ocb** * > **SpaceCallback**
  Callback for Space.

- typedef sigc::signal< int, resmgr_context_t *, io_shutdown_t *, **Ocb** * > **ShutdownCallback**
  Callback for Shutdown.

- typedef sigc::signal< int, resmgr_context_t *, io_mmap_t *, **Ocb** * > **MmapCallback**
  Callback for Mmap.

- typedef sigc::signal< int, resmgr_context_t *, io_msg_t *, **Ocb** * > **MsgCallback**
  Callback for Msg.

- typedef sigc::signal< int, resmgr_context_t *, void *, **Ocb** * > **UmountCallback**
  Callback for Umount.

- typedef sigc::signal< int, resmgr_context_t *, io_dup_t *, **Ocb** * > **DupCallback**
  Callback for Dup.

- typedef sigc::signal< int, resmgr_context_t *, io_close_t *, **Ocb** * > **CloseDupCallback**
  Callback for Close.

- typedef sigc::signal< int, resmgr_context_t *, void *, **Ocb** * > **LockOcbCallback**
  Callback for LockOcb.

- typedef sigc::signal< int, resmgr_context_t *, void *, **Ocb** * > **UnlockOcbCallback**
  Callback for UnlockOcb.

- typedef std::map< IOFUNC_OCB_T *, **Ocb** * > **OcbMap**
  Map of ocbs.

- typedef std::map< int, struct sigevent > **ISREvents**
  Map of ISR events.

- typedef std::map< int, **ISRCallback** * > **ISRCallbacks**
  Map of ISR callbacks.

- typedef std::map< int, **InterruptCallback** * > **InterruptCallbacks**
  Map of Interrupt callbacks.

- typedef std::map< int, **IntrIID** > **IIDs**
  Map of Interrupt IDs.

- typedef short **RepeatRate**
  Repeat rate of periodic operations.

- typedef short **LowWater**
  storage for low water mark

- typedef unsigned char **Increment**
  storage for increment

- typedef int **HighWater**
  storage for high water mark

- typedef int **ThreadMax**
  storage for max threads

- typedef unsigned char **MsgPartsNum**
  storage for max msg parts

- typedef int **MsgSizeMax**
  storage for max msg size

- typedef sigc::signal< const struct sigevent *, struct sigevent * > **ISRCallback**
  Callback for Interrupt Service Routine.

- typedef sigc::signal< void > **InterruptCallback**
  Callback for Interrupt handler.

- typedef sigc::signal< int, message_context_t *, int, unsigned > **PeriodicCallback**
  Callback for periodic operations.

- typedef sigc::signal< int, resmgr_context_t *, io_open_t *, **Device** *, void * > **OpenCallback**
  Callback for Open.

- typedef sigc::signal< int, resmgr_context_t *, io_unlink_t *, **Device** *, void * > **UnlinkCallback**
  Callback for Unlink.

- typedef sigc::signal< int, resmgr_context_t *, io_rename_t *, **Device** *, io_rename_extra_t * > **RenameCallback**
  Callback for Rename.

- typedef sigc::signal< int, resmgr_context_t *, io_mknod_t *, **Device** *, void * > **MknodCallback**
  Callback for Mknod.

- typedef sigc::signal< int, resmgr_context_t *, io_readlink_t *, **Device** *, void * > **ReadLinkCallback**
  Callback for Readlink.

- typedef sigc::signal< int, resmgr_context_t *, io_link_t *, **Device** *, io_link_extra_t * > **MakeLinkCallback**
  Callback for Makelink.

- typedef sigc::signal< int, resmgr_context_t *, io_pulse_t *, **Device** *, void * > **UnblockConCallback**
  Callback for Unblock connection.

- typedef sigc::signal< int, resmgr_context_t *, io_mount_t *, **Device** *, void * > **MountCallback**
  Callback for Mount.

- typedef sigc::signal< int, resmgr_context_t *, io_read_t *, **Ocb** * > **ReadCallback**
  Callback for Read.

- typedef sigc::signal< int, resmgr_context_t *, io_write_t *, **Ocb** * > **WriteCallback**
  Callback for Write.

- typedef sigc::signal< int, resmgr_context_t *, void *, **Ocb** * > **CloseOcbCallback**
  Callback for Close.

- typedef sigc::signal< int, resmgr_context_t *, io_stat_t *, **Ocb** * > **StatCallback**
  Callback for Stat.

- typedef sigc::signal< int, resmgr_context_t *, io_notify_t *, **Ocb** * > **NotifyCallback**
  Callback for IO Notify.

- typedef sigc::signal< int, resmgr_context_t *, io_devctl_t *, **Ocb** * > **DevctlCallback**
  Callback for Devctl.

- typedef sigc::signal< int, resmgr_context_t *, io_pulse_t *, **Ocb** * > **UnblockCallback**
  Callback for Unblock.

- typedef sigc::signal< int, resmgr_context_t *, io_pathconf_t *, **Ocb** * > **PathconfCallback**
  Callback for Pathconf.

- typedef sigc::signal< int, resmgr_context_t *, io_lseek_t *, **Ocb** * > **LseekCallback**
  Callback for Lseek.

- typedef sigc::signal< int, resmgr_context_t *, io_chmod_t *, **Ocb** * > **ChmodCallback**
  Callback for Chmod.

- typedef sigc::signal< int, resmgr_context_t *, io_chown_t *, **Ocb** * > **ChownCallback**
  Callback for Chown.

- typedef sigc::signal< int, resmgr_context_t *, io_utime_t *, **Ocb** * > **UtimeCallback**
  Callback for Utime.

- typedef sigc::signal< int, resmgr_context_t *, io_openfd_t *, **Ocb** * > **FdOpenCallback**
  Callback for FdOpen.

- typedef sigc::signal< int, resmgr_context_t *, io_fdinfo_t *, **Ocb** * > **FdInfoCallback**
  Callback for FdInfo.

- typedef sigc::signal< int, resmgr_context_t *, io_lock_t *, **Ocb** * > **LockCallback**
  Callback for Lock.

- typedef sigc::signal< int, resmgr_context_t *, io_space_t *, **Ocb** * > **SpaceCallback**
  Callback for Space.

- typedef sigc::signal< int, resmgr_context_t *, io_shutdown_t *, **Ocb** * > **ShutdownCallback**
  Callback for Shutdown.

- typedef sigc::signal< int, resmgr_context_t *, io_mmap_t *, **Ocb** * > **MmapCallback**
  Callback for Mmap.

- typedef sigc::signal< int, resmgr_context_t *, io_msg_t *, **Ocb** * > **MsgCallback**
  Callback for Msg.

- typedef sigc::signal< int, resmgr_context_t *, void *, **Ocb** * > **UmountCallback**
  Callback for Umount.

- typedef sigc::signal< int, resmgr_context_t *, io_dup_t *, **Ocb** * > **DupCallback**
  Callback for Dup.

- typedef sigc::signal< int, resmgr_context_t *, io_close_t *, **Ocb** * > **CloseDupCallback**
  Callback for Close.

- typedef sigc::signal< int, resmgr_context_t *, void *, **Ocb** * > **LockOcbCallback**
  Callback for LockOcb.

- typedef sigc::signal< int, resmgr_context_t *, void *, **Ocb** * > **UnlockOcbCallback**
  Callback for UnlockOcb.

- typedef std::map< IOFUNC_OCB_T *, **Ocb** * > **OcbMap**
  Map of ocbs.

- typedef std::map< int, struct sigevent > **ISREvents**
  Map of ISR events.

- typedef std::map< int, **ISRCallback** * > **ISRCallbacks**
  Map of ISR callbacks.

- typedef std::map< int, **InterruptCallback** * > **InterruptCallbacks**
  Map of Interrupt callbacks.

- typedef std::map< int, **IntrIID** > **IIDs**
  Map of Interrupt IDs.

- typedef short **RepeatRate**
  Repeat rate of periodic operations.

- typedef short **LowWater**
  storage for low water mark

- typedef unsigned char **Increment**
  storage for increment

- typedef int **HighWater**
  storage for high water mark

- typedef int **ThreadMax**
  storage for max threads

- typedef unsigned char **MsgPartsNum**
  storage for max msg parts

- typedef int **MsgSizeMax**
  storage for max msg size

Public Member Functions

- **Resmgr** (bool exitSelf=true, bool autoConcurrency=true)
  Construct a resource manager.

- virtual **~Resmgr** ()
  Destroy a resource manager.

- void **Init** (bool exitSelf=false, bool autoConcurrency=false, bool endian=true)
  Init initialize a resource manager.

- bool **HasFlagAfter** ()
  Test for presence of _RESMGR_FLAG_AFTER.

- bool **HasFlagBefore** ()
  Test for presence of _RESMGR_FLAG_BEFORE.

- bool **HasFlagOpaque** ()
  Test for presence of _RESMGR_FLAG_OPAQUE.

- bool **HasFlagDir** ()
  Test for presence of _RESMGR_FLAG_DIR.

- bool **HasFlagFtypeOnly** ()
  Test for presence of _RESMGR_FLAG_FTYPEONLY.

- bool **HasFlagSelf** ()
  Test for presence of _RESMGR_FLAG_SELF.

- void **Flags** (unsigned flags)
  Set the resource manager flags.

- void **Type** (enum _file_type type)
  Set the resource manager type.

- enum _file_type **Type** (void)
  Get the resource manager type.

- resmgr_connect_funcs_t * **ConnectFuncs** (void)
  Get the resource manager connect funcs structure.

- resmgr_io_funcs_t * **IoFuncs** (void)
  Get the resource manager io funcs structure.

- void **Attach** (const std::string &prefix)
  Attach the resource manager prefix.

- void **Attach** (const std::string &prefix, **Device** &**device**)
  Attach the resource manager prefix.

- void **Attach** (const std::string &prefix, **Device** &**device**, enum _file_type type)
  Attach the resource manager prefix.

- void **Attach** (const std::string &prefix, **Device** &**device**, enum _file_type type, unsigned flags)
  Attach the resource manager prefix.

- virtual struct **ocb** * **CreateOcb** (void)
  Create an Open Control Block.

- void **DestroyOcb** (struct **ocb** ***ocb**)
  Destroy an Open Control Block.

- void **RegisterOcb** (resmgr_context_t *ctp, io_open_t *msg, IOFUNC_ATTR_T *attr, IOFUNC_OCB_T ***ocb**, **Ocb** *derived) throw ()
  Register an Open Control Block.

- int **DeregisterOcb** (resmgr_context_t *ctp, IOFUNC_OCB_T ***ocb**) throw ()
  De-register an Open Control Block.

- **Resmgr::Ocb** * **operator[]** (IOFUNC_OCB_T ***ocb**)
  Index an Open Control Block.

- int **DispatchInterruptHandler** (message_context_t *mcp, int code, unsigned flags) throw ()
  Dispatch an interrupt handler.

- void **RegisterInterrupt** (int intr, int pri, sigc::slot< const struct sigevent *, struct sigevent * > isr, sigc::slot< void > interrupt_handler, bool end=false)
  Register an interrupt handler.

- void **RegisterInterrupt** (int intr, int pri, sigc::slot< void > interrupt_handler, bool end=false)
  Register an interrupt handler.

- void **RegisterPeriodic** (sigc::slot< int, message_context_t *, int, unsigned > periodic_task, **RepeatRate** rate=1000)
  Register a periodic handler.

- void **RegisterOpen** (**Device** *, sigc::slot< int, resmgr_context_t *, io_open_t *, **Device** *, void * > open_handler)
  Register a Open handler.

- void **RegisterMknod** (**Device** *, sigc::slot< int, resmgr_context_t *, io_mknod_t *, **Device** *, void * > mknod_handler)
  Register a Mknod handler.

- void **RegisterUnlink** (**Device** *, sigc::slot< int, resmgr_context_t *, io_unlink_t *, **Device** *, void * > unlink_handler)
  Register a Unlink handler.

- void **RegisterRename** (**Device** *, sigc::slot< int, resmgr_context_t *, io_rename_t *, **Device** *, io_rename_extra_t * > rename_handler)
  Register a Rename handler.

- void **RegisterMakeLink** (**Device** *, sigc::slot< int, resmgr_context_t *, io_link_t *, **Device** *, io_link_extra_t * > makelink_handler)
  Register a MakeLink handler.

- void **RegisterReadLink** (**Device** *, sigc::slot< int, resmgr_context_t *, io_readlink_t *, **Device** *, void * > readlink_handler)
  Register a ReadLink handler.

- void **RegisterCloseOcb** (**Device** *, sigc::slot< int, resmgr_context_t *, void *, **Ocb** * > closeocb_handler)
  Register a CloseOcb handler.

- void **RegisterRead** (**Device** *, sigc::slot< int, resmgr_context_t *, io_read_t *, **Ocb** * > read_handler)
  Register a Read handler.

- void **RegisterWrite** (**Device** *, sigc::slot< int, resmgr_context_t *, io_write_t *, **Ocb** * > write_handler)
  Register a Write handler.

- void **Background** (void)
  Place calling process in the background.

- void **DetachAll** (const std::string &prefix)
  Detach all resources associated with prefix.

- void **DetachPathName** (const std::string &prefix)
  Detach prefix only.

- **QNXAPI::IoSelector**< resmgr_context_t >::BindingMap & **SelectorBindings** (void) throw ()
  Obtain a reference to an instance of the BindingMap attached to the selector.

- bool **AlwaysLoop** (int err) const throw ()
  *Default callback (supplied to **Run()**) that will cause the resource manager loop to never cease unless an error occurs.*

- int **Run** (void)
  Run the resource manager using the AlwaysLoop callback (above).

- int **Run** (sigc::slot< bool, int >loopControl)
  Run the resource manager using a client supplied loop callback (above).

- **Resmgr** (bool exitSelf=true, bool autoConcurrency=true)
  Construct a resource manager.

- virtual **~Resmgr** ()
  Destroy a resource manager.

- void **Init** (bool exitSelf=false, bool autoConcurrency=false, bool endian=true)
  Init initialize a resource manager.

- bool **HasFlagAfter** ()
  Test for presence of _RESMGR_FLAG_AFTER.

- bool **HasFlagBefore** ()
  Test for presence of _RESMGR_FLAG_BEFORE.

- bool **HasFlagOpaque** ()
  Test for presence of _RESMGR_FLAG_OPAQUE.

- bool **HasFlagDir** ()
  Test for presence of _RESMGR_FLAG_DIR.

- bool **HasFlagFtypeOnly** ()
  Test for presence of _RESMGR_FLAG_FTYPEONLY.

- bool **HasFlagSelf** ()
  Test for presence of _RESMGR_FLAG_SELF.

- void **Flags** (unsigned flags)
  Set the resource manager flags.

- void **Type** (enum _file_type type)
  Set the resource manager type.

- enum _file_type **Type** (void)
  Get the resource manager type.

- resmgr_connect_funcs_t * **ConnectFuncs** (void)
  Get the resource manager connect funcs structure.

- resmgr_io_funcs_t * **IoFuncs** (void)
  Get the resource manager io funcs structure.

- void **Attach** (const std::string &prefix)
  Attach the resource manager prefix.

- void **Attach** (const std::string &prefix, **Device** &**device**)
  Attach the resource manager prefix.

- void **Attach** (const std::string &prefix, **Device** &**device**, enum _file_type type)
  Attach the resource manager prefix.

- void **Attach** (const std::string &prefix, **Device** &**device**, enum _file_type type, unsigned flags)
  Attach the resource manager prefix.

- virtual struct **ocb** * **CreateOcb** (void)
  Create an Open Control Block.

- void **DestroyOcb** (struct **ocb** ***ocb**)
  Destroy an Open Control Block.

- void **RegisterOcb** (resmgr_context_t *ctp, io_open_t *msg, IOFUNC_ATTR_T *attr, IOFUNC_OCB_T ***ocb**, **Ocb** *derived) throw ()
  Register an Open Control Block.

- int **DeregisterOcb** (resmgr_context_t *ctp, IOFUNC_OCB_T ***ocb**) throw ()
  De-register an Open Control Block.

- **Resmgr::Ocb** * **operator[]** (IOFUNC_OCB_T ***ocb**)
  Index an Open Control Block.

- int **DispatchInterruptHandler** (message_context_t *mcp, int code, unsigned flags) throw ()
  Dispatch an interrupt handler.

- void **RegisterInterrupt** (int intr, int pri, sigc::slot< const struct sigevent *, struct sigevent * > isr, sigc::slot< void > interrupt_handler, bool end=false)
  Register an interrupt handler.

- void **RegisterInterrupt** (int intr, int pri, sigc::slot< void > interrupt_handler, bool end=false)
  Register an interrupt handler.

- void **RegisterPeriodic** (sigc::slot< int, message_context_t *, int, unsigned > periodic_task, **RepeatRate** rate=1000)
  Register a periodic handler.

- void **RegisterOpen** (**Device** *, sigc::slot< int, resmgr_context_t *, io_open_t *, **Device** *, void * > open_handler)
  Register a Open handler.

- void **RegisterMknod** (**Device** *, sigc::slot< int, resmgr_context_t *, io_mknod_t *, **Device** *, void * > mknod_handler)
  Register a Mknod handler.

- void **RegisterUnlink** (**Device** *, sigc::slot< int, resmgr_context_t *, io_unlink_t *, **Device** *, void * > unlink_handler)
  Register a Unlink handler.

- void **RegisterRename** (**Device** *, sigc::slot< int, resmgr_context_t *, io_rename_t *, **Device** *, io_rename_extra_t * > rename_handler)
  Register a Rename handler.

- void **RegisterMakeLink** (**Device** *, sigc::slot< int, resmgr_context_t *, io_link_t *, **Device** *, io_link_extra_t * > makelink_handler)
  Register a MakeLink handler.

- void **RegisterReadLink** (**Device** *, sigc::slot< int, resmgr_context_t *, io_readlink_t *, **Device** *, void * > readlink_handler)
  Register a ReadLink handler.

- void **RegisterCloseOcb** (**Device** *, sigc::slot< int, resmgr_context_t *, void *, **Ocb** * > closeocb_handler)
  Register a CloseOcb handler.

- void **RegisterRead** (**Device** *, sigc::slot< int, resmgr_context_t *, io_read_t *, **Ocb** * > read_handler)
  Register a Read handler.

- void **RegisterWrite** (**Device** *, sigc::slot< int, resmgr_context_t *, io_write_t *, **Ocb** * > write_handler)
  Register a Write handler.

- void **Background** (void)
  Place calling process in the background.

- void **DetachAll** (const std::string &prefix)
  Detach all resources associated with prefix.

- void **DetachPathName** (const std::string &prefix)
  Detach prefix only.

- **QNXAPI::IoSelector**< resmgr_context_t >::BindingMap & **SelectorBindings** (void) throw ()
  Obtain a reference to an instance of the BindingMap attached to the selector.

- bool **AlwaysLoop** (int err) const throw ()
  *Default callback (supplied to **Run()**) that will cause the resource manager loop to never cease unless an error occurs.*

- int **Run** (void)
  Run the resource manager using the AlwaysLoop callback (above).

- int **Run** (sigc::slot< bool, int >loopControl)
  Run the resource manager using a client supplied loop callback (above).

Public Attributes

- **ISREvents m_ISREvents**
  storage for ISR events

- **ISRCallbacks m_ISRCallbacks**
  storage for ISR callbacks

- **InterruptCallbacks m_InterruptCallbacks**
  storage for Interrupt callbacks

- **IIDs m_IIDs**
  storage for interrupt IDs

- intrspin_t **m_Spinlock**
  storage for spin lock

Protected Member Functions

- void **ThreadPoolAttrInit** (bool autoConcurrency=true, **LowWater** lw=1, **Increment** inc=0, **HighWater** hw=1, **ThreadMax** max=1)
  Initialize thread pool attributes.

- void **ResmgrAttrInit** (**MsgPartsNum** nparts_max=10, **MsgSizeMax** msg_max_size=2048, bool endian=true)
  Initialize resource manager attributes.

- void **ThreadPoolAttrInit** (thread_pool_attr_t &thread_pool_attr)
  Initialize thread pool attributes from a 'C' attr structute.

- void **ResmgrAttrInit** (resmgr_attr_t &resmgr_attr)
  Initialize resource manager attributes from a 'C' attr structure.

- void **InitializePeriodicScheduler** (**RepeatRate** rate, int prio=0)
  Initialize periodic scheduler.

- void **PrefixAttach** (const std::string &prefix, **Device** &**device**, enum _file_type type=_FTYPE_ANY, unsigned flags=0)
  Attach a prefix into the namespace.

- void **PrefixAttach** (const std::string &prefix, enum _file_type type=_FTYPE_ANY, unsigned flags=0)
  Attach a prefix into the namespace.

- void **ThreadPoolAttrInit** (bool autoConcurrency=true, **LowWater** lw=1, **Increment** inc=0, **HighWater** hw=1, **ThreadMax** max=1)
  Initialize thread pool attributes.

- void **ResmgrAttrInit** (**MsgPartsNum** nparts_max=10, **MsgSizeMax** msg_max_size=2048, bool endian=true)
  Initialize resource manager attributes.

- void **ThreadPoolAttrInit** (thread_pool_attr_t &thread_pool_attr)
  Initialize thread pool attributes from a 'C' attr structute.

- void **ResmgrAttrInit** (resmgr_attr_t &resmgr_attr)
  Initialize resource manager attributes from a 'C' attr structure.

- void **InitializePeriodicScheduler** (**RepeatRate** rate, int prio=0)
  Initialize periodic scheduler.

- void **PrefixAttach** (const std::string &prefix, **Device** &**device**, enum _file_type type=_FTYPE_ANY, unsigned flags=0)
  Attach a prefix into the namespace.

- void **PrefixAttach** (const std::string &prefix, enum _file_type type=_FTYPE_ANY, unsigned flags=0)
  Attach a prefix into the namespace.

Friends

- int **PeriodicRedir** (message_context_t *mcp, int code, unsigned flags, void *handle) throw ()
  *Redirect a QNX 'C' library pulse_attach callback (attached to a periodic timer), into a **Resmgr** functor of the same purpose.*

- int **OpenRedir** (resmgr_context_t *ctp, io_open_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_open callback into a **Resmgr** functor of the same purpose.*

- int **UnlinkRedir** (resmgr_context_t *ctp, io_unlink_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_unlink callback into a **Resmgr** functor of the same purpose.*

- int **RenameRedir** (resmgr_context_t *ctp, io_rename_t *msg, RESMGR_HANDLE_T *handle, io_rename_extra_t *extra) throw ()
  *Redirect a QNX 'C' library iofunc_rename callback into a **Resmgr** functor of the same purpose.*

- int **MknodRedir** (resmgr_context_t *ctp, io_mknod_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_mknod callback into a **Resmgr** functor of the same purpose.*

- int **ReadLinkRedir** (resmgr_context_t *ctp, io_readlink_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_readlink callback into a **Resmgr** functor of the same purpose.*

- int **MakeLinkRedir** (resmgr_context_t *ctp, io_link_t *msg, RESMGR_HANDLE_T *handle, io_link_extra_t *extra) throw ()
  *Redirect a QNX 'C' library iofunc_link callback into a **Resmgr** functor of the same purpose.*

- int **UnblockConRedir** (resmgr_context_t *ctp, io_pulse_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_unblock callback into a **Resmgr** functor of the same purpose.*

- int **MountRedir** (resmgr_context_t *ctp, io_mount_t *msg, RESMGR_HANDLE_T *handle, io_mount_extra_t *extra) throw ()
  *Redirect a QNX 'C' library iofunc_mount callback into a **Resmgr** functor of the same purpose.*

- int **ReadRedir** (resmgr_context_t *ctp, io_read_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_read callback into a **Resmgr** functor of the same purpose.*

- int **WriteRedir** (resmgr_context_t *ctp, io_write_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_write callback into a **Resmgr** functor of the same purpose.*

- int **CloseRedir** (resmgr_context_t *ctp, void *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_close callback into a **Resmgr** functor of the same purpose.*

- int **StatRedir** (resmgr_context_t *ctp, io_stat_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_stat callback into a **Resmgr** functor of the same purpose.*

- int **NotifyRedir** (resmgr_context_t *ctp, io_notify_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_notify callback into a **Resmgr** functor of the same purpose.*

- int **DevctlRedir** (resmgr_context_t *ctp, io_devctl_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_devctl callback into a **Resmgr** functor of the same purpose.*

- int **UnblockRedir** (resmgr_context_t *ctp, io_pulse_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_unblock callback into a **Resmgr** functor of the same purpose.*

- int **PathconfRedir** (resmgr_context_t *ctp, io_pathconf_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_pathconf callback into a **Resmgr** functor of the same purpose.*

- int **LseekRedir** (resmgr_context_t *ctp, io_lseek_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_lseek callback into a **Resmgr** functor of the same purpose.*

- int **ChmodRedir** (resmgr_context_t *ctp, io_chmod_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_chmod callback into a **Resmgr** functor of the same purpose.*

- int **ChownRedir** (resmgr_context_t *ctp, io_chown_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_chown callback into a **Resmgr** functor of the same purpose.*

- int **UtimeRedir** (resmgr_context_t *ctp, io_utime_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_utime callback into a **Resmgr** functor of the same purpose.*

- int **FdOpenRedir** (resmgr_context_t *ctp, io_openfd_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_openfd callback into a **Resmgr** functor of the same purpose.*

- int **FdInfoRedir** (resmgr_context_t *ctp, io_fdinfo_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_fdinfo callback into a **Resmgr** functor of the same purpose.*

- int **LockRedir** (resmgr_context_t *ctp, io_lock_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_lock callback into a **Resmgr** functor of the same purpose.*

- int **SpaceRedir** (resmgr_context_t *ctp, io_space_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_space callback into a **Resmgr** functor of the same purpose.*

- int **ShutdownRedir** (resmgr_context_t *ctp, io_shutdown_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_shutdown callback into a **Resmgr** functor of the same purpose.*

- int **MmapRedir** (resmgr_context_t *ctp, io_mmap_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_mmap callback into a **Resmgr** functor of the same purpose.*

- int **MsgRedir** (resmgr_context_t *ctp, io_msg_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_msg callback into a **Resmgr** functor of the same purpose.*

- int **UmountRedir** (resmgr_context_t *ctp, void *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_umount callback into a **Resmgr** functor of the same purpose.*

- int **DupRedir** (resmgr_context_t *ctp, io_dup_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_dup callback into a **Resmgr** functor of the same purpose.*

- int **CloseDupRedir** (resmgr_context_t *ctp, io_close_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_close_dup callback into a **Resmgr** functor of the same purpose.*

- int **LockOcbRedir** (resmgr_context_t *ctp, void *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_lock_ocb callback into a **Resmgr** functor of the same purpose.*

- int **UnlockOcbRedir** (resmgr_context_t *ctp, void *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_unlock_ocb callback into a **Resmgr** functor of the same purpose.*

- int **PeriodicRedir** (message_context_t *mcp, int code, unsigned flags, void *handle) throw ()
  *Redirect a QNX 'C' library pulse_attach callback (attached to a periodic timer), into a **Resmgr** functor of the same purpose.*

- int **OpenRedir** (resmgr_context_t *ctp, io_open_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_open callback into a **Resmgr** functor of the same purpose.*

- int **UnlinkRedir** (resmgr_context_t *ctp, io_unlink_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_unlink callback into a **Resmgr** functor of the same purpose.*

- int **RenameRedir** (resmgr_context_t *ctp, io_rename_t *msg, RESMGR_HANDLE_T *handle, io_rename_extra_t *extra) throw ()
  *Redirect a QNX 'C' library iofunc_rename callback into a **Resmgr** functor of the same purpose.*

- int **MknodRedir** (resmgr_context_t *ctp, io_mknod_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_mknod callback into a **Resmgr** functor of the same purpose.*

- int **ReadLinkRedir** (resmgr_context_t *ctp, io_readlink_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_readlink callback into a **Resmgr** functor of the same purpose.*

- int **MakeLinkRedir** (resmgr_context_t *ctp, io_link_t *msg, RESMGR_HANDLE_T *handle, io_link_extra_t *extra) throw ()
  *Redirect a QNX 'C' library iofunc_link callback into a **Resmgr** functor of the same purpose.*

- int **UnblockConRedir** (resmgr_context_t *ctp, io_pulse_t *msg, RESMGR_HANDLE_T *handle, void *extra) throw ()
  *Redirect a QNX 'C' library iofunc_unblock callback into a **Resmgr** functor of the same purpose.*

- int **MountRedir** (resmgr_context_t *ctp, io_mount_t *msg, RESMGR_HANDLE_T *handle, io_mount_extra_t *extra) throw ()
  *Redirect a QNX 'C' library iofunc_mount callback into a **Resmgr** functor of the same purpose.*

- int **ReadRedir** (resmgr_context_t *ctp, io_read_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_read callback into a **Resmgr** functor of the same purpose.*

- int **WriteRedir** (resmgr_context_t *ctp, io_write_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_write callback into a **Resmgr** functor of the same purpose.*

- int **CloseRedir** (resmgr_context_t *ctp, void *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_close callback into a **Resmgr** functor of the same purpose.*

- int **StatRedir** (resmgr_context_t *ctp, io_stat_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_stat callback into a **Resmgr** functor of the same purpose.*

- int **NotifyRedir** (resmgr_context_t *ctp, io_notify_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_notify callback into a **Resmgr** functor of the same purpose.*

- int **DevctlRedir** (resmgr_context_t *ctp, io_devctl_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_devctl callback into a **Resmgr** functor of the same purpose.*

- int **UnblockRedir** (resmgr_context_t *ctp, io_pulse_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_unblock callback into a **Resmgr** functor of the same purpose.*

- int **PathconfRedir** (resmgr_context_t *ctp, io_pathconf_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_pathconf callback into a **Resmgr** functor of the same purpose.*

- int **LseekRedir** (resmgr_context_t *ctp, io_lseek_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_lseek callback into a **Resmgr** functor of the same purpose.*

- int **ChmodRedir** (resmgr_context_t *ctp, io_chmod_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_chmod callback into a **Resmgr** functor of the same purpose.*

- int **ChownRedir** (resmgr_context_t *ctp, io_chown_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_chown callback into a **Resmgr** functor of the same purpose.*

- int **UtimeRedir** (resmgr_context_t *ctp, io_utime_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_utime callback into a **Resmgr** functor of the same purpose.*

- int **FdOpenRedir** (resmgr_context_t *ctp, io_openfd_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_openfd callback into a **Resmgr** functor of the same purpose.*

- int **FdInfoRedir** (resmgr_context_t *ctp, io_fdinfo_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_fdinfo callback into a **Resmgr** functor of the same purpose.*

- int **LockRedir** (resmgr_context_t *ctp, io_lock_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_lock callback into a **Resmgr** functor of the same purpose.*

- int **SpaceRedir** (resmgr_context_t *ctp, io_space_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_space callback into a **Resmgr** functor of the same purpose.*

- int **ShutdownRedir** (resmgr_context_t *ctp, io_shutdown_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_shutdown callback into a **Resmgr** functor of the same purpose.*

- int **MmapRedir** (resmgr_context_t *ctp, io_mmap_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_mmap callback into a **Resmgr** functor of the same purpose.*

- int **MsgRedir** (resmgr_context_t *ctp, io_msg_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_msg callback into a **Resmgr** functor of the same purpose.*

- int **UmountRedir** (resmgr_context_t *ctp, void *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_umount callback into a **Resmgr** functor of the same purpose.*

- int **DupRedir** (resmgr_context_t *ctp, io_dup_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_dup callback into a **Resmgr** functor of the same purpose.*

- int **CloseDupRedir** (resmgr_context_t *ctp, io_close_t *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_close_dup callback into a **Resmgr** functor of the same purpose.*

- int **LockOcbRedir** (resmgr_context_t *ctp, void *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_lock_ocb callback into a **Resmgr** functor of the same purpose.*

- int **UnlockOcbRedir** (resmgr_context_t *ctp, void *msg, RESMGR_OCB_T ***ocb**) throw ()
  *Redirect a QNX 'C' library iofunc_unlock_ocb callback into a **Resmgr** functor of the same purpose.*

Classes

- class **AttrLock**
  Attribute Lock class.

- class **Device**
  *Resource Manager **Device** class.*

- struct **IntrIID**
  Pair for associating IRQ and IID.

- class **Ocb**
  Open Control Block class.

- class **ResmgrDeviceContext**
- class **ResmgrIoSelector**

Detailed Description

The **Resmgr** class.

Encapsulation of the QNX resource manager library.

Author:

`renniеallen@gmail.com`

Examples:

Ramdisk/main.cpp.

Definition at line 317 of file resmgr.svn-base.

Member Function Documentation

void QNXAPI::Resmgr::Flags (unsigned* flags*)

Set the resource manager flags.

Parameters:

flags flags

void QNXAPI::Resmgr::Type (enum _file_type* type*)

Set the resource manager type.

Parameters:

type file type.

void QNXAPI::Resmgr::Attach (const std::string & prefix)

Attach the resource manager prefix.

Parameters:

prefix prefix to register.

Examples:

Ramdisk/main.cpp.

void QNXAPI::Resmgr::Attach (const std::string & prefix, Device & device)

Attach the resource manager prefix.

Parameters:

prefix prefix to register.
device device to attach.

void QNXAPI::Resmgr::Attach (const std::string & prefix, Device & device, enum _file_type type)

Attach the resource manager prefix.

Parameters:

prefix prefix to register.
device device to attach.
type file type of registration.

void QNXAPI::Resmgr::Attach (const std::string & prefix, Device & device, enum _file_type type, unsigned flags)

Attach the resource manager prefix.

Parameters:

prefix prefix to register.
device device to attach.
type file type of registration.
flags flags

virtual struct ocb* QNXAPI::Resmgr::CreateOcb (void) [virtual]

Create an Open Control Block.

Returns:

pointer to ocb.

void QNXAPI::Resmgr::DestroyOcb (struct ocb * ocb)

Destroy an Open Control Block.

Parameters:

ocb pointer to ocb.

void QNXAPI::Resmgr::RegisterOcb (resmgr_context_t * ctp, io_open_t * msg, IOFUNC_ATTR_T * attr, IOFUNC_OCB_T * ocb, Ocb * derived) throw ()

Register an Open Control Block.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_close_t.
attr pointer to a QNX 'C' iofunc attributes structure.
ocb pointer to a QNX 'C' ocb.
derived ocb

int QNXAPI::Resmgr::DeregisterOcb (resmgr_context_t * ctp, IOFUNC_OCB_T * ocb) throw ()

De-register an Open Control Block.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
ocb pointer to a QNX 'C' ocb.

Returns:

small positive integer from errno.h.

Resmgr::Ocb* QNXAPI::Resmgr::operator[] (IOFUNC_OCB_T * ocb) [inline]

Index an Open Control Block.

Returns:

pointer to ocb.

Definition at line 2079 of file resmgr.svn-base.int QNXAPI::Resmgr::DispatchInterruptHandler (message_context_t * mcp, int code, unsigned flags) throw ()

Dispatch an interrupt handler.

Parameters:

mcp pointer to a QNX 'C' library message_context_t.
code code
flags flags

Returns:

small positive integer from errno.h.

void QNXAPI::Resmgr::RegisterInterrupt (int intr, int pri, sigc::slot< const struct sigevent *, struct sigevent * > isr, sigc::slot< void > interrupt_handler, bool end = false)

Register an interrupt handler.

Parameters:

intr IRQ.
pri priority.
isr ISR.
interrupt_handler interrupt handler callback.
end add to end (true) or front (false) of handler list.

void QNXAPI::Resmgr::RegisterInterrupt (int intr, int pri, sigc::slot< void > interrupt_handler, bool end = false)

Register an interrupt handler.

Parameters:

intr IRQ.
pri priority.
interrupt_handler interrupt handler callback.
end add to end (true) or front (false) of handler list.

void QNXAPI::Resmgr::RegisterPeriodic (sigc::slot< int, message_context_t *, int, unsigned > periodic_task, RepeatRate rate = 1000)

Register a periodic handler.

Parameters:

periodic_task periodic handler callback.
rate frequency.

void QNXAPI::Resmgr::RegisterOpen (Device *, sigc::slot< int, resmgr_context_t *, io_open_t *, Device *, void * > open_handler)

Register a Open handler.

Parameters:

open_handler open handler callback.

void QNXAPI::Resmgr::RegisterMknod (Device *, sigc::slot< int, resmgr_context_t *, io_mknod_t *, Device *, void * > mknod_handler)

Register a Mknod handler.

Parameters:

mknod_handler mknod handler callback.

void QNXAPI::Resmgr::RegisterUnlink (Device *, sigc::slot< int, resmgr_context_t *, io_unlink_t *, Device *, void * > unlink_handler)

Register a Unlink handler.

Parameters:

unlink_handler unlink handler callback.

void QNXAPI::Resmgr::RegisterRename (Device *, sigc::slot< int, resmgr_context_t *, io_rename_t *, Device *, io_rename_extra_t * > rename_handler)

Register a Rename handler.

Parameters:

rename_handler rename handler callback.

void QNXAPI::Resmgr::RegisterMakeLink (Device *, sigc::slot< int, resmgr_context_t *, io_link_t *, Device *, io_link_extra_t * > makelink_handler)

Register a MakeLink handler.

Parameters:

makelink_handler makelink handler callback.

void QNXAPI::Resmgr::RegisterReadLink (Device *, sigc::slot< int, resmgr_context_t *, io_readlink_t *, Device *, void * > readlink_handler)

Register a ReadLink handler.

Parameters:

readlink_handler readlink handler callback.

void QNXAPI::Resmgr::RegisterCloseOcb (Device *, sigc::slot< int, resmgr_context_t *, void *, Ocb * > closeocb_handler)

Register a CloseOcb handler.

Parameters:

closeocb_handler closeocb handler callback.

void QNXAPI::Resmgr::RegisterRead (Device *, sigc::slot< int, resmgr_context_t *, io_read_t *, Ocb * > read_handler)

Register a Read handler.

Parameters:

read_handler read handler callback.

void QNXAPI::Resmgr::RegisterWrite (Device *, sigc::slot< int, resmgr_context_t *, io_write_t *, Ocb * > write_handler)

Register a Write handler.

Parameters:

write_handler write handler callback.

void QNXAPI::Resmgr::DetachAll (const std::string & prefix)

Detach all resources associated with prefix.

Parameters:

prefix name of prefix to detach.

void QNXAPI::Resmgr::DetachPathName (const std::string & prefix)

Detach prefix only.

Parameters:

prefix name of prefix to detach.

QNXAPI::IoSelector<resmgr_context_t>::BindingMap& QNXAPI::Resmgr::SelectorBindings (void) throw ()

Obtain a reference to an instance of the BindingMap attached to the selector.

Returns:

reference to an instance of a BindingMap

bool QNXAPI::Resmgr::AlwaysLoop (int* err*) const throw ()

Default callback (supplied to **Run()**) that will cause the resource manager loop to never cease unless an error occurs.

Client code can supply a different function to **Run()**.

Parameters:

err this is the error returned from the preceding resmgr_handler call.

Returns:

true if err is not an error, false otherwise.

int QNXAPI::Resmgr::Run (void)

Run the resource manager using the AlwaysLoop callback (above).

Returns:

small positive integer from errno.h.

Examples:

Ramdisk/main.cpp.

int QNXAPI::Resmgr::Run (sigc::slot< bool, int >* loopControl*)

Run the resource manager using a client supplied loop callback (above).

Parameters:

loopControl client supplied loop control callback.

Returns:

small positive integer from errno.h.

void QNXAPI::Resmgr::ThreadPoolAttrInit (bool* autoConcurrency *= true, LowWater* lw *= 1, Increment* inc *= 0, HighWater* hw *= 1, ThreadMax* max *= 1) [protected]

Initialize thread pool attributes.

Parameters:

autoConcurrency have the resource manager select the optimal level of concurrency.
lw lower water for thread pool.
inc thread increment quantity.
hw high water for thread pool.
max maximum thread count for the pool.

void QNXAPI::Resmgr::ResmgrAttrInit (MsgPartsNum* nparts_max = *10, MsgSizeMax* msg_max_size = *2048, bool* endian = *true) [protected]

Initialize resource manager attributes.

Parameters:

nparts_max maximum number of message parts.
msg_max_size maximum size of a message.
endian support bi-endian messages.

void QNXAPI::Resmgr::ThreadPoolAttrInit (thread_pool_attr_t &* thread_pool_attr*) [protected]

Initialize thread pool attributes from a 'C' attr structute.

Parameters:

thread_pool_attr 'C' attributes structure.

void QNXAPI::Resmgr::ResmgrAttrInit (resmgr_attr_t &* resmgr_attr*) [protected]

Initialize resource manager attributes from a 'C' attr structure.

Parameters:

resmgr_attr 'C' attributes structure.

void QNXAPI::Resmgr::InitializePeriodicScheduler (RepeatRate* rate*, int* prio = *0) [protected]

Initialize periodic scheduler.

Parameters:

rate frequency.
prio priority.

void QNXAPI::Resmgr::PrefixAttach (const std::string &* prefix*, Device &* device*, enum _file_type* type = *_FTYPE_ANY, unsigned* flags = *0) [protected]

Attach a prefix into the namespace.

Parameters:

prefix string representing prefix name.
device device to attach to the prefix.
type prefix type.
flags flags.

void QNXAPI::Resmgr::PrefixAttach (const std::string &* prefix*, enum _file_type* type = *_FTYPE_ANY, unsigned* flags = *0) [protected]

Attach a prefix into the namespace.

Parameters:

prefix string representing prefix name.
type prefix type.
flags flags.

void QNXAPI::Resmgr::Flags (unsigned* flags*)

Set the resource manager flags.

Parameters:

flags flags

void QNXAPI::Resmgr::Type (enum _file_type* type*)

Set the resource manager type.

Parameters:

type file type.

void QNXAPI::Resmgr::Attach (const std::string &* prefix*)

Attach the resource manager prefix.

Parameters:

prefix prefix to register.

void QNXAPI::Resmgr::Attach (const std::string &* prefix*, Device &* device*)

Attach the resource manager prefix.

Parameters:

prefix prefix to register.
device device to attach.

void QNXAPI::Resmgr::Attach (const std::string &* prefix*, Device &* device*, enum _file_type* type*)

Attach the resource manager prefix.

Parameters:

prefix prefix to register.
device device to attach.
type file type of registration.

void QNXAPI::Resmgr::Attach (const std::string &* prefix*, Device &* device*, enum _file_type* type*, unsigned* flags*)

Attach the resource manager prefix.

Parameters:

prefix prefix to register.
device device to attach.
type file type of registration.
flags flags

virtual struct ocb* QNXAPI::Resmgr::CreateOcb (void) [virtual]

Create an Open Control Block.

Returns:

pointer to ocb.

void QNXAPI::Resmgr::DestroyOcb (struct ocb *** ocb*)

Destroy an Open Control Block.

Parameters:

ocb pointer to ocb.

void QNXAPI::Resmgr::RegisterOcb (resmgr_context_t *** ctp*, io_open_t *** msg*, IOFUNC_ATTR_T *** attr*, IOFUNC_OCB_T *** ocb*, Ocb *** derived*) throw ()

Register an Open Control Block.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_close_t.
attr pointer to a QNX 'C' iofunc attributes structure.
ocb pointer to a QNX 'C' ocb.
derived ocb

int QNXAPI::Resmgr::DeregisterOcb (resmgr_context_t *** ctp*, IOFUNC_OCB_T *** ocb*) throw ()

De-register an Open Control Block.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
ocb pointer to a QNX 'C' ocb.

Returns:

small positive integer from errno.h.

Resmgr::Ocb* QNXAPI::Resmgr::operator[] (IOFUNC_OCB_T * ocb) [inline]

Index an Open Control Block.

Returns:

pointer to ocb.

Definition at line 2079 of file resmgr.int QNXAPI::Resmgr::DispatchInterruptHandler (message_context_t * mcp, int code, unsigned flags) throw ()

Dispatch an interrupt handler.

Parameters:

mcp pointer to a QNX 'C' library message_context_t.
code code
flags flags

Returns:

small positive integer from errno.h.

void QNXAPI::Resmgr::RegisterInterrupt (int intr, int pri, sigc::slot< const struct sigevent *, struct sigevent * > isr, sigc::slot< void > interrupt_handler, bool end = false)

Register an interrupt handler.

Parameters:

intr IRQ.
pri priority.
isr ISR.
interrupt_handler interrupt handler callback.
end add to end (true) or front (false) of handler list.

void QNXAPI::Resmgr::RegisterInterrupt (int intr, int pri, sigc::slot< void > interrupt_handler, bool end = false)

Register an interrupt handler.

Parameters:

intr IRQ.
pri priority.

interrupt_handler interrupt handler callback.
end add to end (true) or front (false) of handler list.

void QNXAPI::Resmgr::RegisterPeriodic (sigc::slot< int, message_context_t *, int, unsigned > periodic_task, RepeatRate rate = 1000)

Register a periodic handler.

Parameters:

periodic_task periodic handler callback.
rate frequency.

void QNXAPI::Resmgr::RegisterOpen (Device *, sigc::slot< int, resmgr_context_t *, io_open_t *, Device *, void * > open_handler)

Register a Open handler.

Parameters:

open_handler open handler callback.

void QNXAPI::Resmgr::RegisterMknod (Device *, sigc::slot< int, resmgr_context_t *, io_mknod_t *, Device *, void * > mknod_handler)

Register a Mknod handler.

Parameters:

mknod_handler mknod handler callback.

void QNXAPI::Resmgr::RegisterUnlink (Device *, sigc::slot< int, resmgr_context_t *, io_unlink_t *, Device *, void * > unlink_handler)

Register a Unlink handler.

Parameters:

unlink_handler unlink handler callback.

void QNXAPI::Resmgr::RegisterRename (Device *, sigc::slot< int, resmgr_context_t *, io_rename_t *, Device *, io_rename_extra_t * > rename_handler)

Register a Rename handler.

Parameters:

rename_handler rename handler callback.

void QNXAPI::Resmgr::RegisterMakeLink (Device *, sigc::slot< int, resmgr_context_t *, io_link_t *, Device *, io_link_extra_t * > makelink_handler)

Register a MakeLink handler.

Parameters:

makelink_handler makelink handler callback.

void QNXAPI::Resmgr::RegisterReadLink (Device *, sigc::slot< int, resmgr_context_t *, io_readlink_t *, Device *, void * > readlink_handler)

Register a ReadLink handler.

Parameters:

readlink_handler readlink handler callback.

void QNXAPI::Resmgr::RegisterCloseOcb (Device *, sigc::slot< int, resmgr_context_t *, void *, Ocb * > closeocb_handler)

Register a CloseOcb handler.

Parameters:

closeocb_handler closeocb handler callback.

void QNXAPI::Resmgr::RegisterRead (Device *, sigc::slot< int, resmgr_context_t *, io_read_t *, Ocb * > read_handler)

Register a Read handler.

Parameters:

read_handler read handler callback.

void QNXAPI::Resmgr::RegisterWrite (Device *, sigc::slot< int, resmgr_context_t *, io_write_t *, Ocb * > write_handler)

Register a Write handler.

Parameters:

write_handler write handler callback.

void QNXAPI::Resmgr::DetachAll (const std::string & prefix)

Detach all resources associated with prefix.

Parameters:

prefix name of prefix to detach.

void QNXAPI::Resmgr::DetachPathName (const std::string & prefix)

Detach prefix only.

Parameters:

prefix name of prefix to detach.

QNXAPI::IoSelector<resmgr_context_t>::BindingMap& QNXAPI::Resmgr::SelectorBindings (void) throw ()

Obtain a reference to an instance of the BindingMap attached to the selector.

Returns:

reference to an instance of a BindingMap

bool QNXAPI::Resmgr::AlwaysLoop (int err) const throw ()

Default callback (supplied to **Run()**) that will cause the resource manager loop to never cease unless an error occurs.

Client code can supply a different function to **Run()**.

Parameters:

err this is the error returned from the preceding resmgr_handler call.

Returns:

true if err is not an error, false otherwise.

int QNXAPI::Resmgr::Run (void)

Run the resource manager using the AlwaysLoop callback (above).

Returns:

small positive integer from errno.h.

int QNXAPI::Resmgr::Run (sigc::slot< bool, int > loopControl)

Run the resource manager using a client supplied loop callback (above).

Parameters:

loopControl client supplied loop control callback.

Returns:

small positive integer from errno.h.

void QNXAPI::Resmgr::ThreadPoolAttrInit (bool* autoConcurrency = *true, LowWater* lw = *1, Increment* inc = *0, HighWater* hw = *1, ThreadMax* max = *1) [protected]

Initialize thread pool attributes.

Parameters:

autoConcurrency have the resource manager select the optimal level of concurrency.
lw lower water for thread pool.
inc thread increment quantity.
hw high water for thread pool.
max maximum thread count for the pool.

void QNXAPI::Resmgr::ResmgrAttrInit (MsgPartsNum* nparts_max = *10, MsgSizeMax* msg_max_size = *2048, bool* endian = *true) [protected]

Initialize resource manager attributes.

Parameters:

nparts_max maximum number of message parts.
msg_max_size maximum size of a message.
endian support bi-endian messages.

void QNXAPI::Resmgr::ThreadPoolAttrInit (thread_pool_attr_t &* thread_pool_attr*) [protected]

Initialize thread pool attributes from a 'C' attr structute.

Parameters:

thread_pool_attr 'C' attributes structure.

void QNXAPI::Resmgr::ResmgrAttrInit (resmgr_attr_t &* resmgr_attr*) [protected]

Initialize resource manager attributes from a 'C' attr structure.

Parameters:

resmgr_attr 'C' attributes structure.

void QNXAPI::Resmgr::InitializePeriodicScheduler (RepeatRate* rate*, int* prio = *0) [protected]

Initialize periodic scheduler.

Parameters:

rate frequency.
prio priority.

void QNXAPI::Resmgr::PrefixAttach (const std::string &* **prefix**, *Device &* **device**, *enum _file_type* **type** **= *_FTYPE_ANY*, *unsigned* flags = *0*) [protected]

Attach a prefix into the namespace.

Parameters:

prefix string representing prefix name.
device device to attach to the prefix.
type prefix type.
flags flags.

void QNXAPI::Resmgr::PrefixAttach (const std::string &* **prefix**, *enum _file_type* **type** **= *_FTYPE_ANY*, *unsigned* flags = *0*) [protected]

Attach a prefix into the namespace.

Parameters:

prefix string representing prefix name.
type prefix type.
flags flags.

Friends And Related Function Documentation

int PeriodicRedir (message_context_t *** **mcp**, *int* **code**, *unsigned* **flags**, *void ** **handle**) throw () [friend]

Redirect a QNX 'C' library pulse_attach callback (attached to a periodic timer), into a **Resmgr** functor of the same purpose.

Parameters:

mcp pointer to a QNX 'C' library message_context_t.
code QNX pulse code.
flags flags.
handle user supplied pointer.

Returns:

small positive integer from errno.h.

int OpenRedir (resmgr_context_t *** **ctp**, *io_open_t ** **msg**, *RESMGR_HANDLE_T ** **handle**, *void ** **extra**) throw () [friend]

Redirect a QNX 'C' library iofunc_open callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_open_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.

extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int UnlinkRedir (resmgr_context_t * ctp, io_unlink_t * msg, RESMGR_HANDLE_T * handle, void * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_unlink callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_unlink_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int RenameRedir (resmgr_context_t * ctp, io_rename_t * msg, RESMGR_HANDLE_T * handle, io_rename_extra_t * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_rename callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_rename_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int MknodRedir (resmgr_context_t * ctp, io_mknod_t * msg, RESMGR_HANDLE_T * handle, void * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_mknod callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_mknod_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int ReadLinkRedir (resmgr_context_t * ctp, io_readlink_t * msg, RESMGR_HANDLE_T * handle, void * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_readlink callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_readlink_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int MakeLinkRedir (resmgr_context_t * ctp, io_link_t * msg, RESMGR_HANDLE_T * handle, io_link_extra_t * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_link callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_link_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int UnblockConRedir (resmgr_context_t * ctp, io_pulse_t * msg, RESMGR_HANDLE_T * handle, void * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_unblock callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_pulse_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int MountRedir (resmgr_context_t * ctp, io_mount_t * msg, RESMGR_HANDLE_T * handle, io_mount_extra_t * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_mount callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_mount_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int ReadRedir (resmgr_context_t * ctp, io_read_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_read callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_read_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int WriteRedir (resmgr_context_t * ctp, io_write_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_write callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_write_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int CloseRedir (resmgr_context_t * ctp, void * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_close callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg unused pointer.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int StatRedir (resmgr_context_t * ctp, io_stat_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_stat callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_stat_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int NotifyRedir (resmgr_context_t * ctp, io_notify_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_notify callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_notify_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int DevctlRedir (resmgr_context_t * ctp, io_devctl_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_devctl callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_devctl_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int UnblockRedir (resmgr_context_t * ctp, io_pulse_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_unblock callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_pulse_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int PathconfRedir (resmgr_context_t * ctp, io_pathconf_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_pathconf callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_pathconf_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int LseekRedir (resmgr_context_t * ctp, io_lseek_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_lseek callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_lseek_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int ChmodRedir (resmgr_context_t * ctp, io_chmod_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_chmod callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_chmod_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int ChownRedir (resmgr_context_t * ctp, io_chown_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_chown callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_chown_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int UtimeRedir (resmgr_context_t * ctp, io_utime_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_utime callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_utime_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int FdOpenRedir (resmgr_context_t * ctp, io_openfd_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_openfd callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_openfd_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int FdInfoRedir (resmgr_context_t * ctp, io_fdinfo_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_fdinfo callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_fdinfo_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int LockRedir (resmgr_context_t * ctp, io_lock_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_lock callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_lock_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int SpaceRedir (resmgr_context_t * ctp, io_space_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_space callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_space_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int ShutdownRedir (resmgr_context_t * ctp, io_shutdown_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_shutdown callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_shutdown_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int MmapRedir (resmgr_context_t * ctp, io_mmap_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_mmap callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_mmap_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int MsgRedir (resmgr_context_t * ctp, io_msg_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_msg callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_msg_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int UmountRedir (resmgr_context_t * ctp, void * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_umount callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg unused pointer.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int DupRedir (resmgr_context_t * ctp, io_dup_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_dup callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_dup_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int CloseDupRedir (resmgr_context_t * ctp, io_close_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_close_dup callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_close_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int LockOcbRedir (resmgr_context_t * ctp, void * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_lock_ocb callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg unused pointer.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int UnlockOcbRedir (resmgr_context_t * ctp, void * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_unlock_ocb callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg unused pointer.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int PeriodicRedir (message_context_t * mcp, int code, unsigned flags, void * handle) throw () [friend]

Redirect a QNX 'C' library pulse_attach callback (attached to a periodic timer), into a **Resmgr** functor of the same purpose.

Parameters:

mcp pointer to a QNX 'C' library message_context_t.
code QNX pulse code.
flags flags.
handle user supplied pointer.

Returns:

small positive integer from errno.h.

int OpenRedir (resmgr_context_t * ctp, io_open_t * msg, RESMGR_HANDLE_T * handle, void * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_open callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_open_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int UnlinkRedir (resmgr_context_t * ctp, io_unlink_t * msg, RESMGR_HANDLE_T * handle, void * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_unlink callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_unlink_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int RenameRedir (resmgr_context_t * ctp, io_rename_t * msg, RESMGR_HANDLE_T * handle, io_rename_extra_t * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_rename callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.

msg pointer to a QNX 'C' library io_rename_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int MknodRedir (resmgr_context_t * ctp, io_mknod_t * msg, RESMGR_HANDLE_T * handle, void * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_mknod callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_mknod_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int ReadLinkRedir (resmgr_context_t * ctp, io_readlink_t * msg, RESMGR_HANDLE_T * handle, void * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_readlink callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_readlink_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int MakeLinkRedir (resmgr_context_t * ctp, io_link_t * msg, RESMGR_HANDLE_T * handle, io_link_extra_t * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_link callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_link_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int UnblockConRedir (resmgr_context_t * ctp, io_pulse_t * msg, RESMGR_HANDLE_T * handle, void * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_unblock callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_pulse_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int MountRedir (resmgr_context_t * ctp, io_mount_t * msg, RESMGR_HANDLE_T * handle, io_mount_extra_t * extra) throw () [friend]

Redirect a QNX 'C' library iofunc_mount callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_mount_t.
handle pointer to a QNX 'C' library RESMGR_HANDLE_T.
extra pointer to operation specific data.

Returns:

small positive integer from errno.h.

int ReadRedir (resmgr_context_t * ctp, io_read_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_read callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_read_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int WriteRedir (resmgr_context_t * ctp, io_write_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_write callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_write_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int CloseRedir (resmgr_context_t * ctp, void * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_close callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg unused pointer.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int StatRedir (resmgr_context_t * ctp, io_stat_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_stat callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_stat_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int NotifyRedir (resmgr_context_t * ctp, io_notify_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_notify callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_notify_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int DevctlRedir (resmgr_context_t * ctp, io_devctl_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_devctl callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_devctl_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int UnblockRedir (resmgr_context_t * ctp, io_pulse_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_unblock callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_pulse_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int PathconfRedir (resmgr_context_t * ctp, io_pathconf_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_pathconf callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_pathconf_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int LseekRedir (resmgr_context_t * ctp, io_lseek_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_lseek callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_lseek_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int ChmodRedir (resmgr_context_t * ctp, io_chmod_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_chmod callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_chmod_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int ChownRedir (resmgr_context_t * ctp, io_chown_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_chown callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_chown_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int UtimeRedir (resmgr_context_t * ctp, io_utime_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_utime callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_utime_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int FdOpenRedir (resmgr_context_t * ctp, io_openfd_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_openfd callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_openfd_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int FdInfoRedir (resmgr_context_t * ctp, io_fdinfo_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_fdinfo callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_fdinfo_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int LockRedir (resmgr_context_t * ctp, io_lock_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_lock callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_lock_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int SpaceRedir (resmgr_context_t * ctp, io_space_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_space callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_space_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int ShutdownRedir (resmgr_context_t * ctp, io_shutdown_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_shutdown callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_shutdown_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int MmapRedir (resmgr_context_t * ctp, io_mmap_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_mmap callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_mmap_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int MsgRedir (resmgr_context_t * ctp, io_msg_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_msg callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_msg_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int UmountRedir (resmgr_context_t * ctp, void * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_umount callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg unused pointer.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int DupRedir (resmgr_context_t * ctp, io_dup_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_dup callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_dup_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:

small positive integer from errno.h.

int CloseDupRedir (resmgr_context_t * ctp, io_close_t * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_close_dup callback into a **Resmgr** functor of the same purpose.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_close_t.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:
small positive integer from errno.h.

int LockOcbRedir (resmgr_context_t * ctp, void * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_lock_ocb callback into a **Resmgr** functor of the same purpose.

Parameters:
ctp pointer to a QNX 'C' library resmgr_context_t.
msg unused pointer.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:
small positive integer from errno.h.

int UnlockOcbRedir (resmgr_context_t * ctp, void * msg, RESMGR_OCB_T * ocb) throw () [friend]

Redirect a QNX 'C' library iofunc_unlock_ocb callback into a **Resmgr** functor of the same purpose.

Parameters:
ctp pointer to a QNX 'C' library resmgr_context_t.
msg unused pointer.
ocb pointer to a QNX 'C' library RESMGR_OCB_T.

Returns:
small positive integer from errno.h.

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

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

QNXAPI::Resmgr::AttrLock Class Reference

Attribute Lock class.

Public Member Functions

- **AttrLock** (IOFUNC_ATTR_T ***device**)
  Build an attribute lock, upon successful construction, attribute is locked.

- **~AttrLock** (void)
  Destroy and attribute lock, upon destruction attribute is unlocked.

- **AttrLock** (IOFUNC_ATTR_T ***device**)
  Build an attribute lock, upon successful construction, attribute is locked.

- **~AttrLock** (void)
  Destroy and attribute lock, upon destruction attribute is unlocked.

Detailed Description

Attribute Lock class.

This class provides attribute locking functionality. Attributes are locked on construction and unlocked at destruction.

Definition at line 420 of file resmgr.svn-base.

Constructor & Destructor Documentation

QNXAPI::Resmgr::AttrLock::AttrLock (IOFUNC_ATTR_T * device) [inline]

Build an attribute lock, upon successful construction, attribute is locked.

Parameters:

device attribute structure to lock.

Definition at line 433 of file resmgr.svn-base.QNXAPI::Resmgr::AttrLock::AttrLock (IOFUNC_ATTR_T * device) [inline]

Build an attribute lock, upon successful construction, attribute is locked.

Parameters:

device attribute structure to lock.

Definition at line 433 of file resmgr.

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

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

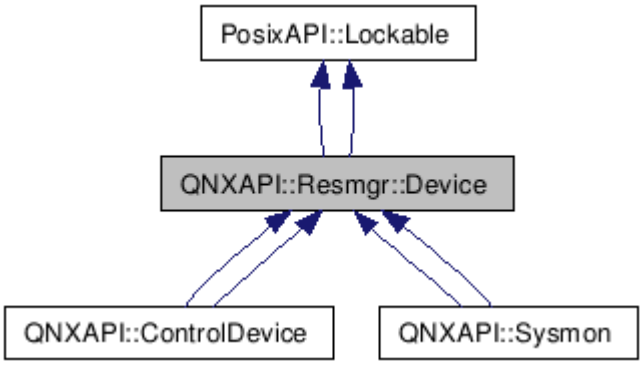
QNXAPI::Resmgr::Device Class Reference

Resource Manager **Device** class.

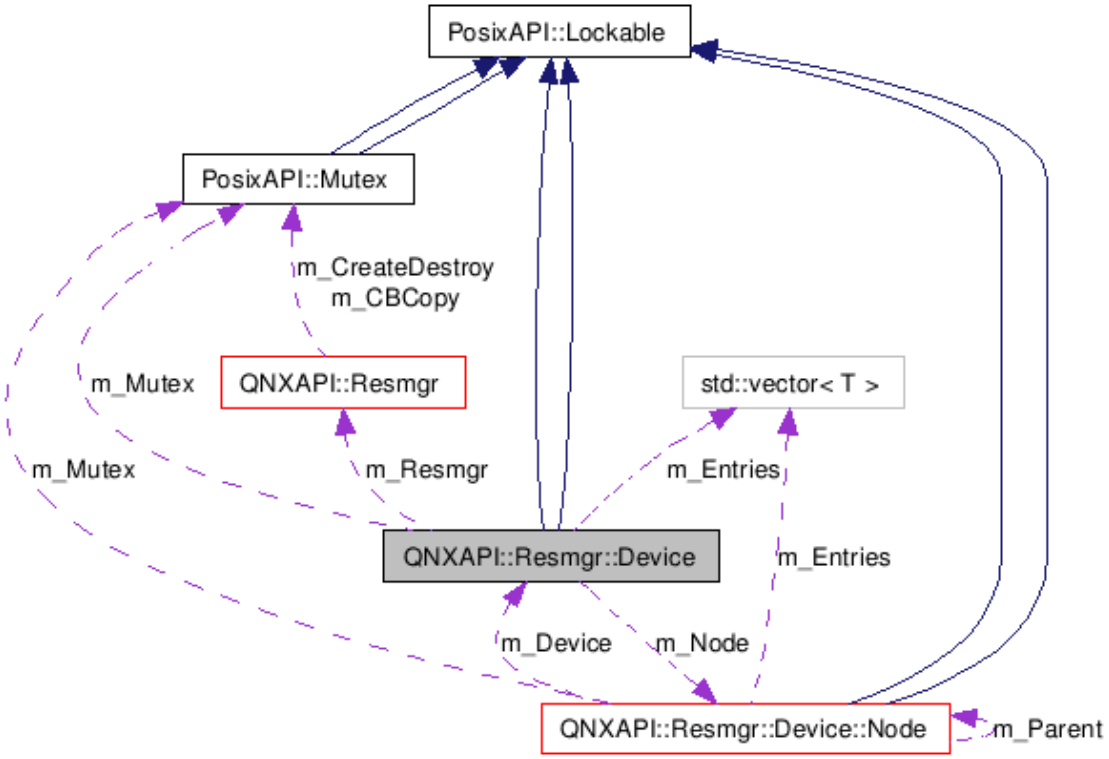
Inherits **PosixAPI::Lockable**, and **PosixAPI::Lockable**.

Inherited by **QNXAPI::ControlDevice**, **QNXAPI::ControlDevice**, **QNXAPI::Sysmon**, and **QNXAPI::Sysmon**.

Inheritance diagram for QNXAPI::Resmgr::Device:



Collaboration diagram for QNXAPI::Resmgr::Device:



Public Types

- typedef std::vector< **QNXAPI::Resmgr::Device::Link** * > **LinkEntries**
  *Typedef for **Link** Entries.*

- typedef std::vector< **QNXAPI::Resmgr::Device::Link** * > **LinkEntries**
  *Typedef for **Link** Entries.*

Public Member Functions

- **Device** (**Resmgr** &resmgr, mode_t mode=(S_IFNAM|0755), int ioflag=0)
  Construct and plug in device.

- virtual **~Device** ()
  Destroy a device.

- void **RegisterWithResmgr** (void)
  Register this device with its resource manager.

- IOFUNC_ATTR_T * **Derived** (void)
  Return a pointer to the internal derived iofunc_attr_t structure.

- virtual **Node** & **RootNode** (void)
  Return devices node.

- mode_t **Mode** (void)
  Return devices mode.

- void **Mode** (mode_t mode)
  Set the devices mode.

- mode_t **Umask** (void) const

Return the devices umask (default mode for new creations).

- void **Umask** (mode_t mask)
  Set the devices umask.

- void **Lock** (**PosixAPI::Lockable::Mode** mode=PosixAPI::Lockable::Exclusive) const
  Lock the device. If called with no parameters, then an exclusive lock is obtained.

- void **TryLock** (**PosixAPI::Lockable::Mode** mode) const
  Attempt to obtain a lock on the device. If called with no parameters, then an exclusive lock is attempted.

- void **Unlock** (void) const
  Unlock the device.

- **Device** (**Resmgr** &resmgr, mode_t mode=(S_IFNAM|0755), int ioflag=0)
  Construct and plug in device.

- virtual **~Device** ()
  Destroy a device.

- void **RegisterWithResmgr** (void)
  Register this device with its resource manager.

- IOFUNC_ATTR_T * **Derived** (void)
  Return a pointer to the internal derived iofunc_attr_t structure.

- virtual **Node** & **RootNode** (void)
  Return devices node.

- mode_t **Mode** (void)
  Return devices mode.

- void **Mode** (mode_t mode)
  Set the devices mode.

- mode_t **Umask** (void) const
  Return the devices umask (default mode for new creations).

- void **Umask** (mode_t mask)
  Set the devices umask.

- void **Lock** (**PosixAPI::Lockable::Mode** mode=PosixAPI::Lockable::Exclusive) const
  Lock the device. If called with no parameters, then an exclusive lock is obtained.

- void **TryLock** (**PosixAPI::Lockable::Mode** mode) const
  Attempt to obtain a lock on the device. If called with no parameters, then an exclusive lock is attempted.

- void **Unlock** (void) const
  Unlock the device.

Protected Member Functions

- virtual int **CheckAccess** (resmgr_context_t *ctp, const std::string &path, unsigned access, struct _client_info &info, IOFUNC_ATTR_T **dattr) throw (SymLinkException)
  Check access permissions.

- virtual bool **TargetExists** (const std::string &path) throw ()
  This method verifies that the target specified by path exists.

- virtual **Node** & **CreateNode** (const std::string &dir, const std::string &target) throw (NodeException)
  Create a new node (and initial link), specifying parent directory, and initial link name.

- virtual **Node** & **CreateNode** (const std::string &dir, const std::string &target, void *data, IOFUNC_ATTR_T *attr, **LinkEntries** &entries) throw (NodeException)

Create a new node (and initial link), specifying parent directory, initial link name, data, attributes, and entries.

- virtual **Node** & **CreateNode** (const std::string &dir, const std::string &target, resmgr_context_t *ctp, dev_t type, mode_t mode, struct _client_info &info, IOFUNC_ATTR_T *dattr) throw (NodeException)
  Create a new node (and initial link), specifying parent directory, initial link name, node type, mode, client credentials, and parent directory attributes.

- virtual int **DestroyNode** (const std::string &dir, const std::string &target) throw ()
  Destroy node. Delete the node. All memory used by node is recovered.

- virtual int **RemoveLink** (const std::string &dir, const std::string &target) throw ()
  *Remove link. **Link** named by target is removed from directory named by dir. Linked node is not affected.*

- virtual **Link** & **FindLink** (const std::string &path) throw (NodeException)
  Retrieve the link named by path.

- virtual void **AddLink** (**Link** *link)
  Add an entry to the devices node.

- virtual bool **CreateTest** (uint32_t mode) const
  Determine if mode has create flags set.

- virtual void **AddLink** (const std::string &directory, **Link** *link)
  Add the link specified by link, to the directory named by directory,.

- virtual **Ocb** & **CreateOcb** (**Resmgr** &resmgr, **Device** &**device**, **Node** &node, struct _client_info &info, resmgr_context_t *ctp, io_open_t *msg) throw (OcbException)
  Create an open control block.

- virtual int **Open** (resmgr_context_t *ctp, io_open_t *msg, void *extra) throw ()
  Open a resource.

- virtual int **Unlink** (resmgr_context_t *ctp, io_unlink_t *msg, void *extra) throw ()
  Unlink (remove) a resource.

- virtual int **Rename** (resmgr_context_t *ctp, io_rename_t *msg, io_rename_extra_t *extra) throw ()
  Rename a resource.

- virtual int **Mknod** (resmgr_context_t *ctp, io_mknod_t *msg, void *extra) throw ()
  Mknod make a node.

- virtual int **ReadLink** (resmgr_context_t *ctp, io_readlink_t *msg, void *extra) throw ()
  Read the contents of a link resource.

- virtual int **MakeLink** (resmgr_context_t *ctp, io_link_t *msg, io_link_extra_t *extra) throw ()
  Create a link resource.

- virtual int **HardLink** (resmgr_context_t *ctp, io_link_t *msg, **Ocb** ***ocb**) throw ()
  Create a hard link.

- virtual int **SymLink** (resmgr_context_t *ctp, io_link_t *msg, const std::string &target) throw ()
  Create a symbolic link.

- virtual int **ReadDir** (resmgr_context_t *ctp, io_read_t *msg, **Ocb** ***ocb**) throw ()
  Perform a readdir() operation on OCB.

- virtual int **Read** (resmgr_context_t *ctp, io_read_t *msg, **Ocb** ***ocb**) throw ()
  Perform a read() operation on OCB.

- virtual int **Write** (resmgr_context_t *ctp, io_write_t *msg, **Ocb** ***ocb**) throw ()
  Perform a write() operation on OCB.

- virtual int **CloseOcb** (resmgr_context_t *ctp, void *msg, **Ocb** ***ocb**) throw ()

Close an Open Control Block.

- int **BounceLink** (resmgr_context_t *ctp, io_open_t *msg, const char *linkname)
  Bounce a link (perform a redirection).

- int **BounceLink** (resmgr_context_t *ctp, io_open_t *msg, const char *link, const char *target, const char *remainder)
- virtual int **CheckAccess** (resmgr_context_t *ctp, const std::string &path, unsigned access, struct _client_info &info, IOFUNC_ATTR_T **dattr) throw (SymLinkException)
  Check access permissions.

- virtual bool **TargetExists** (const std::string &path) throw ()
  This method verifies that the target specified by path exists.

- virtual **Node** & **CreateNode** (const std::string &dir, const std::string &target) throw (NodeException)
  Create a new node (and initial link), specifying parent directory, and initial link name.

- virtual **Node** & **CreateNode** (const std::string &dir, const std::string &target, void *data, IOFUNC_ATTR_T *attr, **LinkEntries** &entries) throw (NodeException)
  Create a new node (and initial link), specifying parent directory, initial link name, data, attributes, and entries.

- virtual **Node** & **CreateNode** (const std::string &dir, const std::string &target, resmgr_context_t *ctp, dev_t type, mode_t mode, struct _client_info &info, IOFUNC_ATTR_T *dattr) throw (NodeException)
  Create a new node (and initial link), specifying parent directory, initial link name, node type, mode, client credentials, and parent directory attributes.

- virtual int **DestroyNode** (const std::string &dir, const std::string &target) throw ()
  Destroy node. Delete the node. All memory used by node is recovered.

- virtual int **RemoveLink** (const std::string &dir, const std::string &target) throw ()
  *Remove link. **Link** named by target is removed from directory named by dir. Linked node is not affected.*

- virtual **Link** & **FindLink** (const std::string &path) throw (NodeException)

Retrieve the link named by path.

- virtual void **AddLink** (**Link** *link)
  Add an entry to the devices node.

- virtual bool **CreateTest** (uint32_t mode) const
  Determine if mode has create flags set.

- virtual void **AddLink** (const std::string &directory, **Link** *link)
  Add the link specified by link, to the directory named by directory,.

- virtual **Ocb** & **CreateOcb** (**Resmgr** &resmgr, **Device** &**device**, **Node** &node, struct _client_info &info, resmgr_context_t *ctp, io_open_t *msg) throw (OcbException)
  Create an open control block.

- virtual int **Open** (resmgr_context_t *ctp, io_open_t *msg, void *extra) throw ()
  Open a resource.

- virtual int **Unlink** (resmgr_context_t *ctp, io_unlink_t *msg, void *extra) throw ()
  Unlink (remove) a resource.

- virtual int **Rename** (resmgr_context_t *ctp, io_rename_t *msg, io_rename_extra_t *extra) throw ()
  Rename a resource.

- virtual int **Mknod** (resmgr_context_t *ctp, io_mknod_t *msg, void *extra) throw ()
  Mknod make a node.

- virtual int **ReadLink** (resmgr_context_t *ctp, io_readlink_t *msg, void *extra) throw ()
  Read the contents of a link resource.

- virtual int **MakeLink** (resmgr_context_t *ctp, io_link_t *msg, io_link_extra_t *extra) throw ()
  Create a link resource.

- virtual int **HardLink** (resmgr_context_t *ctp, io_link_t *msg, **Ocb** ***ocb**) throw ()
  Create a hard link.

- virtual int **SymLink** (resmgr_context_t *ctp, io_link_t *msg, const std::string &target) throw ()
  Create a symbolic link.

- virtual int **ReadDir** (resmgr_context_t *ctp, io_read_t *msg, **Ocb** ***ocb**) throw ()
  Perform a readdir() operation on OCB.

- virtual int **Read** (resmgr_context_t *ctp, io_read_t *msg, **Ocb** ***ocb**) throw ()
  Perform a read() operation on OCB.

- virtual int **Write** (resmgr_context_t *ctp, io_write_t *msg, **Ocb** ***ocb**) throw ()
  Perform a write() operation on OCB.

- virtual int **CloseOcb** (resmgr_context_t *ctp, void *msg, **Ocb** ***ocb**) throw ()
  Close an Open Control Block.

- int **BounceLink** (resmgr_context_t *ctp, io_open_t *msg, const char *linkname)
  Bounce a link (perform a redirection).

- int **BounceLink** (resmgr_context_t *ctp, io_open_t *msg, const char *link, const char *target, const char *remainder)

Classes

- class **Link**
  *Resource Manager **Device Link**.*

- class **Node**
  *Resource Manager **Device Node**.*

Detailed Description

Resource Manager **Device** class.

This class provides a framework for "Devices". In **Resmgr** terminology a **Device** is the object attached to a mountpoint. The mountpoint may be an attachment point (via **Resmgr::Attach**) or it may be a synthetic mountpoint created by another **Device**.

Examples:

Ramdisk/main.cpp.

Definition at line 455 of file resmgr.svn-base.

Constructor & Destructor Documentation

QNXAPI::Resmgr::Device::Device (Resmgr &* resmgr*, mode_t* mode = *(S_IFNAM|0755), int* ioflag = *0)

Construct and plug in device.

Parameters:

resmgr instance of resmgr to plug this device into.
mode devices mode.
ioflag

QNXAPI::Resmgr::Device::Device (Resmgr &* resmgr*, mode_t* mode = *(S_IFNAM|0755), int* ioflag = *0)

Construct and plug in device.

Parameters:

resmgr instance of resmgr to plug this device into.
mode devices mode.
ioflag

Member Function Documentation

virtual Node& QNXAPI::Resmgr::Device::RootNode (void) [virtual]

Return devices node.

Returns:

the root **Node** for the device.

Reimplemented in *QNXAPI::ControlDevice* **(p.55)**, ***and*** *QNXAPI::ControlDevice* **(p.55)**.***mode_t QNXAPI::Resmgr::Device::Mode (void)***

Return devices mode.

Returns:

a mode_t for the device.

void QNXAPI::Resmgr::Device::Mode (mode_t **mode)**

Set the devices mode.

Parameters:

mode new mode for device.

mode_t QNXAPI::Resmgr::Device::Umask (void) const

Return the devices umask (default mode for new creations).

Returns:

a mode_t mask.

void QNXAPI::Resmgr::Device::Umask (mode_t **mask)**

Set the devices umask.

Parameters:

mask mode mask for new creations.

void QNXAPI::Resmgr::Device::Lock (PosixAPI::Lockable::Mode **mode =** ***PosixAPI::Lockable::Exclusive) const [virtual]***

Lock the device. If called with no parameters, then an exclusive lock is obtained.

Parameters:

mode boolean; true if caller wants exclusive lock false otherwise

Returns:

small positive integer from errno.h (see pthread*trylock docs).

Implements *PosixAPI::Lockable* **(p.98)*.void QNXAPI::Resmgr::Device::TryLock (PosixAPI::Lockable::Mode mode) const [virtual]***

Attempt to obtain a lock on the device. If called with no parameters, then an exclusive lock is attempted.

Parameters:

mode boolean; true if caller wants exclusive lock false otherwise

Returns:

small positive integer from errno.h (see pthread*trylock docs).

Implements *PosixAPI::Lockable* **(p.98)*.virtual int QNXAPI::Resmgr::Device::CheckAccess (resmgr_context_t * ctp, const std::string & path, unsigned access, struct _client_info & info, IOFUNC_ATTR_T ** dattr) throw (SymLinkException) [protected, virtual]***

Check access permissions.

Verify that the currently registered client info provides sufficient credentials to access *all* of the resources described by the path. As a convenience, the attributes for the component of the path immediately ahead of the terminal component are returned in dattr. This is useful when the terminal component is a resource that is being created. In this case, dattr can be consulted to insure that the client has create permission. This function may be overridden to provide different semantics.

Parameters:

ctp resource manager context pointer.
path cstring containing path to be checked.
access access mode.
info client info structure.
dattr attribute structure of the second last component of path (out).

Returns:

error code from errno.h

virtual bool QNXAPI::Resmgr::Device::TargetExists (const std::string & path) throw () [protected, virtual]

This method verifies that the target specified by path exists.

Parameters:

path path of target to be verified.

Returns:

bool indicating whether target exists (true) or not (false)

virtual Node& QNXAPI::Resmgr::Device::CreateNode (const std::string & dir, const std::string & target) throw (NodeException) [protected, virtual]

Create a new node (and initial link), specifying parent directory, and initial link name.

Parameters:

dir parent node.

target name of initial link to node.

Returns:

reference to new node (link will have been placed in dir)

***virtual* Node& QNXAPI::Resmgr::Device::CreateNode (*const* std::string & dir, *const* std::string & target, *void* * data, *IOFUNC_ATTR_T* * attr, *LinkEntries* & entries) *throw* (NodeException) [protected, virtual]**

Create a new node (and initial link), specifying parent directory, initial link name, data, attributes, and entries.

Parameters:

dir parent node.
target name of node to create.
data pointer to allocated data for node.
attr pointer to attributes to copy.
entries entries to copy.

Returns:

reference to new node (link will have been placed in dir)

***virtual* Node& QNXAPI::Resmgr::Device::CreateNode (*const* std::string & dir, *const* std::string & target, *resmgr_context_t* * ctp, *dev_t* type, *mode_t* mode, *struct _client_info* & info, *IOFUNC_ATTR_T* * dattr) *throw* (NodeException) [protected, virtual]**

Create a new node (and initial link), specifying parent directory, initial link name, node type, mode, client credentials, and parent directory attributes.

Parameters:

dir parent node.
target name of node to create.
ctp 'C' resmgr library context pointer.
type node (file) type.
mode node mode.
info client credentials.
dattr pointer to parent directory attributes.

Returns:

reference to new node (link will have been placed in dir)

***virtual int* QNXAPI::Resmgr::Device::DestroyNode (*const* std::string & dir, *const* std::string & target) *throw* () [protected, virtual]**

Destroy node. Delete the node. All memory used by node is recovered.

Parameters:

dir parent node.
target name of node to destroy.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::RemoveLink (const std::string & dir, const std::string & target) throw () [protected, virtual]

Remove link. **Link** named by target is removed from directory named by dir. Linked node is not affected.

Parameters:

dir parent node.
target name of node to destroy.

Returns:

small positive integer from errno.h

virtual Link& QNXAPI::Resmgr::Device::FindLink (const std::string & path) throw (NodeException) [protected, virtual]

Retrieve the link named by path.

Parameters:

path complete path of target.

Returns:

reference to located link.

virtual void QNXAPI::Resmgr::Device::AddLink (Link * link) [inline, protected, virtual]

Add an entry to the devices node.

Parameters:

link pointer to link.

Definition at line 1025 of file resmgr.svn-base.

References QNXAPI::Resmgr::Device::Node::AddLink().

Here is the call graph for this function:

virtual bool QNXAPI::Resmgr::Device::CreateTest (uint32_t mode) const [inline, protected, virtual]

Determine if mode has create flags set.

Parameters:

mode mode to test.

Definition at line 1035 of file resmgr.svn-base.virtual void QNXAPI::Resmgr::Device::AddLink (const std::string & directory, Link * link) [protected, virtual]

Add the link specified by link, to the directory named by directory,.

Parameters:

directory parent directory for link.
link link to add.

virtual Ocb& QNXAPI::Resmgr::Device::CreateOcb (Resmgr & resmgr, Device & device, Node & node, struct _client_info & info, resmgr_context_t * ctp, io_open_t * msg) throw (OcbException) [protected, virtual]

Create an open control block.

Parameters:

resmgr reference to the owning resource manager.
device reference to the managing device.
node reference to the managing node.
info credentials of client this OCB represents.
ctp 'C' resmgr library context pointer.
msg 'C' resmgr library open message.

Returns:

reference to created OCB.

virtual int QNXAPI::Resmgr::Device::Open (resmgr_context_t * ctp, io_open_t * msg, void * extra) throw () [protected, virtual]

Open a resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library open message.
extra 'C' resmgr library "extra" information for open.

Returns:

small positive integer from errno.h

Reimplemented in QNXAPI::Sysmon (p.338), and QNXAPI::Sysmon (p.338).virtual int QNXAPI::Resmgr::Device::Unlink (resmgr_context_t * ctp, io_unlink_t * msg, void * extra) throw () [protected, virtual]

Unlink (remove) a resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library unlink message.
extra 'C' resmgr library "extra" information for unlink.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::Rename (resmgr_context_t * ctp, io_rename_t * msg, io_rename_extra_t * extra) throw () [protected, virtual]

Rename a resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library rename message.
extra 'C' resmgr library "extra" information for rename (new path).

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::Mknod (resmgr_context_t * ctp, io_mknod_t * msg, void * extra) throw () [protected, virtual]

Mknod make a node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library mknod message.
extra 'C' resmgr library "extra" information for mknod.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::ReadLink (resmgr_context_t * ctp, io_readlink_t * msg, void * extra) throw () [protected, virtual]

Read the contents of a link resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library readlink message.
extra 'C' resmgr library "extra" information for a readlink.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::MakeLink (resmgr_context_t * ctp, io_link_t * msg, io_link_extra_t * extra) throw () [protected, virtual]

Create a link resource.

Parameters:

ctp 'C' resmgr library context pointer.

msg 'C' resmgr library link message.
extra 'C' resmgr library "extra" information for link.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::HardLink (resmgr_context_t * ctp, io_link_t * msg, Ocb * ocb) throw () [protected, virtual]

Create a hard link.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library link message.
ocb open control block on target node.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::SymLink (resmgr_context_t * ctp, io_link_t * msg, const std::string & target) throw () [protected, virtual]

Create a symbolic link.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library link message.
target contents of symlink.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::ReadDir (resmgr_context_t * ctp, io_read_t * msg, Ocb * ocb) throw () [protected, virtual]

Perform a readdir() operation on OCB.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library readdir message.
ocb Open Control Block of directory node.

Returns:

number of bytes returned.

virtual int QNXAPI::Resmgr::Device::Read (resmgr_context_t * ctp, io_read_t * msg, Ocb * ocb) throw () [protected, virtual]

Perform a read() operation on OCB.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb Open Control Block of non-directory node.

Returns:

number of bytes returned.

virtual int QNXAPI::Resmgr::Device::Write (resmgr_context_t * ctp, io_write_t * msg, Ocb * ocb) throw () [protected, virtual]

Perform a write() operation on OCB.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb Open Control Block of non-directory node.

Returns:

number of bytes returned.

virtual int QNXAPI::Resmgr::Device::CloseOcb (resmgr_context_t * ctp, void * msg, Ocb * ocb) throw () [protected, virtual]

Close an Open Control Block.

Parameters:

ctp 'C' resmgr library context pointer.
msg void pointer (not used).
ocb Open Control Block to close.

Returns:

small positive integer from errno.h

Reimplemented in QNXAPI::Sysmon (p.338), and QNXAPI::Sysmon (p.338).int QNXAPI::Resmgr::Device::BounceLink (resmgr_context_t * ctp, io_open_t * msg, const char * linkname) [protected]

Bounce a link (perform a redirection).

Bounce a connection operation on a symbolic link. This is intended to be called from connection callbacks when the node opened IsSymLink(). While this call does return, connection callbacks that make use of it should simply return the return from this function to their caller.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library io_open message pointer.
linkname symbolic link name.

Returns:

small positive integer from errno.h

int QNXAPI::Resmgr::Device::BounceLink (resmgr_context_t * ctp, io_open_t * msg, const char * link, const char * target, const char * remainder) [protected]

Bounce a connection operation on a symbolic link. This form of the call is designed to expand intermediate links (i.e. directories).

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library io_open message pointer.
link intermediate symbolic link component to be replaced.
target target path to substitute for link.
remainder remainder of path following substitution.

Returns:

small positive integer from errno.h

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

virtual Node& QNXAPI::Resmgr::Device::RootNode (void) [virtual]

Return devices node.

Returns:

the root **Node** for the device.

Reimplemented in *QNXAPI::ControlDevice* ***(p.55), and*** *QNXAPI::ControlDevice* ***(p.55).mode_t QNXAPI::Resmgr::Device::Mode (void)***

Return devices mode.

Returns:

a mode_t for the device.

void QNXAPI::Resmgr::Device::Mode (mode_t mode)

Set the devices mode.

Parameters:

mode new mode for device.

mode_t QNXAPI::Resmgr::Device::Umask (void) const

Return the devices umask (default mode for new creations).

Returns:

a mode_t mask.

void QNXAPI::Resmgr::Device::Umask (mode_t* mask*)

Set the devices umask.

Parameters:

mask mode mask for new creations.

void QNXAPI::Resmgr::Device::Lock (PosixAPI::Lockable::Mode* mode = *PosixAPI::Lockable::Exclusive) const [virtual]

Lock the device. If called with no parameters, then an exclusive lock is obtained.

Parameters:

mode boolean; true if caller wants exclusive lock false otherwise

Returns:

small positive integer from errno.h (see pthread*trylock docs).

Implements** PosixAPI::Lockable (*p.98*).void QNXAPI::Resmgr::Device::TryLock (PosixAPI::Lockable::Mode* mode*) const [virtual]

Attempt to obtain a lock on the device. If called with no parameters, then an exclusive lock is attempted.

Parameters:

mode boolean; true if caller wants exclusive lock false otherwise

Returns:

small positive integer from errno.h (see pthread*trylock docs).

Implements** PosixAPI::Lockable (*p.98*).virtual int QNXAPI::Resmgr::Device::CheckAccess (resmgr_context_t * ctp, const std::string & path, unsigned access, struct _client_info & info, IOFUNC_ATTR_T ** dattr) throw (SymLinkException) [protected, virtual]

Check access permissions.

Verify that the currently registered client info provides sufficient credentials to access *all* of the resources described by the path. As a convenience, the attributes for the component of the path immediately ahead of the terminal component are returned in dattr. This is useful when the terminal component is a resource that is being created. In this case, dattr can be consulted to insure that the client has create permission. This function may be overridden to provide different semantics.

Parameters:

ctp resource manager context pointer.
path cstring containing path to be checked.
access access mode.
info client info structure.
dattr attribute structure of the second last component of path (out).

Returns:

error code from errno.h

virtual bool QNXAPI::Resmgr::Device::TargetExists (const std::string &* **path) throw () [protected, virtual]***

This method verifies that the target specified by path exists.

Parameters:

path path of target to be verified.

Returns:

bool indicating whether target exists (true) or not (false)

virtual Node& QNXAPI::Resmgr::Device::CreateNode (const std::string &* **dir, const std::string &* **target***) throw (NodeException) [protected, virtual]***

Create a new node (and initial link), specifying parent directory, and initial link name.

Parameters:

dir parent node.
target name of initial link to node.

Returns:

reference to new node (link will have been placed in dir)

virtual Node& QNXAPI::Resmgr::Device::CreateNode (const std::string &* **dir, const std::string &* **target***, void **** **data***, IOFUNC_ATTR_T **** **attr***, LinkEntries &* **entries***) throw (NodeException) [protected, virtual]***

Create a new node (and initial link), specifying parent directory, initial link name, data, attributes, and entries.

Parameters:

dir parent node.
target name of node to create.
data pointer to allocated data for node.
attr pointer to attributes to copy.
entries entries to copy.

Returns:

reference to new node (link will have been placed in dir)

virtual Node& QNXAPI::Resmgr::Device::CreateNode (const std::string &* **dir, const std::string &* **target***, resmgr_context_t **** **ctp***, dev_t* **type***, mode_t* **mode***, struct _client_info &* **info***, IOFUNC_ATTR_T **** **dattr***) throw (NodeException) [protected, virtual]***

Create a new node (and initial link), specifying parent directory, initial link name, node type, mode, client credentials, and parent directory attributes.

Parameters:

dir parent node.
target name of node to create.

ctp 'C' resmgr library context pointer.
type node (file) type.
mode node mode.
info client credentials.
dattr pointer to parent directory attributes.

Returns:

reference to new node (link will have been placed in dir)

virtual int QNXAPI::Resmgr::Device::DestroyNode (const std::string & dir, const std::string & target) throw () [protected, virtual]

Destroy node. Delete the node. All memory used by node is recovered.

Parameters:

dir parent node.
target name of node to destroy.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::RemoveLink (const std::string & dir, const std::string & target) throw () [protected, virtual]

Remove link. **Link** named by target is removed from directory named by dir. Linked node is not affected.

Parameters:

dir parent node.
target name of node to destroy.

Returns:

small positive integer from errno.h

virtual Link& QNXAPI::Resmgr::Device::FindLink (const std::string & path) throw (NodeException) [protected, virtual]

Retrieve the link named by path.

Parameters:

path complete path of target.

Returns:

reference to located link.

virtual void QNXAPI::Resmgr::Device::AddLink (Link * link) [inline, protected, virtual]

Add an entry to the devices node.

Parameters:

link pointer to link.

Definition at line 1025 of file resmgr.

References QNXAPI::Resmgr::Device::Node::AddLink().

Here is the call graph for this function:

virtual bool QNXAPI::Resmgr::Device::CreateTest (uint32_t* mode*) const [inline, protected, virtual]

Determine if mode has create flags set.

Parameters:

mode mode to test.

Definition at line 1035 of file resmgr.virtual void QNXAPI::Resmgr::Device::AddLink (const std::string & directory, Link * link) [protected, virtual]

Add the link specified by link, to the directory named by directory,.

Parameters:

directory parent directory for link.
link link to add.

virtual Ocb& QNXAPI::Resmgr::Device::CreateOcb (Resmgr & resmgr, Device & device, Node & node, struct _client_info & info, resmgr_context_t * ctp, io_open_t * msg) throw (OcbException) [protected, virtual]

Create an open control block.

Parameters:

resmgr reference to the owning resource manager.
device reference to the managing device.
node reference to the managing node.
info credentials of client this OCB represents.
ctp 'C' resmgr library context pointer.
msg 'C' resmgr library open message.

Returns:

reference to created OCB.

virtual int QNXAPI::Resmgr::Device::Open (resmgr_context_t * ctp, io_open_t * msg, void * extra) throw () [protected, virtual]

Open a resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library open message.
extra 'C' resmgr library "extra" information for open.

Returns:

small positive integer from errno.h

Reimplemented in** QNXAPI::Sysmon **(p.338), and** QNXAPI::Sysmon **(p.338).virtual int QNXAPI::Resmgr::Device::Unlink (resmgr_context_t * ctp, io_unlink_t * msg, void * extra) throw () [protected, virtual]

Unlink (remove) a resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library unlink message.
extra 'C' resmgr library "extra" information for unlink.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::Rename (resmgr_context_t * ctp, io_rename_t * msg, io_rename_extra_t * extra) throw () [protected, virtual]

Rename a resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library rename message.
extra 'C' resmgr library "extra" information for rename (new path).

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::Mknod (resmgr_context_t * ctp, io_mknod_t * msg, void * extra) throw () [protected, virtual]

Mknod make a node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library mknod message.
extra 'C' resmgr library "extra" information for mknod.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::ReadLink (resmgr_context_t * ctp, io_readlink_t * msg, void * extra) throw () [protected, virtual]

Read the contents of a link resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library readlink message.
extra 'C' resmgr library "extra" information for a readlink.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::MakeLink (resmgr_context_t * ctp, io_link_t * msg, io_link_extra_t * extra) throw () [protected, virtual]

Create a link resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library link message.
extra 'C' resmgr library "extra" information for link.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::HardLink (resmgr_context_t * ctp, io_link_t * msg, Ocb * ocb) throw () [protected, virtual]

Create a hard link.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library link message.
ocb open control block on target node.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::SymLink (resmgr_context_t * ctp, io_link_t * msg, const std::string & target) throw () [protected, virtual]

Create a symbolic link.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library link message.
target contents of symlink.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::ReadDir (resmgr_context_t * ctp, io_read_t * msg, Ocb * ocb) throw () [protected, virtual]

Perform a readdir() operation on OCB.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library readdir message.
ocb Open Control Block of directory node.

Returns:

number of bytes returned.

virtual int QNXAPI::Resmgr::Device::Read (resmgr_context_t * ctp, io_read_t * msg, Ocb * ocb) throw () [protected, virtual]

Perform a read() operation on OCB.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb Open Control Block of non-directory node.

Returns:

number of bytes returned.

virtual int QNXAPI::Resmgr::Device::Write (resmgr_context_t * ctp, io_write_t * msg, Ocb * ocb) throw () [protected, virtual]

Perform a write() operation on OCB.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb Open Control Block of non-directory node.

Returns:

number of bytes returned.

virtual int QNXAPI::Resmgr::Device::CloseOcb (resmgr_context_t * ctp, void * msg, Ocb * ocb) throw () [protected, virtual]

Close an Open Control Block.

Parameters:

ctp 'C' resmgr library context pointer.

msg void pointer (not used).
ocb Open Control Block to close.

Returns:

small positive integer from errno.h

Reimplemented in** QNXAPI::Sysmon **(p.338), and** QNXAPI::Sysmon **(p.338).int QNXAPI::Resmgr::Device::BounceLink (resmgr_context_t * ctp, io_open_t * msg, const char * linkname) [protected]

Bounce a link (perform a redirection).

Bounce a connection operation on a symbolic link. This is intended to be called from connection callbacks when the node opened IsSymLink(). While this call does return, connection callbacks that make use of it should simply return the return from this function to their caller.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library io_open message pointer.
linkname symbolic link name.

Returns:

small positive integer from errno.h

int QNXAPI::Resmgr::Device::BounceLink (resmgr_context_t * ctp, io_open_t * msg, const char * link, const char * target, const char * remainder) [protected]

Bounce a connection operation on a symbolic link. This form of the call is designed to expand intermediate links (i.e. directories).

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library io_open message pointer.
link intermediate symbolic link component to be replaced.
target target path to substitute for link.
remainder remainder of path following substitution.

Returns:

small positive integer from errno.h

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

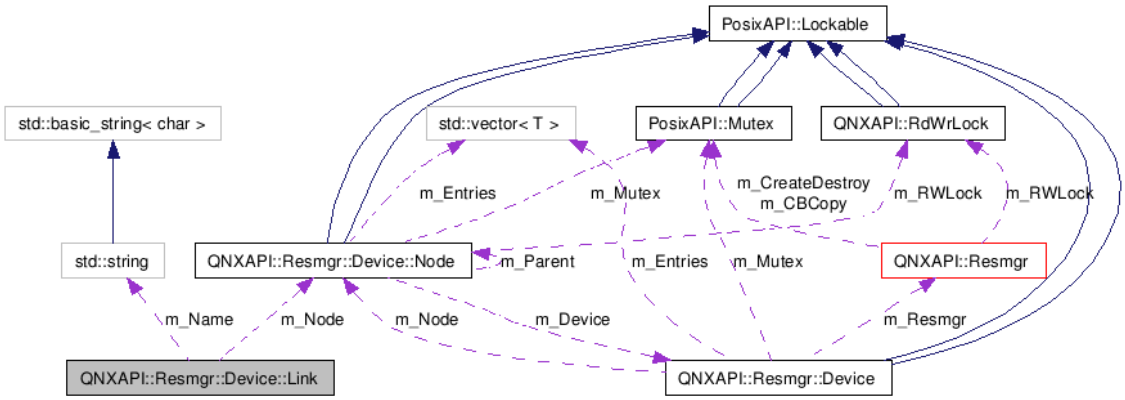
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

QNXAPI::Resmgr::Device::Link Class Reference

Resource Manager **Device Link**.

Collaboration diagram for QNXAPI::Resmgr::Device::Link:



Public Member Functions

- **Link** (const std::string &name, **Node** &node)
  Construct a link.

- **Link** (const std::string &name, **Node** &node)
  Construct a link.

Public Attributes

- std::string **m_Name**
  Links name.

- **Node** * **m_Node**
  Linked node.

- **Node** * **m_Node**
  Linked node.

Detailed Description

Resource Manager **Device Link**.

Since nodes are the actual data elements, and Posix supports multiple names for nodes (i.e. links), nodes themselves do not have a name member. Links provide names, and are the entities stored in the nodes entries member.

Definition at line 805 of file resmgr.svn-base.

Constructor & Destructor Documentation

QNXAPI::Resmgr::Device::Link::Link (const std::string &* name*, Node &* node*) [inline]

Construct a link.

Parameters:

name links name.
node linked node.

Definition at line 819 of file resmgr.svn-base.QNXAPI::Resmgr::Device::Link::Link (const std::string &* name*, Node &* node*) [inline]

Construct a link.

Parameters:

name links name.
node linked node.

Definition at line 819 of file resmgr.

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

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

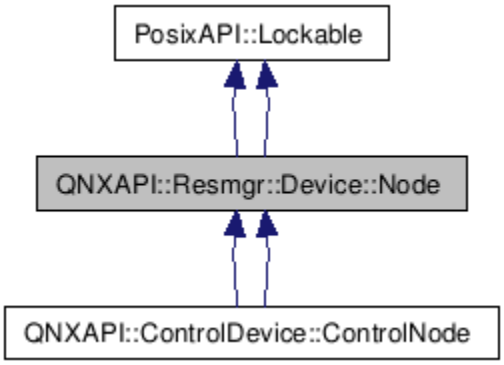
QNXAPI::Resmgr::Device::Node Class Reference

Resource Manager **Device Node**.

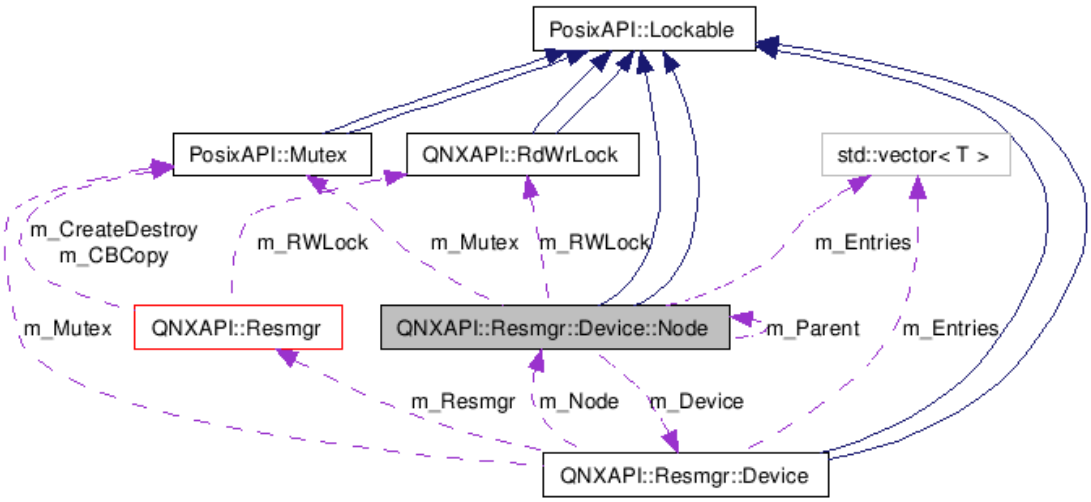
Inherits **PosixAPI::Lockable**, and **PosixAPI::Lockable**.

Inherited by **QNXAPI::ControlDevice::ControlNode**, and **QNXAPI::ControlDevice::ControlNode**.

Inheritance diagram for QNXAPI::Resmgr::Device::Node:



Collaboration diagram for QNXAPI::Resmgr::Device::Node:



Public Types

- typedef sigc::signal< void, const **Node** & > **OnDestructionCallback**
  Typedef for OnDestruction callback.

- typedef sigc::signal< void, const **Node** & > **OnDestructionCallback**
  Typedef for OnDestruction callback.

Public Member Functions

- **Node** (void)
  *Construct a new directory **Node**.*

- **Node** (**Resmgr::Device** ***device**, **Node** &parent, dev_t type, mode_t mode)
  *Construct a new **Node**, specifying type, and mode.*

- **Node** (**Resmgr::Device** ***device**, **Node** &parent, void *data, IOFUNC_ATTR_T *attr, **LinkEntries** &entries)
  *Construct a new **Node**, specifying data, attributes and entries.*

- **Node** (**Resmgr::Device** ***device**, **Node** &parent, dev_t type, mode_t mode, struct _client_info &info, IOFUNC_ATTR_T *dattr)
  *Construct a new **Node**, specifying type, mode, client information, and directory attributes.*

- ino_t **Inode** (void) const
  Return the nodes serial number (Inode).

- void **Inode** (ino_t inode)
  Set the nodes serial number (Inode).

- **Device** * **ControllingDevice** (void)
  Return the controlling device.

- IOFUNC_ATTR_T * **Attributes** (void)
  Return a pointer to the nodes attributes.

- void **Attributes** (mode_t mode)
  Set the nodes mode attribute.

- mode_t **Mode** (void) const
  Return the nodes mode.

- void **Mode** (mode_t mode)
  Set the nodes mode attribute.

- void * **Data** (void)

Return a pointer to the nodes data.

- void **Data** (void *data)
  Set the nodes data pointer.

- int **Fill** (void *data, int length, int offset=0) throw ()
  Fill the nodes data container.

- int **Truncate** (int length=0) throw ()
  Truncate the nodes data container.

- **Link** * **operator[]** (int idx) const
  Return a reference to a link maintained by node.

- **LinkEntries** & **Entries** (void)
  Return a reference to the nodes link entries vector.

- void **AddLink** (**Link** *link) throw (NodeException)
  Add a link to this nodes entries vector.

- void **RemoveLink** (**Link** &link)
  Remove a link from this nodes entries vector.

- **Link** & **Find** (const std::string &path) throw (NodeException)
  Get a reference to the link named by path.

- int **CheckAccess** (resmgr_context_t *ctp, const std::string &path, unsigned access, struct _client_info &client_info, IOFUNC_ATTR_T **attr) throw (SymLinkException)
  Check the accessability of path (relative to this node) by a specific client.

- bool **TargetExists** (const std::string &path) const throw ()
  Determine if path exists within node.

- int **ReadDir** (resmgr_context_t *ctp, io_read_t *msg, **Resmgr::Ocb** &**ocb**) throw ()
  Perform readdir() operation on node.

- virtual int **Read** (resmgr_context_t *ctp, io_read_t *msg, **Resmgr::Ocb** &**ocb**) throw ()
  Perform read() operation on node.

- virtual int **Write** (resmgr_context_t *ctp, io_write_t *msg, **Resmgr::Ocb** &**ocb**) throw ()
  Perform write() operation on node.

- bool **IsDirectory** () const
  Determine if node is a directory.

- bool **IsRegular** () const
  Determine if node is a regular file.

- bool **IsSymLink** () const
  Determine if node is a symbolic link.

- void **IncRefCount** (void)
  Increment the nodes reference count.

- bool **DecRefCount** (void)
  Decrement the nodes reference count.

- void **Lock** (**PosixAPI::Lockable::Mode** mode=PosixAPI::Lockable::Exclusive) const
  Lock the node. If called with no parameters, then an exclusive lock is obtained.

- void **TryLock** (**PosixAPI::Lockable::Mode** mode=PosixAPI::Lockable::Exclusive) const
  Attempt to obtain a lock on the node. If called with no parameters, then an exclusive lock is attempted.

- void **Unlock** (void) const
  Release lock on node.

- **Node** & **Parent** (void)
  Return a reference to the nodes parent (node in which this node was created).

- virtual **~Node** ()
  Destroy a node.

- **Node** (void)
  *Construct a new directory **Node**.*

- **Node** (**Resmgr::Device** ***device**, **Node** &parent, dev_t type, mode_t mode)
  *Construct a new **Node**, specifying type, and mode.*

- **Node** (**Resmgr::Device** ***device**, **Node** &parent, void *data, IOFUNC_ATTR_T *attr, **LinkEntries** &entries)
  *Construct a new **Node**, specifying data, attributes and entries.*

- **Node** (**Resmgr::Device** ***device**, **Node** &parent, dev_t type, mode_t mode, struct _client_info &info, IOFUNC_ATTR_T *dattr)
  *Construct a new **Node**, specifying type, mode, client information, and directory attributes.*

- ino_t **Inode** (void) const
  Return the nodes serial number (Inode).

- void **Inode** (ino_t inode)
  Set the nodes serial number (Inode).

- **Device** * **ControllingDevice** (void)
  Return the controlling device.

- IOFUNC_ATTR_T * **Attributes** (void)

Return a pointer to the nodes attributes.

- void **Attributes** (mode_t mode)
  Set the nodes mode attribute.

- mode_t **Mode** (void) const
  Return the nodes mode.

- void **Mode** (mode_t mode)
  Set the nodes mode attribute.

- void * **Data** (void)
  Return a pointer to the nodes data.

- void **Data** (void *data)
  Set the nodes data pointer.

- int **Fill** (void *data, int length, int offset=0) throw ()
  Fill the nodes data container.

- int **Truncate** (int length=0) throw ()
  Truncate the nodes data container.

- **Link** * **operator[]** (int idx) const
  Return a reference to a link maintained by node.

- **LinkEntries** & **Entries** (void)
  Return a reference to the nodes link entries vector.

- void **AddLink** (**Link** *link) throw (NodeException)
  Add a link to this nodes entries vector.

- void **RemoveLink** (**Link** &link)
  Remove a link from this nodes entries vector.

- **Link** & **Find** (const std::string &path) throw (NodeException)
  Get a reference to the link named by path.

- int **CheckAccess** (resmgr_context_t *ctp, const std::string &path, unsigned access, struct _client_info &client_info, IOFUNC_ATTR_T **attr) throw (SymLinkException)
  Check the accessability of path (relative to this node) by a specific client.

- bool **TargetExists** (const std::string &path) const throw ()
  Determine if path exists within node.

- int **ReadDir** (resmgr_context_t *ctp, io_read_t *msg, **Resmgr::Ocb** &**ocb**) throw ()
  Perform readdir() operation on node.

- virtual int **Read** (resmgr_context_t *ctp, io_read_t *msg, **Resmgr::Ocb** &**ocb**) throw ()
  Perform read() operation on node.

- virtual int **Write** (resmgr_context_t *ctp, io_write_t *msg, **Resmgr::Ocb** &**ocb**) throw ()
  Perform write() operation on node.

- bool **IsDirectory** () const
  Determine if node is a directory.

- bool **IsRegular** () const
  Determine if node is a regular file.

- bool **IsSymLink** () const
  Determine if node is a symbolic link.

- void **IncRefCount** (void)
  Increment the nodes reference count.

- bool **DecRefCount** (void)
  Decrement the nodes reference count.

- void **Lock** (**PosixAPI::Lockable::Mode** mode=PosixAPI::Lockable::Exclusive) const
  Lock the node. If called with no parameters, then an exclusive lock is obtained.

- void **TryLock** (**PosixAPI::Lockable::Mode** mode=PosixAPI::Lockable::Exclusive) const
  Attempt to obtain a lock on the node. If called with no parameters, then an exclusive lock is attempted.

- void **Unlock** (void) const
  Release lock on node.

- **Node** & **Parent** (void)
  Return a reference to the nodes parent (node in which this node was created).

- virtual **~Node** ()
  Destroy a node.

Detailed Description

Resource Manager **Device Node**.

This class provides a default node for insertion into the prefix namespace.

Definition at line 470 of file resmgr.svn-base.

Constructor & Destructor Documentation

QNXAPI::Resmgr::Device::Node::Node (Resmgr::Device * device, Node & parent, dev_t type, mode_t mode)

Construct a new **Node**, specifying type, and mode.

Parameters:

device device responsible for this node.
parent parent node.
type node type (eg. S_IFDIR or S_IFLNK).
mode node mode (see sys/stat.h).

QNXAPI::Resmgr::Device::Node::Node (Resmgr::Device * device, Node & parent, void * data, IOFUNC_ATTR_T * attr, LinkEntries & entries)

Construct a new **Node**, specifying data, attributes and entries.

Parameters:

device device responsible for this node.
parent parent node.
data pointer to data for node.
attr pointer to attributes to be copied into node.
entries LinkEntries to copy.

QNXAPI::Resmgr::Device::Node::Node (Resmgr::Device * device, Node & parent, dev_t type, mode_t mode, struct _client_info & info, IOFUNC_ATTR_T * dattr)

Construct a new **Node**, specifying type, mode, client information, and directory attributes.

Parameters:

device device responsible for this node.
parent parent node.
type node type (eg. S_IFDIR or S_IFLNK).
mode node mode (see sys/stat.h).
info client credentials.
dattr pointer to parent directory attributes.

QNXAPI::Resmgr::Device::Node::Node (Resmgr::Device * device, Node & parent, dev_t type, mode_t mode)

Construct a new **Node**, specifying type, and mode.

Parameters:

device device responsible for this node.
parent parent node.
type node type (eg. S_IFDIR or S_IFLNK).
mode node mode (see sys/stat.h).

QNXAPI::Resmgr::Device::Node::Node (Resmgr::Device * device, Node & parent, void * data, IOFUNC_ATTR_T * attr, LinkEntries & entries)

Construct a new **Node**, specifying data, attributes and entries.

Parameters:

device device responsible for this node.
parent parent node.
data pointer to data for node.
attr pointer to attributes to be copied into node.
entries LinkEntries to copy.

QNXAPI::Resmgr::Device::Node::Node (Resmgr::Device * device, Node & parent, dev_t type, mode_t mode, struct _client_info & info, IOFUNC_ATTR_T * dattr)

Construct a new **Node**, specifying type, mode, client information, and directory attributes.

Parameters:

device device responsible for this node.
parent parent node.
type node type (eg. S_IFDIR or S_IFLNK).
mode node mode (see sys/stat.h).
info client credentials.
dattr pointer to parent directory attributes.

Member Function Documentation

ino_t QNXAPI::Resmgr::Device::Node::Inode (void) const

Return the nodes serial number (Inode).

Returns:

nodes serial number.

void QNXAPI::Resmgr::Device::Node::Inode (ino_t inode)

Set the nodes serial number (Inode).

Parameters:

inode nodes serial number.

Device* QNXAPI::Resmgr::Device::Node::ControllingDevice (void) [inline]

Return the controlling device.

Returns:

pointer to controlling device.

Definition at line 535 of file resmgr.svn-base.IOFUNC_ATTR_T* QNXAPI::Resmgr::Device::Node::Attributes (void)

Return a pointer to the nodes attributes.

Returns:

pointer to nodes attribute structure.

void QNXAPI::Resmgr::Device::Node::Attributes (mode_t* mode*)

Set the nodes mode attribute.

Parameters:

mode mode to set on attributes structure.

mode_t QNXAPI::Resmgr::Device::Node::Mode (void) const

Return the nodes mode.

Returns:

mode member of nodes attribute structure.

void QNXAPI::Resmgr::Device::Node::Mode (mode_t* mode*)

Set the nodes mode attribute.

Parameters:

mode mode to set on attributes structure.

void* QNXAPI::Resmgr::Device::Node::Data (void)

Return a pointer to the nodes data.

Returns:

pointer to nodes data.

void QNXAPI::Resmgr::Device::Node::Data (void * data*)

Set the nodes data pointer.

Caution: pointer to data is replaced, old data *is not* freed.

Parameters:

data pointer to allocated data for node.

int QNXAPI::Resmgr::Device::Node::Fill (void * data, int length, int offset = 0) throw ()

Fill the nodes data container.

Parameters:

data pointer to fill data.
length size of fill data.
offset offset in data container to fill at (default 0).

Returns:

new offset (after fill), or a negative number from errno.h.

int QNXAPI::Resmgr::Device::Node::Truncate (int length = 0) throw ()

Truncate the nodes data container.

Parameters:

length size to truncate to (default 0).

Returns:

small positive int from errno.h.

Link* QNXAPI::Resmgr::Device::Node::operator[] (int idx) const

Return a reference to a link maintained by node.

Useful only on directory nodes.

Parameters:

idx sub-node to retrieve.

Returns:

reference to indexed link.

LinkEntries& QNXAPI::Resmgr::Device::Node::Entries (void)

Return a reference to the nodes link entries vector.

Returns:

reference to nodes entries vector.

void QNXAPI::Resmgr::Device::Node::AddLink (Link * link) throw (NodeException)

Add a link to this nodes entries vector.

Parameters:

link link to add to this nodes entries vector.

Referenced by QNXAPI::Resmgr::Device::AddLink().void QNXAPI::Resmgr::Device::Node::RemoveLink (Link & link)

Remove a link from this nodes entries vector.

Parameters:

link link to be removed from this nodes entries vector.

Link& QNXAPI::Resmgr::Device::Node::Find (const std::string & path) throw (NodeException)

Get a reference to the link named by path.

Parameters:

path complete path to link.

Returns:

reference to located link.

int QNXAPI::Resmgr::Device::Node::CheckAccess (resmgr_context_t * ctp, const std::string & path, unsigned access, struct _client_info & client_info, IOFUNC_ATTR_T ** attr) throw (SymLinkException)

Check the accessability of path (relative to this node) by a specific client.

Parameters:

ctp 'C' resource manager context pointer.
path relative path of target (from this node).
access access desired by client.
client_info credentials of client.
attr address of pointer where attributes of last existing component is store.

Returns:

a small positive integer from errno.h indicating accessability.

bool QNXAPI::Resmgr::Device::Node::TargetExists (const std::string & path) const throw ()

Determine if path exists within node.

Parameters:

path relative (to this node) to test.

Returns:

boolean true if path exists, false if path doesn't exist.

int QNXAPI::Resmgr::Device::Node::ReadDir (resmgr_context_t * ctp, io_read_t * msg, Resmgr::Ocb & ocb) throw ()

Perform readdir() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb open control block to control read.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::Node::Read (resmgr_context_t * ctp, io_read_t * msg, Resmgr::Ocb & ocb) throw () [virtual]

Perform read() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb open control block to control read.

Returns:

small positive integer from errno.h

Reimplemented in QNXAPI::ControlDevice::ControlNode (p.58), and QNXAPI::ControlDevice::ControlNode (p.58).virtual int QNXAPI::Resmgr::Device::Node::Write (resmgr_context_t * ctp, io_write_t * msg, Resmgr::Ocb & ocb) throw () [virtual]

Perform write() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library write message.
ocb open control block to control write.

Returns:

small positive integer from errno.h

Reimplemented in QNXAPI::ControlDevice::ControlNode (p.58), and QNXAPI::ControlDevice::ControlNode (p.58).bool QNXAPI::Resmgr::Device::Node::IsDirectory () const

Determine if node is a directory.

Returns:

true if node is a directory, false otherwise.

bool QNXAPI::Resmgr::Device::Node::IsRegular () const

Determine if node is a regular file.

Returns:

true if node is a regular file, false otherwise.

bool QNXAPI::Resmgr::Device::Node::IsSymLink () const

Determine if node is a symbolic link.

Returns:

true if node is a symbolic link, false otherwise.

bool QNXAPI::Resmgr::Device::Node::DecRefCount (void)

Decrement the nodes reference count.

Returns:

true if node is now unreferenced.

void QNXAPI::Resmgr::Device::Node::Lock (PosixAPI::Lockable::Mode **mode =** ***PosixAPI::Lockable::Exclusive) const [virtual]***

Lock the node. If called with no parameters, then an exclusive lock is obtained.

Parameters:

mode boolean; true if caller wants exclusive lock false otherwise

Returns:

small positive integer from errno.h (see pthread*trylock docs).

Implements *PosixAPI::Lockable* ***(*****p.98*****).void QNXAPI::Resmgr::Device::Node::TryLock (PosixAPI::Lockable::Mode*** **mode =** ***PosixAPI::Lockable::Exclusive) const [virtual]***

Attempt to obtain a lock on the node. If called with no parameters, then an exclusive lock is attempted.

Parameters:

mode boolean; true if caller wants exclusive lock false otherwise

Returns:

small positive integer from errno.h (see pthread*trylock docs).

***Implements** PosixAPI::Lockable **(p.98)**.Node& QNXAPI::Resmgr::Device::Node::Parent (void) [inline]*

Return a reference to the nodes parent (node in which this node was created).

Returns:

reference to parent node.

Definition at line 768 of file resmgr.svn-base.ino_t QNXAPI::Resmgr::Device::Node::Inode (void) const

Return the nodes serial number (Inode).

Returns:

nodes serial number.

void QNXAPI::Resmgr::Device::Node::Inode (ino_t inode)

Set the nodes serial number (Inode).

Parameters:

inode nodes serial number.

Device* QNXAPI::Resmgr::Device::Node::ControllingDevice (void) [inline]

Return the controlling device.

Returns:

pointer to controlling device.

Definition at line 535 of file resmgr.IOFUNC_ATTR_T* QNXAPI::Resmgr::Device::Node::Attributes (void)

Return a pointer to the nodes attributes.

Returns:

pointer to nodes attribute structure.

void QNXAPI::Resmgr::Device::Node::Attributes (mode_t mode)

Set the nodes mode attribute.

Parameters:

mode mode to set on attributes structure.

mode_t QNXAPI::Resmgr::Device::Node::Mode (void) const

Return the nodes mode.

Returns:

mode member of nodes attribute structure.

void QNXAPI::Resmgr::Device::Node::Mode (mode_t* mode*)

Set the nodes mode attribute.

Parameters:

mode mode to set on attributes structure.

void* QNXAPI::Resmgr::Device::Node::Data (void)

Return a pointer to the nodes data.

Returns:

pointer to nodes data.

void QNXAPI::Resmgr::Device::Node::Data (void * data*)

Set the nodes data pointer.

Caution: pointer to data is replaced, old data *is not* freed.

Parameters:

data pointer to allocated data for node.

int QNXAPI::Resmgr::Device::Node::Fill (void * data, int length, int offset = 0) throw ()

Fill the nodes data container.

Parameters:

data pointer to fill data.
length size of fill data.
offset offset in data container to fill at (default 0).

Returns:

new offset (after fill), or a negative number from errno.h.

int QNXAPI::Resmgr::Device::Node::Truncate (int* length = *0) throw ()

Truncate the nodes data container.

Parameters:

length size to truncate to (default 0).

Returns:

small positive int from errno.h.

Link* QNXAPI::Resmgr::Device::Node::operator[] (int* idx*) const

Return a reference to a link maintained by node.

Useful only on directory nodes.

Parameters:

idx sub-node to retrieve.

Returns:

reference to indexed link.

LinkEntries& QNXAPI::Resmgr::Device::Node::Entries (void)

Return a reference to the nodes link entries vector.

Returns:

reference to nodes entries vector.

void QNXAPI::Resmgr::Device::Node::AddLink (Link** link*) throw (NodeException)

Add a link to this nodes entries vector.

Parameters:

link link to add to this nodes entries vector.

void QNXAPI::Resmgr::Device::Node::RemoveLink (Link &* link*)

Remove a link from this nodes entries vector.

Parameters:

link link to be removed from this nodes entries vector.

Link& QNXAPI::Resmgr::Device::Node::Find (const std::string &* path*) throw (NodeException)

Get a reference to the link named by path.

Parameters:

path complete path to link.

Returns:

reference to located link.

int QNXAPI::Resmgr::Device::Node::CheckAccess (resmgr_context_t * ctp, const std::string & path, unsigned access, struct _client_info & client_info, IOFUNC_ATTR_T ** attr) throw (SymLinkException)

Check the accessability of path (relative to this node) by a specific client.

Parameters:

ctp 'C' resource manager context pointer.
path relative path of target (from this node).
access access desired by client.
client_info credentials of client.
attr address of pointer where attributes of last existing component is store.

Returns:

a small positive integer from errno.h indicating accessability.

bool QNXAPI::Resmgr::Device::Node::TargetExists (const std::string & path) const throw ()

Determine if path exists within node.

Parameters:

path relative (to this node) to test.

Returns:

boolean true if path exists, false if path doesn't exist.

int QNXAPI::Resmgr::Device::Node::ReadDir (resmgr_context_t * ctp, io_read_t * msg, Resmgr::Ocb & ocb) throw ()

Perform readdir() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb open control block to control read.

Returns:

small positive integer from errno.h

virtual int QNXAPI::Resmgr::Device::Node::Read (resmgr_context_t * ctp, io_read_t * msg, Resmgr::Ocb & ocb) throw () [virtual]

Perform read() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library read message.
ocb open control block to control read.

Returns:

small positive integer from errno.h

Reimplemented in** QNXAPI::ControlDevice::ControlNode **(**p.58**), and** QNXAPI::ControlDevice::ControlNode **(**p.58**).virtual int QNXAPI::Resmgr::Device::Node::Write (resmgr_context_t * **ctp**, io_write_t * **msg**, Resmgr::Ocb & **ocb**) throw () [virtual]

Perform write() operation on node.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library write message.
ocb open control block to control write.

Returns:

small positive integer from errno.h

Reimplemented in** QNXAPI::ControlDevice::ControlNode **(**p.58**), and** QNXAPI::ControlDevice::ControlNode **(**p.58**).bool QNXAPI::Resmgr::Device::Node::IsDirectory () const

Determine if node is a directory.

Returns:

true if node is a directory, false otherwise.

bool QNXAPI::Resmgr::Device::Node::IsRegular () const

Determine if node is a regular file.

Returns:

true if node is a regular file, false otherwise.

bool QNXAPI::Resmgr::Device::Node::IsSymLink () const

Determine if node is a symbolic link.

Returns:

true if node is a symbolic link, false otherwise.

bool QNXAPI::Resmgr::Device::Node::DecRefCount (void)

Decrement the nodes reference count.

Returns:

true if node is now unreferenced.

void QNXAPI::Resmgr::Device::Node::Lock (PosixAPI::Lockable::Mode* mode = *PosixAPI::Lockable::Exclusive) const [virtual]

Lock the node. If called with no parameters, then an exclusive lock is obtained.

Parameters:

mode boolean; true if caller wants exclusive lock false otherwise

Returns:

small positive integer from errno.h (see pthread*trylock docs).

Implements** PosixAPI::Lockable* **(p.98)*.void QNXAPI::Resmgr::Device::Node::TryLock (PosixAPI::Lockable::Mode* mode = *PosixAPI::Lockable::Exclusive) const [virtual]

Attempt to obtain a lock on the node. If called with no parameters, then an exclusive lock is attempted.

Parameters:

mode boolean; true if caller wants exclusive lock false otherwise

Returns:

small positive integer from errno.h (see pthread*trylock docs).

Implements** PosixAPI::Lockable* **(p.98)*.Node& QNXAPI::Resmgr::Device::Node::Parent (void) [inline]

Return a reference to the nodes parent (node in which this node was created).

Returns:

reference to parent node.

Definition at line 768 of file resmgr.

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

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

QNXAPI::Resmgr::IntrIID Struct Reference

Pair for associating IRQ and IID.

Detailed Description

Pair for associating IRQ and IID.

Definition at line 398 of file resmgr.svn-base.

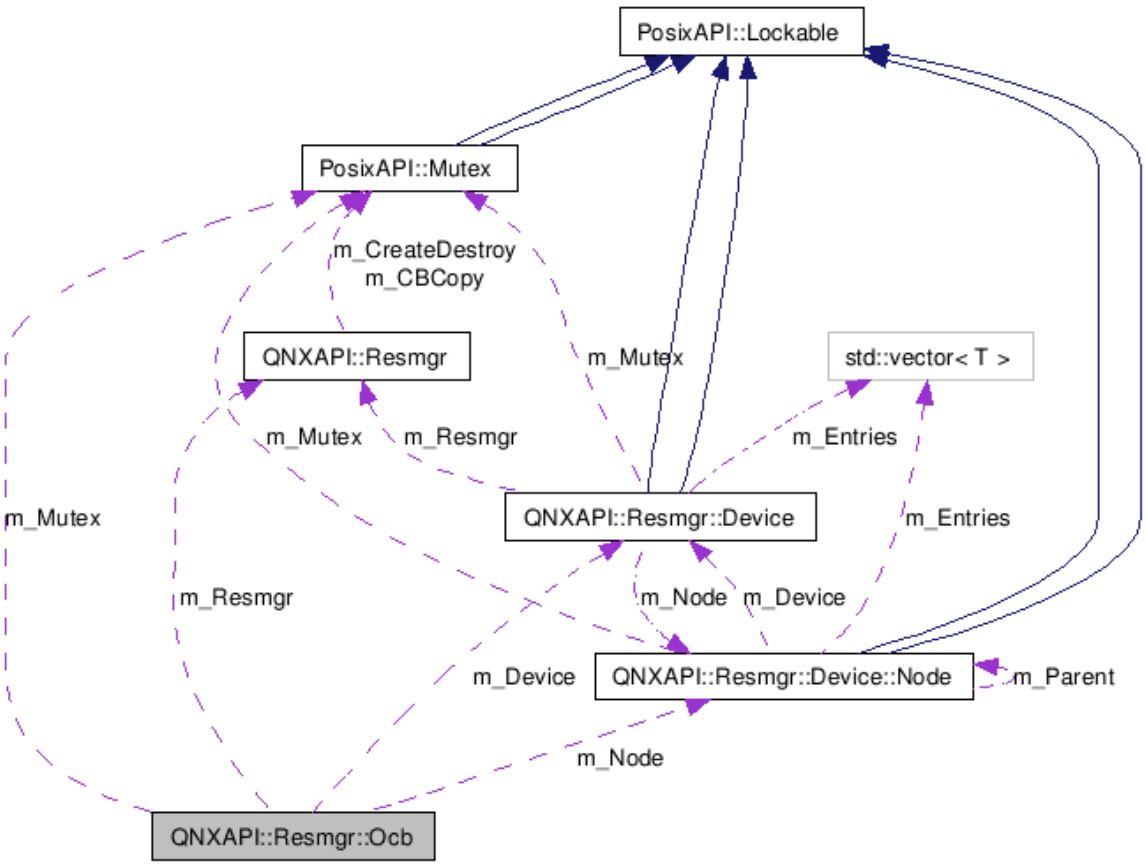
The documentation for this struct was generated from the following file:

- include/qfc/.svn/text-base/resmgr.svn-base

QNXAPI::Resmgr::Ocb Class Reference

Open Control Block class.

Collaboration diagram for QNXAPI::Resmgr::Ocb:



Public Member Functions

- **Ocb** (**Resmgr** &resmgr)
  Create a new OCB.

- **Ocb** (**Resmgr** &resmgr, **Device** &**device**, **Device::Node** &node, struct _client_info &info)
  Create a new OCB.

- IOFUNC_OCB_T * **Derived** (void) throw ()
  Return a pointer to the derived iofunc_ocb_t structure.

- IOFUNC_ATTR_T * **RootAttributes** (void) throw ()
  Return a pointer to the derived iofunc_ocb_t structure.

- IOFUNC_ATTR_T * **Attributes** (void) throw ()
  Return a pointer to the OCB's attribute structure.

- off_t **operator+=** (int inc) throw ()
  Add an increment to the OCB's offset.

- off_t **operator-=** (int inc) throw ()
  Subtract an increment to the OCB's offset.

- off_t **Offset** (void) throw ()
  Return the current offset for the OCB.

- void **Offset** (off_t offset) throw ()
  Set the current offset for the OCB.

- **Device::Node** & **Node** (void) throw ()
  Return a reference to the node associated with this OCB.

- virtual int **ReadDir** (resmgr_context_t *ctp, io_read_t *msg) throw ()
  Perform a readdir() operation on the node associated with this OCB.

- virtual int **Read** (resmgr_context_t *ctp, io_read_t *msg) throw ()

Perform a read() operation on the node associated with this OCB.

- virtual int **Write** (resmgr_context_t *ctp, io_write_t *msg) throw ()
  Perform a write() operation on the node associated with this OCB.

- bool **IsDirectory** () const
  *Determine if node upon which this **Ocb** is open is a directory.*

- bool **IsRegular** () const
  *Determine if node upon which this **Ocb** is open is a regular file.*

- virtual **~Ocb** ()
  Destroy an OCB.

- **Ocb** (**Resmgr** &resmgr)
  Create a new OCB.

- **Ocb** (**Resmgr** &resmgr, **Device** &**device**, **Device::Node** &node, struct _client_info &info)
  Create a new OCB.

- IOFUNC_OCB_T * **Derived** (void) throw ()
  Return a pointer to the derived iofunc_ocb_t structure.

- IOFUNC_ATTR_T * **RootAttributes** (void) throw ()
  Return a pointer to the derived iofunc_ocb_t structure.

- IOFUNC_ATTR_T * **Attributes** (void) throw ()
  Return a pointer to the OCB's attribute structure.

- off_t **operator+=** (int inc) throw ()
  Add an increment to the OCB's offset.

- off_t **operator-=** (int inc) throw ()
  Subtract an increment to the OCB's offset.

- off_t **Offset** (void) throw ()
  Return the current offset for the OCB.

- void **Offset** (off_t offset) throw ()
  Set the current offset for the OCB.

- **Device::Node** & **Node** (void) throw ()
  Return a reference to the node associated with this OCB.

- virtual int **ReadDir** (resmgr_context_t *ctp, io_read_t *msg) throw ()
  Perform a readdir() operation on the node associated with this OCB.

- virtual int **Read** (resmgr_context_t *ctp, io_read_t *msg) throw ()
  Perform a read() operation on the node associated with this OCB.

- virtual int **Write** (resmgr_context_t *ctp, io_write_t *msg) throw ()
  Perform a write() operation on the node associated with this OCB.

- bool **IsDirectory** () const
  *Determine if node upon which this **Ocb** is open is a directory.*

- bool **IsRegular** () const
  *Determine if node upon which this **Ocb** is open is a regular file.*

- virtual **~Ocb** ()
  Destroy an OCB.

Public Attributes

- int **m_NonBlock**
  flag indicating wether this open is non-blocking.

Detailed Description

Open Control Block class.

This class provides a framework for OCB's (Open Control Blocks).

Definition at line 1342 of file resmgr.svn-base.

Constructor & Destructor Documentation

QNXAPI::Resmgr::Ocb::Ocb (Resmgr & resmgr)

Create a new OCB.

Parameters:

resmgr instance of resmanager that manages this OCB.

QNXAPI::Resmgr::Ocb::Ocb (Resmgr & resmgr, Device & device, Device::Node & node, struct _client_info & info)

Create a new OCB.

Parameters:

resmgr instance of resmanager that manages this OCB.
device instance of device that manages this OCB.
node instance of node representing the parent.
info pointer to client credentials.

QNXAPI::Resmgr::Ocb::Ocb (Resmgr & resmgr)

Create a new OCB.

Parameters:

resmgr instance of resmanager that manages this OCB.

QNXAPI::Resmgr::Ocb::Ocb (Resmgr & resmgr, Device & device, Device::Node & node, struct _client_info & info)

Create a new OCB.

Parameters:

resmgr instance of resmanager that manages this OCB.
device instance of device that manages this OCB.
node instance of node representing the parent.
info pointer to client credentials.

Member Function Documentation

IOFUNC_OCB_T* QNXAPI::Resmgr::Ocb::Derived (void) throw ()

Return a pointer to the derived iofunc_ocb_t structure.

Returns:

a pointer to a derived iofunc_ocb_t structure.

IOFUNC_ATTR_T* QNXAPI::Resmgr::Ocb::RootAttributes (void) throw () [inline]

Return a pointer to the derived iofunc_ocb_t structure.

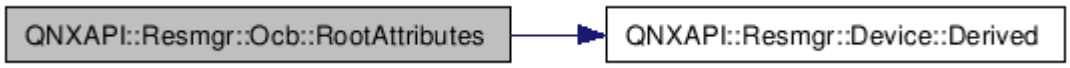
Returns:

a pointer to a derived iofunc_ocb_t structure.

Definition at line 1376 of file resmgr.svn-base.

References QNXAPI::Resmgr::Device::Derived().

Here is the call graph for this function:



IOFUNC_ATTR_T* QNXAPI::Resmgr::Ocb::Attributes (void) throw ()

Return a pointer to the OCB's attribute structure.

Returns:

a pointer to a derived iofunc_attr_t structure.

off_t QNXAPI::Resmgr::Ocb::operator+= (int inc) throw ()

Add an increment to the OCB's offset.

Returns:
new OCB offset.

off_t QNXAPI::Resmgr::Ocb::operator-= (int inc) throw ()

Subtract an increment to the OCB's offset.

Returns:
new OCB offset.

off_t QNXAPI::Resmgr::Ocb::Offset (void) throw ()

Return the current offset for the OCB.

Returns:
current OCB offset.

void QNXAPI::Resmgr::Ocb::Offset (off_t offset) throw ()

Set the current offset for the OCB.

Parameters:
offset new current offset for OCB.

Device::Node& QNXAPI::Resmgr::Ocb::Node (void) throw ()

Return a reference to the node associated with this OCB.

Returns:
a reference to node associated with this OCB.

virtual int QNXAPI::Resmgr::Ocb::ReadDir (resmgr_context_t * ctp, io_read_t * msg) throw () [virtual]

Perform a readdir() operation on the node associated with this OCB.

Parameters:
ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_open_t.

Returns:
number of bytes "read".

virtual int QNXAPI::Resmgr::Ocb::Read (resmgr_context_t * ctp, io_read_t * msg) throw () [virtual]

Perform a read() operation on the node associated with this OCB.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_open_t.

Returns:

number of bytes "read".

virtual int QNXAPI::Resmgr::Ocb::Write (resmgr_context_t * ctp, io_write_t * msg) throw () [virtual]

Perform a write() operation on the node associated with this OCB.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_open_t.

Returns:

number of bytes "written".

bool QNXAPI::Resmgr::Ocb::IsDirectory () const

Determine if node upon which this **Ocb** is open is a directory.

Returns:

true if node is a directory, false otherwise.

bool QNXAPI::Resmgr::Ocb::IsRegular () const

Determine if node upon which this **Ocb** is open is a regular file.

Returns:

true if node is a regular file, false otherwise.

IOFUNC_OCB_T* QNXAPI::Resmgr::Ocb::Derived (void) throw ()

Return a pointer to the derived iofunc_ocb_t structure.

Returns:

a pointer to a derived iofunc_ocb_t structure.

IOFUNC_ATTR_T* QNXAPI::Resmgr::Ocb::RootAttributes (void) throw () [inline]

Return a pointer to the derived iofunc_ocb_t structure.

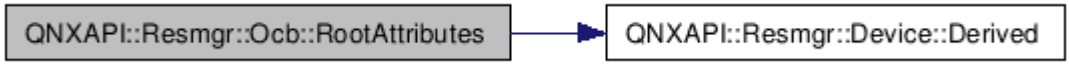
Returns:

a pointer to a derived iofunc_ocb_t structure.

Definition at line 1376 of file resmgr.

References QNXAPI::Resmgr::Device::Derived().

Here is the call graph for this function:



IOFUNC_ATTR_T* QNXAPI::Resmgr::Ocb::Attributes (void) throw ()

Return a pointer to the OCB's attribute structure.

Returns:

a pointer to a derived iofunc_attr_t structure.

off_t QNXAPI::Resmgr::Ocb::operator+= (int* inc*) throw ()

Add an increment to the OCB's offset.

Returns:

new OCB offset.

off_t QNXAPI::Resmgr::Ocb::operator-= (int* inc*) throw ()

Subtract an increment to the OCB's offset.

Returns:

new OCB offset.

off_t QNXAPI::Resmgr::Ocb::Offset (void) throw ()

Return the current offset for the OCB.

Returns:

current OCB offset.

void QNXAPI::Resmgr::Ocb::Offset (off_t* offset*) throw ()

Set the current offset for the OCB.

Parameters:

offset new current offset for OCB.

Device::Node& QNXAPI::Resmgr::Ocb::Node (void) throw ()

Return a reference to the node associated with this OCB.

Returns:

a reference to node associated with this OCB.

virtual int QNXAPI::Resmgr::Ocb::ReadDir (resmgr_context_t * ctp, io_read_t * msg) throw () [virtual]

Perform a readdir() operation on the node associated with this OCB.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_open_t.

Returns:

number of bytes "read".

virtual int QNXAPI::Resmgr::Ocb::Read (resmgr_context_t * ctp, io_read_t * msg) throw () [virtual]

Perform a read() operation on the node associated with this OCB.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_open_t.

Returns:

number of bytes "read".

virtual int QNXAPI::Resmgr::Ocb::Write (resmgr_context_t * ctp, io_write_t * msg) throw () [virtual]

Perform a write() operation on the node associated with this OCB.

Parameters:

ctp pointer to a QNX 'C' library resmgr_context_t.
msg pointer to a QNX 'C' library io_open_t.

Returns:

number of bytes "written".

bool QNXAPI::Resmgr::Ocb::IsDirectory () const

Determine if node upon which this **Ocb** is open is a directory.

Returns:

true if node is a directory, false otherwise.

bool QNXAPI::Resmgr::Ocb::IsRegular () const

Determine if node upon which this **Ocb** is open is a regular file.

Returns:

true if node is a regular file, false otherwise.

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

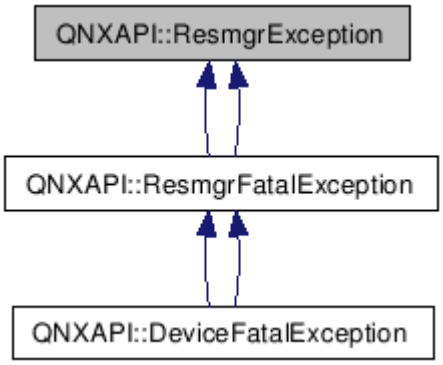
- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

QNXAPI::ResmgrException Class Reference

Resource Manager non-fatal exception base class.

Inherited by **QNXAPI::ResmgrFatalException**, and **QNXAPI::ResmgrFatalException**.

Inheritance diagram for QNXAPI::ResmgrException:



Public Member Functions

- **ResmgrException** (int err)
  *Build a new **ResmgrException**.*

- const char * **ErrorDescription** (void)
  Retrieve a description of a standard error.

- uint32_t **ErrorCode** (void)
  Retrieve the raw error code.

- **ResmgrException** (int err)
  *Build a new **ResmgrException**.*

- const char * **ErrorDescription** (void)
  Retrieve a description of a standard error.

- uint32_t **ErrorCode** (void)
  Retrieve the raw error code.

Detailed Description

Resource Manager non-fatal exception base class.

This base class provides basic information that travels with exceptions.

Definition at line 88 of file resmgr.svn-base.

Constructor & Destructor Documentation

QNXAPI::ResmgrException::ResmgrException (int err) [inline]

Build a new **ResmgrException**.

Parameters:

err standard error code.

Definition at line 101 of file resmgr.svn-base.QNXAPI::ResmgrException::ResmgrException (int err) [inline]

Build a new **ResmgrException**.

Parameters:

err standard error code.

Definition at line 101 of file resmgr.

Member Function Documentation

const char* QNXAPI::ResmgrException::ErrorDescription (void) [inline]

Retrieve a description of a standard error.

Returns:

cstring description of error.

Definition at line 110 of file resmgr.svn-base.const char* QNXAPI::ResmgrException::ErrorDescription (void) [inline]

Retrieve a description of a standard error.

Returns:

cstring description of error.

Definition at line 110 of file resmgr.

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

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

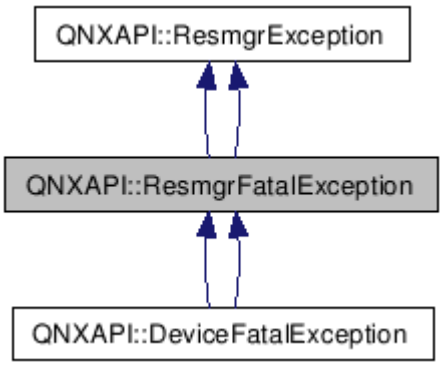
QNXAPI::ResmgrFatalException Class Reference

Resource Manager fatal exception base class.

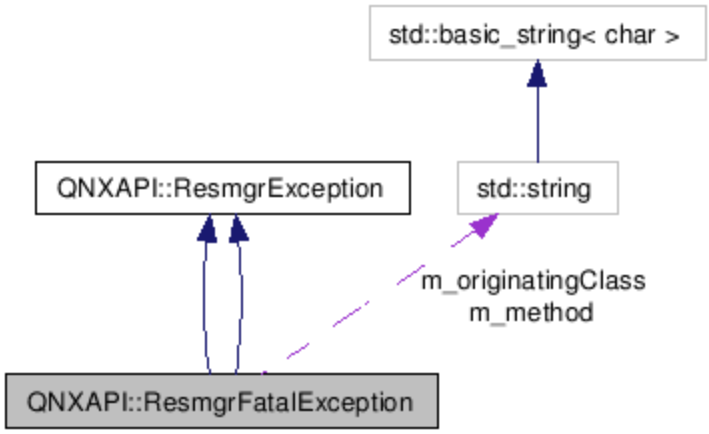
Inherits **QNXAPI::ResmgrException**, and **QNXAPI::ResmgrException**.

Inherited by **QNXAPI::DeviceFatalException**, and **QNXAPI::DeviceFatalException**.

Inheritance diagram for QNXAPI::ResmgrFatalException:



Collaboration diagram for QNXAPI::ResmgrFatalException:



Public Member Functions

- **ResmgrFatalException** (const char *func, int linenum, int err, const char *originatingClass="Resmgr")
  *Build a new **ResmgrFatalException**.*

- const std::string & **Class** (void)
  Retrieve the originating class.

- const std::string & **Method** (void)
  Retrieve the originating method.

- int **Line** (void)
  Retrieve the line number.

- **ResmgrFatalException** (const char *func, int linenum, int err, const char *originatingClass="Resmgr")
  *Build a new **ResmgrFatalException**.*

- const std::string & **Class** (void)
  Retrieve the originating class.

- const std::string & **Method** (void)
  Retrieve the originating method.

- int **Line** (void)
  Retrieve the line number.

Detailed Description

Resource Manager fatal exception base class.

This class provides information that travels with **Resmgr** fatal exceptions.

Definition at line 128 of file resmgr.svn-base.

Constructor & Destructor Documentation

QNXAPI::ResmgrFatalException::ResmgrFatalException (const char * func, int linenum, int err, const char * originatingClass = "Resmgr") [inline]

Build a new **ResmgrFatalException**.

Parameters:

func method that is calling constructor.
linenum line number of file.
err standard error code.
originatingClass class that threw exception (default: **Resmgr**)

Definition at line 147 of file resmgr.svn-base.QNXAPI::ResmgrFatalException::ResmgrFatalException (const char * func, int linenum, int err, const char * originatingClass = "Resmgr") [inline]

Build a new **ResmgrFatalException**.

Parameters:

func method that is calling constructor.
linenum line number of file.
err standard error code.
originatingClass class that threw exception (default: **Resmgr**)

Definition at line 147 of file resmgr.

Member Function Documentation

const std::string& QNXAPI::ResmgrFatalException::Class (void) [inline]

Retrieve the originating class.

Returns:

std::string containing originating class name.

Definition at line 160 of file resmgr.svn-base.const std::string& QNXAPI::ResmgrFatalException::Method (void) [inline]

Retrieve the originating method.

Returns:

std::string containing originating method name.

Definition at line 170 of file resmgr.svn-base.int QNXAPI::ResmgrFatalException::Line (void) [inline]

Retrieve the line number.

Returns:

std::int thats value is equal to the line number where exception originated

Definition at line 180 of file resmgr.svn-base.const std::string& QNXAPI::ResmgrFatalException::Class (void) [inline]

Retrieve the originating class.

Returns:

std::string containing originating class name.

Definition at line 160 of file resmgr.const std::string& QNXAPI::ResmgrFatalException::Method (void) [inline]

Retrieve the originating method.

Returns:

std::string containing originating method name.

Definition at line 170 of file resmgr.int QNXAPI::ResmgrFatalException::Line (void) [inline]

Retrieve the line number.

Returns:

std::int thats value is equal to the line number where exception originated

Definition at line 180 of file resmgr.

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

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

PosixAPI::Rootable Class Reference

Interface rootable.

Public Member Functions

- virtual void **Start** (int argc, char **argv)=0
- virtual void **Start** (int argc, char **argv)=0

Detailed Description

Interface rootable.

Interface for classes that are "rootable" (i.e. can be instantiated by the operating system).

Rootable classes, must not require constructor arguments (although they may have constructor arguments with default values). **Rootable** classes may have another entry point (typically OnRun) which will allow it to be used in a non-root configuration. When derived (via multiple inheritence) from **PosixAPI::Thread** the class can be entered directly from the O/S or via **PosixAPI::Thread::Start** called from another thread.

The overhead of making a class rootable is insignificant (it is only 1 virtual entry), so it is not unreasonable to make the majority of high functionality classes rootable.

Definition at line 37 of file interfaces.svn-base.

Member Function Documentation

virtual void PosixAPI::Rootable::Start (int* argc*, char *** argv*) [pure virtual]

Entry point to a **Rootable** class from the operating system.

Parameters:

argc number of arguments in argv.
argv vector of 'c' strings provided to class by the operating system.

virtual void PosixAPI::Rootable::Start (int* argc*, char *** argv*) [pure virtual]

Entry point to a **Rootable** class from the operating system.

Parameters:

argc number of arguments in argv.
argv vector of 'c' strings provided to class by the operating system.

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

- include/qfc/.svn/text-base/interfaces.svn-base
- include/qfc/interfaces

QNXAPI::RwLockAttr Class Reference

Read/Write lock attributes.

Public Member Functions

- **RwLockAttr** ()
- void **Shared** (bool shared)
- bool **Shared** (void) const
- pthread_rwlockattr_t * **operator &** (void)
- **~RwLockAttr** ()
- **RwLockAttr** ()
- void **Shared** (bool shared)
- bool **Shared** (void) const
- pthread_rwlockattr_t * **operator &** (void)
- **~RwLockAttr** ()

Detailed Description

Read/Write lock attributes.

Definition at line 141 of file lock.svn-base.

Constructor & Destructor Documentation

QNXAPI::RwLockAttr::RwLockAttr ()

Construct a mutex attribute with defaults.

QNXAPI::RwLockAttr::~RwLockAttr ()

Destroy the read/write lock attribute.

QNXAPI::RwLockAttr::RwLockAttr ()

Construct a mutex attribute with defaults.

QNXAPI::RwLockAttr::~RwLockAttr ()

Destroy the read/write lock attribute.

Member Function Documentation

void QNXAPI::RwLockAttr::Shared (bool* **shared)***

Set attributes process shared enable member.

Parameters:

shared boolean; true - enable shared, false - disable shared.

bool QNXAPI::RwLockAttr::Shared (void) const

Retrieve attributes shared setting.

Returns:

current shared setting (true - inter-process sharing; false - private).

pthread_rwlockattr_t* QNXAPI::RwLockAttr::operator & (void)

Retrieve a pointer to the encapsulated pthread_rwlockattr_t structure.

Returns:

pointer to pthread_rwlockattr_t structure.

void QNXAPI::RwLockAttr::Shared (bool* **shared)***

Set attributes process shared enable member.

Parameters:

shared boolean; true - enable shared, false - disable shared.

bool QNXAPI::RwLockAttr::Shared (void) const

Retrieve attributes shared setting.

Returns:

current shared setting (true - inter-process sharing; false - private).

pthread_rwlockattr_t* QNXAPI::RwLockAttr::operator & (void)

Retrieve a pointer to the encapsulated pthread_rwlockattr_t structure.

Returns:

pointer to pthread_rwlockattr_t structure.

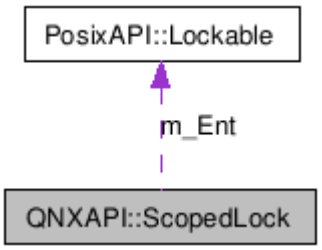
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/lock.svn-base
- include/qfc/lock

QNXAPI::ScopedLock Class Reference

Scoped lock (exclusive).

Collaboration diagram for QNXAPI::ScopedLock:



Public Member Functions

- **ScopedLock** (const **PosixAPI::Lockable** &ent)
- **~ScopedLock** ()
- **ScopedLock** (const **PosixAPI::Lockable** &ent)
- **~ScopedLock** ()

Detailed Description

Scoped lock (exclusive).

Can be used on any class that implements interface "Lockable".

Definition at line 69 of file lock.svn-base.

Constructor & Destructor Documentation

QNXAPI::ScopedLock::ScopedLock (const PosixAPI::Lockable & ent)

Construct a scoped lock, and lock the supplied entity.

Obtains an exclusive lock on the resource provided, and releases the lock when destroyed (i.e. goes out of scope).

Parameters:

ent entity to be locked.

QNXAPI::ScopedLock::~ScopedLock ()

Destroy a scoped lock, and unlock the supplied entity.

QNXAPI::ScopedLock::ScopedLock (const PosixAPI::Lockable & ent)

Construct a scoped lock, and lock the supplied entity.

Obtains an exclusive lock on the resource provided, and releases the lock when destroyed (i.e. goes out of scope).

Parameters:

ent entity to be locked.

QNXAPI::ScopedLock::~ScopedLock ()

Destroy a scoped lock, and unlock the supplied entity.

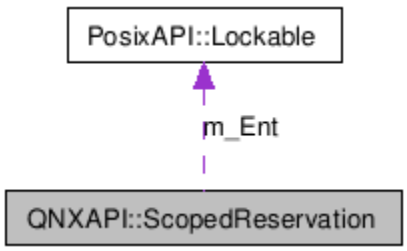
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/lock.svn-base
- include/qfc/lock

QNXAPI::ScopedReservation Class Reference

Scoped reservation (non-exclusive lock).

Collaboration diagram for QNXAPI::ScopedReservation:



Public Member Functions

- **ScopedReservation** (const **PosixAPI::Lockable** &ent)
  Construct a scoped lock, and lock the supplied entity.

- **~ScopedReservation** ()
- **ScopedReservation** (const **PosixAPI::Lockable** &ent)
  Construct a scoped lock, and lock the supplied entity.

- **~ScopedReservation** ()

Detailed Description

Scoped reservation (non-exclusive lock).

Can be used on any class that implements interface "Lockable".

Definition at line 96 of file lock.svn-base.

Constructor & Destructor Documentation

QNXAPI::ScopedReservation::ScopedReservation (const PosixAPI::Lockable & ent)

Construct a scoped lock, and lock the supplied entity.

Obtains a non-exclusive lock on the resource provided, and releases the lock when destroyed (i.e. goes out of scope).

Parameters:

ent entity to be locked.

QNXAPI::ScopedReservation::~ScopedReservation ()

Destroy a scoped lock, and unlock the supplied entity.

QNXAPI::ScopedReservation::ScopedReservation (const PosixAPI::Lockable & ent)

Construct a scoped lock, and lock the supplied entity.

Obtains a non-exclusive lock on the resource provided, and releases the lock when destroyed (i.e. goes out of scope).

Parameters:

ent entity to be locked.

QNXAPI::ScopedReservation::~ScopedReservation ()

Destroy a scoped lock, and unlock the supplied entity.

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

- include/qfc/.svn/text-base/lock.svn-base
- include/qfc/lock

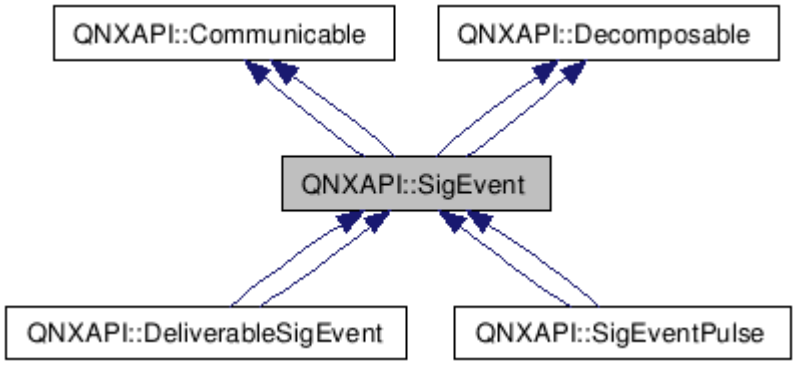
QNXAPI::SigEvent Class Reference

The **SigEvent** class.

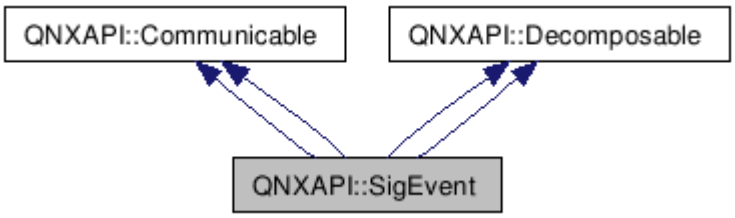
Inherits **QNXAPI::Communicable**, **QNXAPI::Decomposable**, **QNXAPI::Communicable**, and **QNXAPI::Decomposable**.

Inherited by **QNXAPI::DeliverableSigEvent**, **QNXAPI::DeliverableSigEvent**, **QNXAPI::SigEventPulse**, and **QNXAPI::SigEventPulse**.

Inheritance diagram for QNXAPI::SigEvent:



Collaboration diagram for QNXAPI::SigEvent:



Public Types

- enum **Notification**
  *Notification variants of a **SigEvent**.*

- enum **Notification**
  *Notification variants of a **SigEvent**.*

Public Member Functions

- **SigEvent** (void)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, SigNum sig)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, SigNum sig, **Value** val)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, SigNum sig, **Value** val, Code code)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, **Connection** con, Priority pri, Code code)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, **Connection** con, Priority pri, Code code, **Value** val)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, EntryPoint func, Priority pri, **Value** val, **PosixAPI::Thread::Attributes** attr)
  *Construct an instance of a **SigEvent** class (don't use).*

- **SigEvent** (const **SigEvent** &other)
  *Copy construct an instance of a **SigEvent** class.*

- virtual void * **Pod** (size_t &size) throw ()
  *Obtain the underlying 'C' representation (Plain Old Data) of the **SigEvent** (with size).*

- virtual void * **Pod** (void) throw ()
  *Obtain the underlying 'C' representation (Plain Old Data) of the **SigEvent**.*

- virtual iov_t * **Vect** (size_t &size) throw ()
  *Vectorize the **SigEvent**.*

- **SigEvent** (void)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, SigNum sig)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, SigNum sig, **Value** val)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, SigNum sig, **Value** val, Code code)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, **Connection** con, Priority pri, Code code)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, **Connection** con, Priority pri, Code code, **Value** val)
  *Construct an instance of a **SigEvent** class.*

- **SigEvent** (**Notification** notify, EntryPoint func, Priority pri, **Value** val, **PosixAPI::Thread::Attributes** attr)
  *Construct an instance of a **SigEvent** class (don't use).*

- **SigEvent** (const **SigEvent** &other)
  *Copy construct an instance of a **SigEvent** class.*

- virtual void * **Pod** (size_t &size) throw ()
  *Obtain the underlying 'C' representation (Plain Old Data) of the **SigEvent** (with size).*

- virtual void * **Pod** (void) throw ()
  *Obtain the underlying 'C' representation (Plain Old Data) of the **SigEvent**.*

- virtual iov_t * **Vect** (size_t &size) throw ()
  *Vectorize the **SigEvent**.*

Classes

- struct **Value**
  *Holds a value of a **SigEvent**.*

Detailed Description

The **SigEvent** class.

Implements Signal/Events.

Author:

rennieallen@gmail.com

Definition at line 39 of file sigevent.svn-base.

Constructor & Destructor Documentation

QNXAPI::SigEvent::SigEvent (Notification* notify*)

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify,** ***SigNum*** **sig)**

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
sig Signal number of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify,** ***SigNum*** **sig,** ***Value*** **val)**

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
sig Signal number of **SigEvent**
val **Value** of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify,** ***SigNum*** **sig,** ***Value*** **val,** ***Code*** **code)**

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
sig Signal number of **SigEvent**
val **Value** of **SigEvent**
code Code of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify,** ***Connection*** **con,** ***Priority*** **pri,** ***Code*** **code)**

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
con Connect to be associated with (serve as the delivery channel for) **SigEvent**
pri Priority to be associated with **SigEvent**
code Code of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify*****, Connection*** **con*****, Priority*** **pri*****, Code*** **code*****, Value*** **val*****)***

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
con Connect to be associated with (serve as the delivery channel for) **SigEvent**
pri Priority to be associated with **SigEvent**
code Code of **SigEvent**
val **Value** of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify*****, EntryPoint*** **func*****, Priority*** **pri*****, Value*** **val*****,***
PosixAPI::Thread::Attributes **attr*****)***

Construct an instance of a **SigEvent** class (don't use).

This is provided only for completeness, do not use this variant of **SigEvent**.

Parameters:

notify Notification variant of the constructed **SigEvent**
func Function to be run upon **SigEvent**
pri Priority to be associated with **SigEvent**
val **Value** of **SigEvent**
attr **Thread** attributes for **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (const SigEvent & **other*****) [inline]***

Copy construct an instance of a **SigEvent** class.

Parameters:

other Other **SigEvent** to copy from

Definition at line 188 of file sigevent.svn-base.QNXAPI::SigEvent::SigEvent (Notification **notify*****)***

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify,** ***SigNum*** **sig)**

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
sig Signal number of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify,** ***SigNum*** **sig,** ***Value*** **val)**

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
sig Signal number of **SigEvent**
val **Value** of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify,** ***SigNum*** **sig,** ***Value*** **val,** ***Code*** **code)**

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
sig Signal number of **SigEvent**
val **Value** of **SigEvent**
code Code of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify,** ***Connection*** **con,** ***Priority*** **pri,** ***Code*** **code)**

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
con Connect to be associated with (serve as the delivery channel for) **SigEvent**
pri Priority to be associated with **SigEvent**
code Code of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify*****, Connection*** **con*****, Priority*** **pri*****, Code*** **code*****, Value*** **val*****)***

Construct an instance of a **SigEvent** class.

Parameters:

notify Notification variant of the constructed **SigEvent**
con Connect to be associated with (serve as the delivery channel for) **SigEvent**
pri Priority to be associated with **SigEvent**
code Code of **SigEvent**
val **Value** of **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (Notification **notify*****, EntryPoint*** **func*****, Priority*** **pri*****, Value*** **val*****,***
PosixAPI::Thread::Attributes **attr*****)***

Construct an instance of a **SigEvent** class (don't use).

This is provided only for completeness, do not use this variant of **SigEvent**.

Parameters:

notify Notification variant of the constructed **SigEvent**
func Function to be run upon **SigEvent**
pri Priority to be associated with **SigEvent**
val **Value** of **SigEvent**
attr **Thread** attributes for **SigEvent**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

QNXAPI::SigEvent::SigEvent (const SigEvent & **other*****) [inline]***

Copy construct an instance of a **SigEvent** class.

Parameters:

other Other **SigEvent** to copy from

Definition at line 188 of file sigevent.

Member Function Documentation

virtual void* QNXAPI::SigEvent::Pod (size_t & size) throw () [inline, virtual]

Obtain the underlying 'C' representation (Plain Old Data) of the **SigEvent** (with size).

Parameters:

size Filled with size of underlying 'C' representation

Returns:

Pointer to the underlying 'C' representation

Implements **QNXAPI::Decomposable** (*p.58*).

Definition at line 198 of file sigevent.svn-base.virtual void* QNXAPI::SigEvent::Pod (void) throw () [inline, virtual]

Obtain the underlying 'C' representation (Plain Old Data) of the **SigEvent**.

Returns:

Pointer to the underlying 'C' representation

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Implements **QNXAPI::Decomposable** (*p.58*).

Definition at line 210 of file sigevent.svn-base.virtual iov_t* QNXAPI::SigEvent::Vect (size_t & size) throw () [inline, virtual]

Vectorize the **SigEvent**.

Parameters:

size Filled with size of the vector (number of elements)

Returns:

Pointer to an iov_t array

Implements **QNXAPI::Communicable** (*p.43*).

Definition at line 221 of file sigevent.svn-base.virtual void* QNXAPI::SigEvent::Pod (size_t & size) throw () [inline, virtual]

Obtain the underlying 'C' representation (Plain Old Data) of the **SigEvent** (with size).

Parameters:

size Filled with size of underlying 'C' representation

Returns:

Pointer to the underlying 'C' representation

Implements **QNXAPI::Decomposable** (*p.58*).

Definition at line 198 of file sigevent.virtual void* QNXAPI::SigEvent::Pod (void) throw () [inline, virtual]

Obtain the underlying 'C' representation (Plain Old Data) of the **SigEvent**.

Returns:

Pointer to the underlying 'C' representation

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Implements **QNXAPI::Decomposable** (*p.58*).

Definition at line 210 of file sigevent.virtual iov_t* QNXAPI::SigEvent::Vect (size_t & size) throw () [inline, virtual]

Vectorize the **SigEvent**.

Parameters:

size Filled with size of the vector (number of elements)

Returns:

Pointer to an iov_t array

Implements **QNXAPI::Communicable** (*p.43*).

Definition at line 221 of file sigevent.

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

- include/qfc/.svn/text-base/sigevent.svn-base
- include/qfc/sigevent

QNXAPI::SigEvent::Value Struct Reference

Holds a value of a **SigEvent**.

Public Member Functions

- **Value** (void)
  *Construct an instance of **Value** class for a **SigEvent**.*

- **Value** (int num)
  *Set the **Value** class for a **SigEvent**.*

- **Value** (void *ptr)
  *Set the **Value** class for a **SigEvent**.*

- **Value** (void)
  *Construct an instance of **Value** class for a **SigEvent**.*

- **Value** (int num)
  *Set the **Value** class for a **SigEvent**.*

- **Value** (void *ptr)
  *Set the **Value** class for a **SigEvent**.*

Detailed Description

Holds a value of a **SigEvent**.

Definition at line 45 of file sigevent.svn-base.

Constructor & Destructor Documentation

QNXAPI::SigEvent::Value::Value (int num) [inline]

Set the **Value** class for a **SigEvent**.

Parameters:

num Integer to set as the value

Definition at line 61 of file sigevent.svn-base.

References sival_int.QNXAPI::SigEvent::Value::Value (void * ptr) [inline]

Set the **Value** class for a **SigEvent**.

Parameters:

ptr Void pointer to set as the value

Definition at line 71 of file sigevent.svn-base.

References sival_ptr.QNXAPI::SigEvent::Value::Value (int num) [inline]

Set the **Value** class for a **SigEvent**.

Parameters:

num Integer to set as the value

Definition at line 61 of file sigevent.

References sival_int.QNXAPI::SigEvent::Value::Value (void * ptr) [inline]

Set the **Value** class for a **SigEvent**.

Parameters:

ptr Void pointer to set as the value

Definition at line 71 of file sigevent.

References sival_ptr.

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

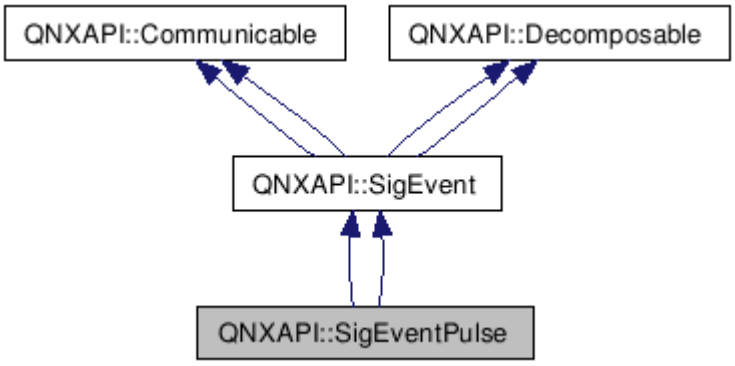
- include/qfc/.svn/text-base/sigevent.svn-base
- include/qfc/sigevent

QNXAPI::SigEventPulse Class Reference

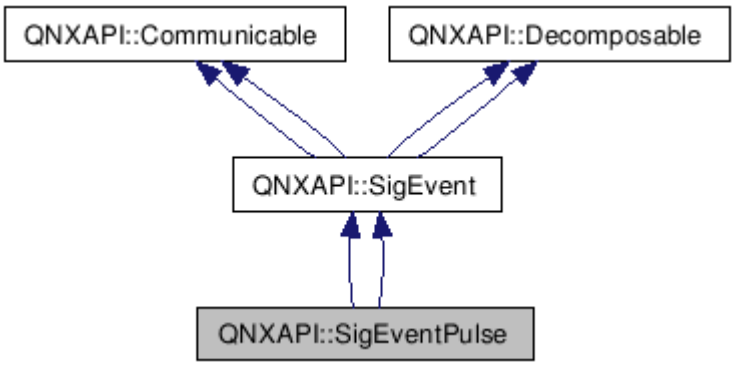
SigEvent pulse variant.

Inherits **QNXAPI::SigEvent**, and **QNXAPI::SigEvent**.

Inheritance diagram for QNXAPI::SigEventPulse:



Collaboration diagram for QNXAPI::SigEventPulse:



Public Member Functions

- **SigEventPulse** (**Connection** con, Priority pri, Code code)
  *Construct a pulse variant of a **SigEvent**.*

- **SigEventPulse** (**Connection** con, Priority pri, Code code, Value val)
  *Construct a pulse variant of a **SigEvent**.*

- **SigEventPulse** (**Connection** con, Priority pri, Code code)
  *Construct a pulse variant of a **SigEvent**.*

- **SigEventPulse** (**Connection** con, Priority pri, Code code, Value val)
  *Construct a pulse variant of a **SigEvent**.*

Detailed Description

SigEvent pulse variant.

SigEvent variant pulse

Author:

Rennie Allen `renniеallen@gmail.com`

Examples:

Cli/Cli.cpp.

Definition at line 317 of file sigevent.svn-base.

Constructor & Destructor Documentation

QNXAPI::SigEventPulse::SigEventPulse (Connection* con*, Priority* pri*, Code* code*) [inline]

Construct a pulse variant of a **SigEvent**.

Parameters:

con **Connection** to associated with this **SigEventPulse**
pri Priority to associated with this **SigEventPulse**
code Pulse code to associated with this **SigEventPulse**

Definition at line 327 of file sigevent.svn-base.QNXAPI::SigEventPulse::SigEventPulse (Connection* con*, Priority* pri*, Code* code*, Value* val*) [inline]

Construct a pulse variant of a **SigEvent**.

Parameters:

con **Connection** to associate with this **SigEventPulse**
pri Priority to associate with this **SigEventPulse**
code Pulse code to associate with this **SigEventPulse**
val Value to associate with this **SigEventPulse**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 340 of file sigevent.svn-base.QNXAPI::SigEventPulse::SigEventPulse (Connection* con*, Priority* pri*, Code* code*) [inline]

Construct a pulse variant of a **SigEvent**.

Parameters:

con **Connection** to associated with this **SigEventPulse**
pri Priority to associated with this **SigEventPulse**
code Pulse code to associated with this **SigEventPulse**

Definition at line 327 of file sigevent.QNXAPI::SigEventPulse::SigEventPulse (Connection* con*, Priority* pri*, Code* code*, Value* val*) [inline]

Construct a pulse variant of a **SigEvent**.

Parameters:

con **Connection** to associate with this **SigEventPulse**
pri Priority to associate with this **SigEventPulse**
code Pulse code to associate with this **SigEventPulse**
val Value to associate with this **SigEventPulse**

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 340 of file sigevent.

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

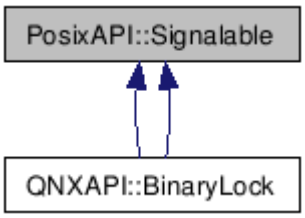
- include/qfc/.svn/text-base/sigevent.svn-base
- include/qfc/sigevent

PosixAPI::Signalable Class Reference

Interface: **Signalable**.

Inherited by **QNXAPI::BinaryLock**, and **QNXAPI::BinaryLock**.

Inheritance diagram for PosixAPI::Signalable:



Detailed Description

Interface: **Signalable**.

Definition at line 65 of file interfaces.svn-base.

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

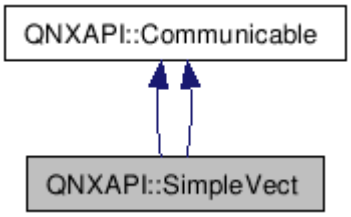
- include/qfc/.svn/text-base/interfaces.svn-base
- include/qfc/interfaces

QNXAPI::SimpleVect Class Reference

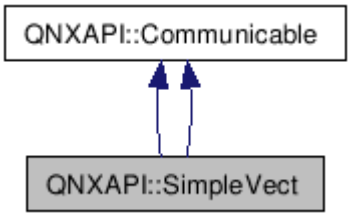
Simple vector class.

Inherits **QNXAPI::Communicable**, and **QNXAPI::Communicable**.

Inheritance diagram for QNXAPI::SimpleVect:



Collaboration diagram for QNXAPI::SimpleVect:



Public Member Functions

- **SimpleVect** (void *base, size_t len)
  *Construct an instance of a **SimpleVect** class.*

- virtual iov_t * **Vect** (size_t &size) throw ()
  Obtain pointer to underlying iov_t.

- **SimpleVect** (void *base, size_t len)
  *Construct an instance of a **SimpleVect** class.*

- virtual iov_t * **Vect** (size_t &size) throw ()
  Obtain pointer to underlying iov_t.

Detailed Description

Simple vector class.

Implements a simple unitary vector. Used to handle simple scalar types, or non visitor classes (

See also:

VectLoader).

Definition at line 41 of file ipc.svn-base.

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

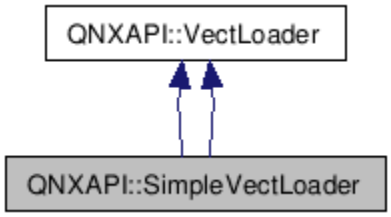
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

QNXAPI::SimpleVectLoader Class Reference

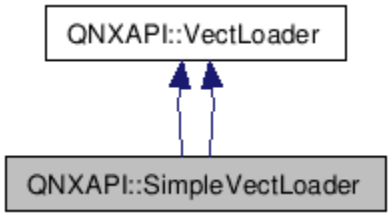
Default vector loader visitor (for **SimpleVect**).

Inherits **QNXAPI::VectLoader**, and **QNXAPI::VectLoader**.

Inheritance diagram for QNXAPI::SimpleVectLoader:



Collaboration diagram for QNXAPI::SimpleVectLoader:



Public Member Functions

- void **operator()** (**QNXAPI::SimpleVect** &v)
  *Load vector pointer from an instance of a **SimpleVect** class.*

- void **operator()** (**QNXAPI::SimpleVect** &v)
  *Load vector pointer from an instance of a **SimpleVect** class.*

Detailed Description

Default vector loader visitor (for **SimpleVect**).

Definition at line 99 of file ipc.svn-base.

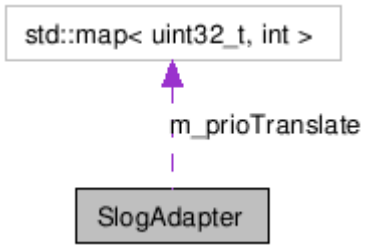
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

SlogAdapter Class Reference

The **SlogAdapter** class.

Collaboration diagram for SlogAdapter:



Public Member Functions

- **SlogAdapter** ()
  *Construct an instance of **SlogAdapter** class.*

- void **open** (const char *ident, **UnixAPI::Syslog::LogOption** opt, **UnixAPI::Syslog::Facility** fac) throw ()
  Open the slog interface (NULL operation).

- void **close** (void) throw ()
  Close the slog interface (NULL operation).

- void **log** (**UnixAPI::Syslog::Priority** prio, const char *c_str) throw ()
  *Log something to the **SlogAdapter**.*

- **SlogAdapter** ()
  *Construct an instance of **SlogAdapter** class.*

- void **open** (const char *ident, **UnixAPI::Syslog::LogOption** opt, **UnixAPI::Syslog::Facility** fac) throw ()
  Open the slog interface (NULL operation).

- void **close** (void) throw ()
  Close the slog interface (NULL operation).

- void **log** (**UnixAPI::Syslog::Priority** prio, const char *c_str) throw ()
  *Log something to the **SlogAdapter**.*

Detailed Description

The **SlogAdapter** class.

Implements an adapter to adapt "slog" style logging to iostreams.

Author:
`rennieallen@gmail.com`

Definition at line 34 of file slogadapt.svn-base.

Member Function Documentation

void SlogAdapter::open (const char * ident, UnixAPI::Syslog::LogOption opt, UnixAPI::Syslog::Facility fac) throw () [inline]

Open the slog interface (NULL operation).

Parameters:

ident Identification associated with this instance of **SlogAdapter**
opt Logging option
fac Facility

Definition at line 60 of file slogadapt.svn-base.void SlogAdapter::log (UnixAPI::Syslog::Priority prio, const char * c_str) throw () [inline]

Log something to the **SlogAdapter**.

Parameters:

prio Priority of log
c_str 'C' style string to be logged

Definition at line 78 of file slogadapt.svn-base.void SlogAdapter::open (const char * ident, UnixAPI::Syslog::LogOption opt, UnixAPI::Syslog::Facility fac) throw () [inline]

Open the slog interface (NULL operation).

Parameters:

ident Identification associated with this instance of **SlogAdapter**
opt Logging option
fac Facility

Definition at line 60 of file slogadapt.void SlogAdapter::log (UnixAPI::Syslog::Priority prio, const char * c_str) throw () [inline]

Log something to the **SlogAdapter**.

Parameters:

prio Priority of log
c_str 'C' style string to be logged

Definition at line 78 of file slogadapt.

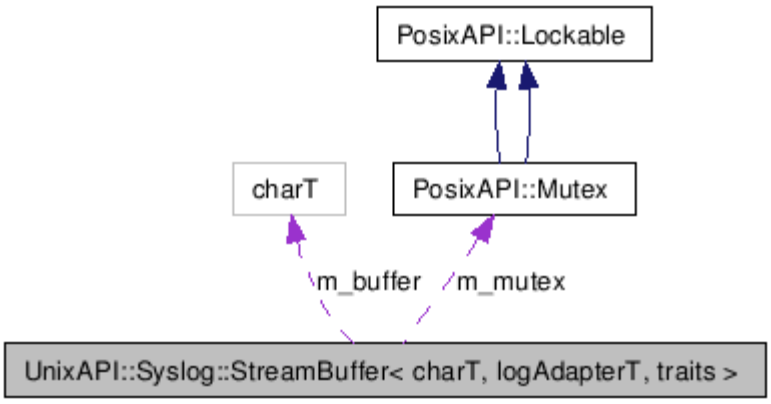
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/slogadapt.svn-base
- include/qfc/slogadapt

UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits > Class Template Reference

The **Syslog StreamBuffer** class.

Collaboration diagram for UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >:



Public Member Functions

- **StreamBuffer** ()
- **~StreamBuffer** ()
- void **open** (const char *ident, **LogOption** opt=NoDelay, **Facility** fac=User)
- virtual int **sync** ()
- void **pri** (**Priority** n)
- void **Lock** (void) throw (PosixAPI::MutexLockFailure)
- void **Unlock** (void) throw (PosixAPI::MutexUnlockFailure)
- **StreamBuffer** ()
- **~StreamBuffer** ()
- void **open** (const char *ident, **LogOption** opt=NoDelay, **Facility** fac=User)
- virtual int **sync** ()
- void **pri** (**Priority** n)
- void **Lock** (void) throw (PosixAPI::MutexLockFailure)
- void **Unlock** (void) throw (PosixAPI::MutexUnlockFailure)

Detailed Description

template<class charT, class logAdapterT, class traits = std::char_traits<charT>> class UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >

The **Syslog StreamBuffer** class.

This class adapts the stream buffer to a logging adapter. Data from the stream buffer is supplied to the logging system adapter. By default at least two logging system adapters are provided:

SyslogAdapter (syslogd). **SlogAdapter** (slogger).

This is a template and the character class and log adapter class must supplied.

Author:

rennieallen@gmail.com

Definition at line 58 of file syslog.svn-base.

Constructor & Destructor Documentation

template<class charT, class logAdapterT, class traits = std::char_traits<charT>> UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::StreamBuffer () [inline]

Construct a **Syslog::StreamBuffer**.

Sets the buffer pointer to the beginning of the buffer.

Definition at line 67 of file syslog.svn-base.

References UnixAPI::Syslog::BufferSize.template<class charT, class logAdapterT, class traits = std::char_traits<charT>> UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::~StreamBuffer () [inline]

Destroy a **Syslog::StreamBuffer**.

Calls the logging adapters close method.

Definition at line 77 of file syslog.svn-base.template<class charT, class logAdapterT, class traits = std::char_traits<charT>> UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::StreamBuffer () [inline]

Construct a **Syslog::StreamBuffer**.

Sets the buffer pointer to the beginning of the buffer.

Definition at line 67 of file syslog.

References UnixAPI::Syslog::BufferSize.template<class charT, class logAdapterT, class traits = std::char_traits<charT>> UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::~StreamBuffer () [inline]

Destroy a **Syslog::StreamBuffer**.

Calls the logging adapters close method.

Definition at line 77 of file syslog.

Member Function Documentation

template<class charT, class logAdapterT, class traits = std::char_traits<charT>> void UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::open (const char * ident, LogOption opt = NoDelay, Facility fac = User) [inline]

Open the underlying logging adapter.

Parameters:

ident 'C' style string identifying for this instance (not supported by all underyling logging adapters).
opt Logging option (see **UnixAPI::Syslog::LogOption**).
fac (see **UnixAPI::Syslog::Facility**).

Definition at line 92 of file syslog.svn-base.template<class charT, class logAdapterT, class traits = std::char_traits<charT>> virtual int UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::sync () [inline, virtual]

Sync the **StreamBuffer** with the underlying logging adapter.

Calls the logging adapters log member; supplies priority and pointer to **StreamBuffer**.

Definition at line 105 of file syslog.svn-base.

References UnixAPI::Syslog::BufferSize.template<class charT, class logAdapterT, class traits = std::char_traits<charT>> void UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::pri (Priority n) [inline]

Set the priority member

Definition at line 118 of file syslog.svn-base.template<class charT, class logAdapterT, class traits = std::char_traits<charT>> void UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::Lock (void) throw (PosixAPI::MutexLockFailure) [inline]

Lock the **StreamBuffer**.

Definition at line 126 of file syslog.svn-base.

References PosixAPI::Mutex::Lock().

Here is the call graph for this function:

template<class charT, class logAdapterT, class traits = std::char_traits<charT>> void UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::Unlock (void) throw (PosixAPI::MutexUnlockFailure) [inline]

Unlock the **StreamBuffer**.

Definition at line 134 of file syslog.svn-base.

References PosixAPI::Mutex::Unlock().

Here is the call graph for this function:

template<class charT, class logAdapterT, class traits = std::char_traits<charT>> void UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::open (const char * ident, LogOption opt = NoDelay, Facility fac = User) [inline]

Open the underlying logging adapter.

Parameters:

ident 'C' style string identifying for this instance (not supported by all underyling logging adapters).
opt Logging option (see **UnixAPI::Syslog::LogOption**).
fac (see **UnixAPI::Syslog::Facility**).

Definition at line 92 of file syslog.template<class charT, class logAdapterT, class traits = std::char_traits<charT>> virtual int UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::sync () [inline, virtual]

Sync the **StreamBuffer** with the underlying logging adapter.

Calls the logging adapters log member; supplies priority and pointer to **StreamBuffer**.

Definition at line 105 of file syslog.

References UnixAPI::Syslog::BufferSize.template<class charT, class logAdapterT, class traits = std::char_traits<charT>> void UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::pri (Priority n) [inline]

Set the priority member

Definition at line 118 of file syslog.template<class charT, class logAdapterT, class traits = std::char_traits<charT>> void UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::Lock (void) throw (PosixAPI::MutexLockFailure) [inline]

Lock the **StreamBuffer**.

Definition at line 126 of file syslog.

References PosixAPI::Mutex::Lock().

Here is the call graph for this function:

template<class charT, class logAdapterT, class traits = std::char_traits<charT>> void UnixAPI::Syslog::StreamBuffer< charT, logAdapterT, traits >::Unlock (void) throw (PosixAPI::MutexUnlockFailure) [inline]

Unlock the **StreamBuffer**.

Definition at line 134 of file syslog.

References PosixAPI::Mutex::Unlock().

Here is the call graph for this function:

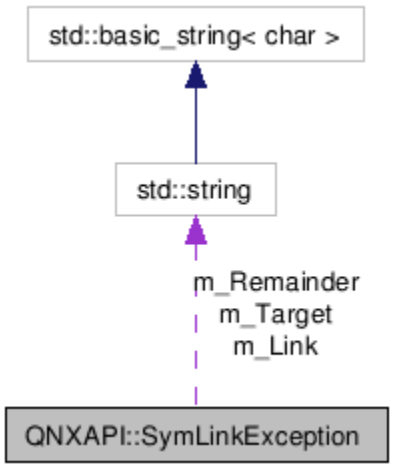
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/syslog.svn-base
- include/qfc/syslog

QNXAPI::SymLinkException Class Reference

SymLinkException (thrown when a redirect is encountered deep in a path).

Collaboration diagram for QNXAPI::SymLinkException:



Public Member Functions

- **SymLinkException** (std::string &link, std::string &target, std::string &remainder)
  *Constuct an instance of the **SymLinkException** class.*

- **SymLinkException** (std::string &link, char *target, std::string &remainder)
  *Constuct an instance of the **SymLinkException** class.*

- std::string & **Link** (void)
  Get a string representing the link.

- std::string & **Target** (void)
  Get a string representing the target.

- std::string & **Remainder** (void)
  Get a string representing the remainder.

- **SymLinkException** (std::string &link, std::string &target, std::string &remainder)
  *Constuct an instance of the **SymLinkException** class.*

- **SymLinkException** (std::string &link, char *target, std::string &remainder)
  *Constuct an instance of the **SymLinkException** class.*

- std::string & **Link** (void)
  Get a string representing the link.

- std::string & **Target** (void)
  Get a string representing the target.

- std::string & **Remainder** (void)
  Get a string representing the remainder.

Detailed Description

SymLinkException (thrown when a redirect is encountered deep in a path).

SymLinkException is a special exception. It is not used to report an error, but a redirection. While I realize this is an abuse of exceptions, I believe that it's value in simplying the code outweighs the abuse.

Definition at line 237 of file resmgr.svn-base.

Constructor & Destructor Documentation

QNXAPI::SymLinkException::SymLinkException (std::string &* link*, std::string &* target*, std::string &* remainder*) [inline]

Constuct an instance of the **SymLinkException** class.

Parameters:

link
target
remainder

Definition at line 247 of file resmgr.svn-base.QNXAPI::SymLinkException::SymLinkException (std::string & link, char * target, std::string & remainder) [inline]

Constuct an instance of the **SymLinkException** class.

Parameters:

link
target
remainder

Definition at line 260 of file resmgr.svn-base.QNXAPI::SymLinkException::SymLinkException (std::string & link, std::string & target, std::string & remainder) [inline]

Constuct an instance of the **SymLinkException** class.

Parameters:

link
target
remainder

Definition at line 247 of file resmgr.QNXAPI::SymLinkException::SymLinkException (std::string & link, char * target, std::string & remainder) [inline]

Constuct an instance of the **SymLinkException** class.

Parameters:

link
target
remainder

Definition at line 260 of file resmgr.

Member Function Documentation

std::string& QNXAPI::SymLinkException::Link (void) [inline]

Get a string representing the link.

Returns:

a reference to std::string representing the link.

Definition at line 272 of file resmgr.svn-base.std::string& QNXAPI::SymLinkException::Target (void) [inline]

Get a string representing the target.

Returns:

a reference to std::string representing the target.

Definition at line 282 of file resmgr.svn-base.std::string& QNXAPI::SymLinkException::Remainder (void) [inline]

Get a string representing the remainder.

Returns:

a reference to std::string representing the remainder.

Definition at line 292 of file resmgr.svn-base.std::string& QNXAPI::SymLinkException::Link (void) [inline]

Get a string representing the link.

Returns:

a reference to std::string representing the link.

Definition at line 272 of file resmgr.std::string& QNXAPI::SymLinkException::Target (void) [inline]

Get a string representing the target.

Returns:

a reference to std::string representing the target.

Definition at line 282 of file resmgr.std::string& QNXAPI::SymLinkException::Remainder (void) [inline]

Get a string representing the remainder.

Returns:

a reference to std::string representing the remainder.

Definition at line 292 of file resmgr.

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

- include/qfc/.svn/text-base/resmgr.svn-base
- include/qfc/resmgr

SyslogAdapter Class Reference

The **SyslogAdapter** class.

Public Member Functions

- **SyslogAdapter** ()
  *Construct an instance of **SyslogAdapter** class.*

- void **open** (const char *ident, **UnixAPI::Syslog::LogOption** opt, **UnixAPI::Syslog::Facility** fac)
  Open the slog interface (NULL operation).

- void **close** (void)
  Close the slog interface (NULL operation).

- void **log** (int prio, const char *c_str)
  *Log something to the **SyslogAdapter**.*

- **SyslogAdapter** ()
  *Construct an instance of **SyslogAdapter** class.*

- void **open** (const char *ident, **UnixAPI::Syslog::LogOption** opt, **UnixAPI::Syslog::Facility** fac)
  Open the slog interface (NULL operation).

- void **close** (void)
  Close the slog interface (NULL operation).

- void **log** (int prio, const char *c_str)
  *Log something to the **SyslogAdapter**.*

Detailed Description

The **SyslogAdapter** class.

Implements an adapter to adapt "syslog" style logging to iostreams.

Author:

rennieallen@gmail.com

Definition at line 27 of file syslogadapt.svn-base.

Member Function Documentation

void SyslogAdapter::open (const char * ident, UnixAPI::Syslog::LogOption opt, UnixAPI::Syslog::Facility fac) [inline]

Open the slog interface (NULL operation).

Parameters:

ident Identification associated with this instance of **SyslogAdapter**
opt Logging option
fac Facility

Definition at line 44 of file syslogadapt.svn-base.void SyslogAdapter::log (int prio, const char * c_str) [inline]

Log something to the **SyslogAdapter**.

Parameters:

prio Priority of log
c_str 'C' style string to be logged

Definition at line 63 of file syslogadapt.svn-base.void SyslogAdapter::open (const char * ident, UnixAPI::Syslog::LogOption opt, UnixAPI::Syslog::Facility fac) [inline]

Open the slog interface (NULL operation).

Parameters:

ident Identification associated with this instance of **SyslogAdapter**

opt Logging option
fac Facility

Definition at line 44 of file syslogadapt.void SyslogAdapter::log (int prio, const char * c_str) [inline]

Log something to the **SyslogAdapter**.

Parameters:

prio Priority of log
c_str 'C' style string to be logged

Definition at line 63 of file syslogadapt.

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

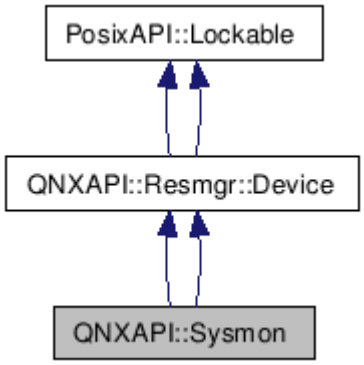
- include/qfc/.svn/text-base/syslogadapt.svn-base
- include/qfc/syslogadapt

QNXAPI::Sysmon Class Reference

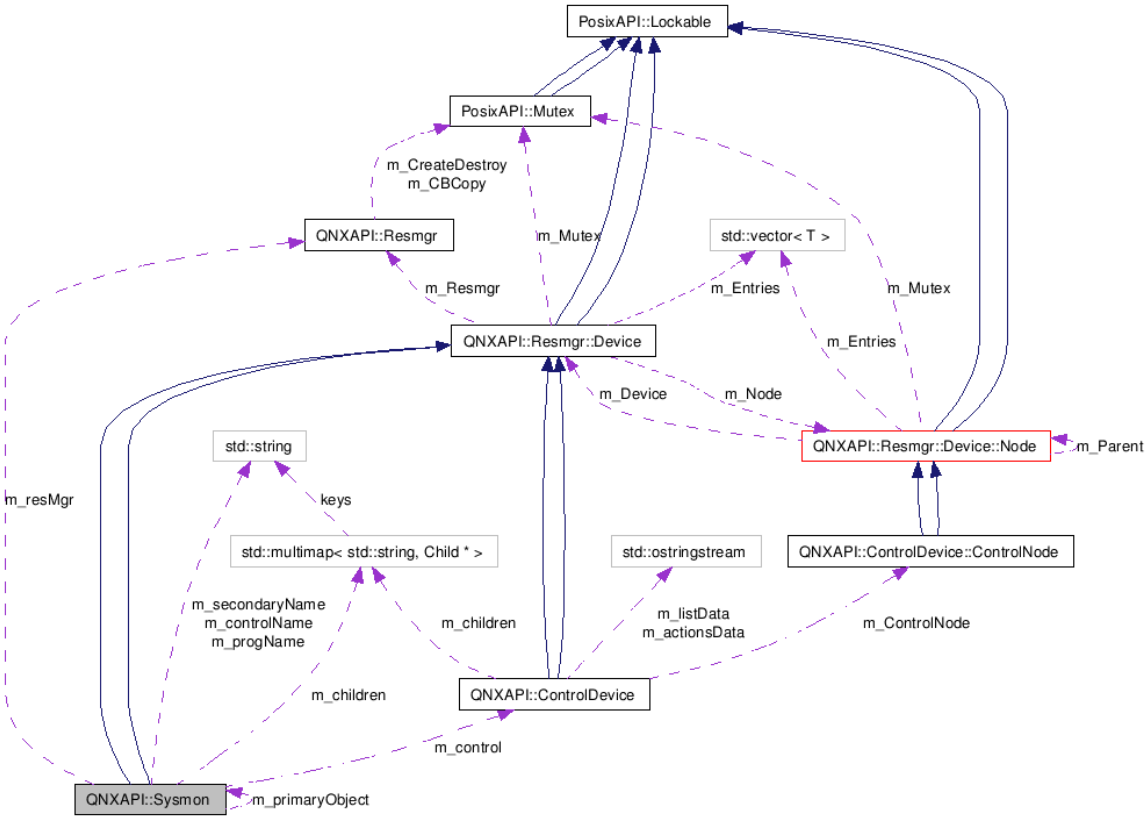
Provides a fault tolerant process manager.

Inherits **QNXAPI::Resmgr::Device**, and **QNXAPI::Resmgr::Device**.

Inheritance diagram for QNXAPI::Sysmon:



Collaboration diagram for QNXAPI::Sysmon:



Public Member Functions

- **Sysmon** (const char *progName, const char *guardianName="/dev/guardian")
  Construct a system monitor.

- void **addChild** (const char *pathName, std::list< const char * > &args, pid_t pid, bool responsible=false)
  Add a child process to the system monitor.

- virtual int **Open** (resmgr_context_t *ctp, io_open_t *msg, void *extra) throw ()
  Open a resource.

- virtual int **CloseOcb** (resmgr_context_t *ctp, void *msg, **QNXAPI::Resmgr::Ocb** ***ocb**) throw ()
  Close an Open Control Block.

- **Sysmon** (const char *progName, const char *guardianName="/dev/guardian")
  Construct a system monitor.

- void **addChild** (const char *pathName, std::list< const char * > &args, pid_t pid, bool responsible=false)
  Add a child process to the system monitor.

- virtual int **Open** (resmgr_context_t *ctp, io_open_t *msg, void *extra) throw ()
  Open a resource.

- virtual int **CloseOcb** (resmgr_context_t *ctp, void *msg, **QNXAPI::Resmgr::Ocb** ***ocb**) throw ()
  Close an Open Control Block.

Detailed Description

Provides a fault tolerant process manager.

To use this class, simply instantiate, populate with children processes, and call the monitor method.

Warning:

Sysmon class calls fork, so the calling thread must be the main thread, and this thread must not have created any other threads prior to instantiating an instance of this class.

Author:

rennieallen@gmail.com

Definition at line 122 of file sysmon.svn-base.

Constructor & Destructor Documentation

QNXAPI::Sysmon::Sysmon (const char * progName, const char * guardianName = "/dev/guardian")

Construct a system monitor.

Parameters:

progName name of program instantiating this class
guardianName name of prefix to be registered by the backup

QNXAPI::Sysmon::Sysmon (const char * progName, const char * guardianName = "/dev/guardian")

Construct a system monitor.

Parameters:

progName name of program instantiating this class
guardianName name of prefix to be registered by the backup

Member Function Documentation

void QNXAPI::Sysmon::addChild (const char * pathName, std::list< const char * > & args, pid_t pid, bool responsible = false)

Add a child process to the system monitor.

Parameters:

pathName Name of the file containing an executable process image
args Arguments to be supplied to the process on start
pid Process ID
responsible Flag to indicate if this child is the responsible process (used internally only)

virtual int QNXAPI::Sysmon::Open (resmgr_context_t * ctp, io_open_t * msg, void * extra) throw () [virtual]

Open a resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library open message.
extra 'C' resmgr library "extra" information for open.

Returns:

small positive integer from errno.h

Reimplemented from QNXAPI::Resmgr::Device (p.254).virtual int QNXAPI::Sysmon::CloseOcb (resmgr_context_t * ctp, void * msg, QNXAPI::Resmgr::Ocb * ocb) throw () [virtual]

Close an Open Control Block.

Parameters:

ctp 'C' resmgr library context pointer.
msg void pointer (not used).
ocb Open Control Block to close.

Returns:

small positive integer from errno.h

Reimplemented from QNXAPI::Resmgr::Device (p.257).void QNXAPI::Sysmon::addChild (const char * pathName, std::list< const char * > & args, pid_t pid, bool responsible = false)

Add a child process to the system monitor.

Parameters:

pathName Name of the file containing an executable process image
args Arguments to be supplied to the process on start
pid Process ID
responsible Flag to indicate if this child is the responsible process (used internally only)

virtual int QNXAPI::Sysmon::Open (resmgr_context_t * ctp, io_open_t * msg, void * extra) throw () [virtual]

Open a resource.

Parameters:

ctp 'C' resmgr library context pointer.
msg 'C' resmgr library open message.
extra 'C' resmgr library "extra" information for open.

Returns:

small positive integer from errno.h

Reimplemented from QNXAPI::Resmgr::Device (p.254).virtual int QNXAPI::Sysmon::CloseOcb (resmgr_context_t * ctp, void * msg, QNXAPI::Resmgr::Ocb * ocb) throw () [virtual]

Close an Open Control Block.

Parameters:

ctp 'C' resmgr library context pointer.
msg void pointer (not used).
ocb Open Control Block to close.

Returns:

small positive integer from errno.h

Reimplemented from **QNXAPI::Resmgr::Device** (*p.257*).

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

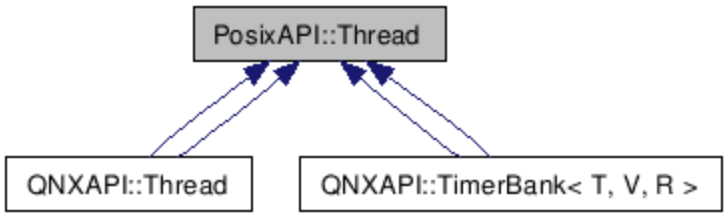
- include/qfc/.svn/text-base/sysmon.svn-base
- include/qfc/sysmon

PosixAPI::Thread Class Reference

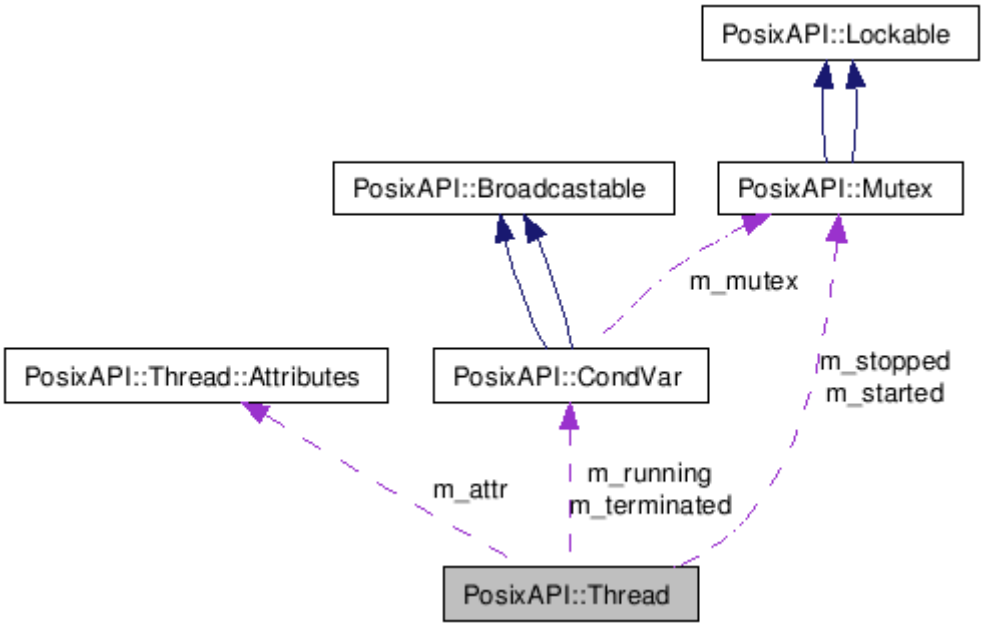
Thread class.

Inherited by **QNXAPI::Thread**, **QNXAPI::Thread**, **QNXAPI::TimerBank< T, V, R >**, and **QNXAPI::TimerBank< T, V, R >**.

Inheritance diagram for PosixAPI::Thread:



Collaboration diagram for PosixAPI::Thread:



Public Member Functions

- **Thread** ()
- **Thread** (**Attributes** &attr)
- virtual **~Thread** ()
- void **Start** (void)
- pthread_t **Id** (void)
- void **WaitForStart** (uint64_t to=UINT64_MAX)
- void **Stop** (void)
- bool **ShouldStop** (void)
- void **WaitForStop** (uint64_t to=UINT64_MAX)
- void **Detach** (void)
- void **Join** (void *&ptr, uint64_t to=UINT64_MAX)
- void **Cancel** (void)
- **Thread** ()
- **Thread** (**Attributes** &attr)
- virtual **~Thread** ()
- void **Start** (void)
- pthread_t **Id** (void)
- void **WaitForStart** (uint64_t to=UINT64_MAX)
- void **Stop** (void)

- bool **ShouldStop** (void)
- void **WaitForStop** (uint64_t to=UINT64_MAX)
- void **Detach** (void)
- void **Join** (void *&ptr, uint64_t to=UINT64_MAX)
- void **Cancel** (void)

Classes

- class **Attributes**
  ***Thread** attributes class.*

Detailed Description

Thread class.

Encapsulates basic thread functionality.

Author:
`rennieallen@gmail.com`

Definition at line 70 of file thread.svn-base.

Constructor & Destructor Documentation

PosixAPI::Thread::Thread ()

Create a thread.

PosixAPI::Thread::Thread (Attributes & attr)

Create a thread (supplying attributes).

Parameters:
attr thread attributes.

virtual PosixAPI::Thread::~Thread () [virtual]

Destroy a thread.

PosixAPI::Thread::Thread ()

Create a thread.

PosixAPI::Thread::Thread (Attributes & attr)

Create a thread (supplying attributes).

Parameters:
attr thread attributes.

virtual PosixAPI::Thread::~Thread () [virtual]

Destroy a thread.

Member Function Documentation

void PosixAPI::Thread::Start (void)

Begin scheduling a thread.

pthread_t PosixAPI::Thread::Id (void) [inline]

Get the threads id.

Returns:

thread id.

Definition at line 204 of file thread.svn-base.void PosixAPI::Thread::WaitForStart (uint64_t to = UINT64_MAX)

Wait for a thread to begin being scheduled.

Parameters:

to timeout to wait for thread to begin.

void PosixAPI::Thread::Stop (void)

Stop thread from being scheduled.

bool PosixAPI::Thread::ShouldStop (void) [inline]

Determine if thread should continue.

Returns:

bool indicating whether thread should stop (true) or not (false).

Definition at line 226 of file thread.svn-base.

Referenced by QNXAPI::TimerBank< T, V, R >::OnRun().void PosixAPI::Thread::WaitForStop (uint64_t to = UINT64_MAX)

Wait for a thread to stop being scheduled.

Parameters:

to timeout to wait for thread to end.

void PosixAPI::Thread::Detach (void)

Detach thread.

void PosixAPI::Thread::Join (void *& ptr, uint64_t to = UINT64_MAX)

Join the thread.

Parameters:

ptr pointer to a pointer to void to be filled with threads exit.

to timeout to wait for thread to end.

void PosixAPI::Thread::Cancel (void)

Cancel the thread.

void PosixAPI::Thread::Start (void)

Begin scheduling a thread.

pthread_t PosixAPI::Thread::Id (void) [inline]

Get the threads id.

Returns:

thread id.

Definition at line 204 of file thread.void PosixAPI::Thread::WaitForStart (uint64_t to = UINT64_MAX)

Wait for a thread to begin being scheduled.

Parameters:

to timeout to wait for thread to begin.

void PosixAPI::Thread::Stop (void)

Stop thread from being scheduled.

bool PosixAPI::Thread::ShouldStop (void) [inline]

Determine if thread should continue.

Returns:

bool indicating whether thread should stop (true) or not (false).

Definition at line 226 of file thread.void PosixAPI::Thread::WaitForStop (uint64_t to = UINT64_MAX)

Wait for a thread to stop being scheduled.

Parameters:

to timeout to wait for thread to end.

void PosixAPI::Thread::Detach (void)

Detach thread.

void PosixAPI::Thread::Join (void *& ptr, uint64_t to = UINT64_MAX)

Join the thread.

Parameters:

ptr pointer to a pointer to void to be filled with threads exit.
to timeout to wait for thread to end.

void PosixAPI::Thread::Cancel (void)

Cancel the thread.

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

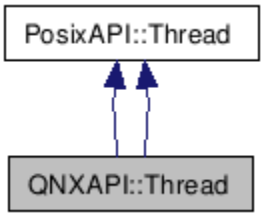
- include/qfc/.svn/text-base/thread.svn-base
- include/qfc/thread

QNXAPI::Thread Class Reference

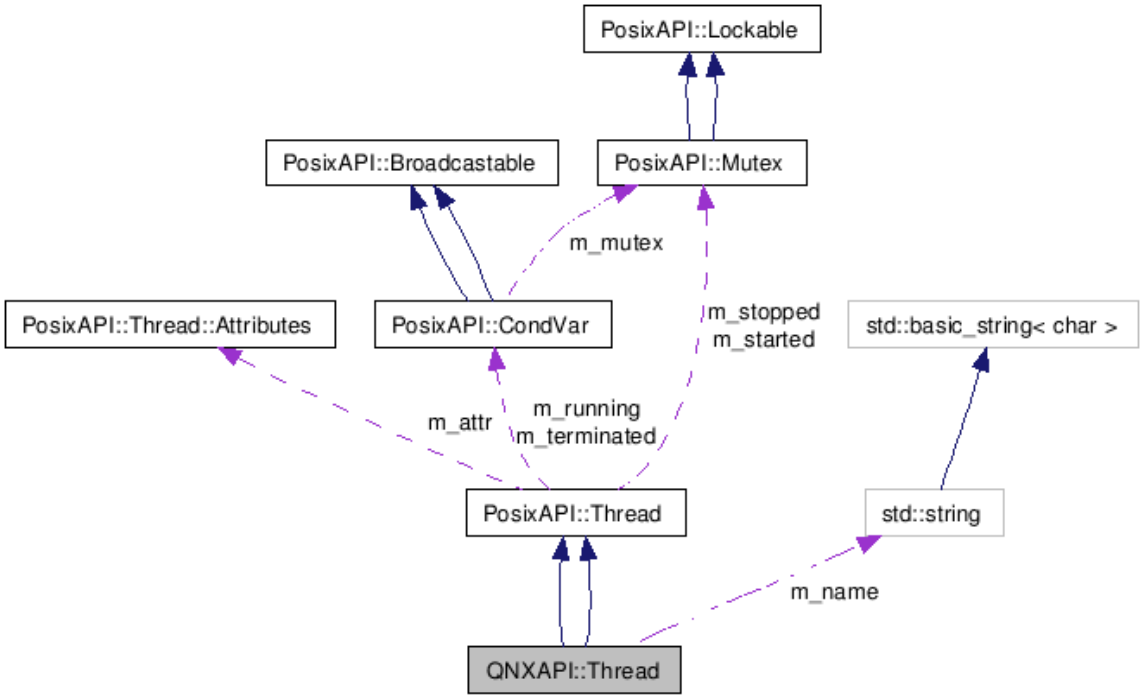
Thread class - QNX extensions.

Inherits **PosixAPI::Thread**, and **PosixAPI::Thread**.

Inheritance diagram for QNXAPI::Thread:



Collaboration diagram for QNXAPI::Thread:



Public Member Functions

- const std::string & **Name** (void) const throw ()
  Get the threads name.

- void **Name** (std::string &name)
  Name the thread.

- void **Name** (const char *name)
  Name the thread.

- const std::string & **Name** (void) const throw ()
  Get the threads name.

- void **Name** (std::string &name)
  Name the thread.

- void **Name** (const char *name)
  Name the thread.

Detailed Description

Thread class - QNX extensions.

Adds non-Posix naming.

Author:

rennieallen@gmail.com

Definition at line 288 of file thread.svn-base.

Member Function Documentation

const std::string& QNXAPI::Thread::Name (void) const throw ()

Get the threads name.

Returns:

a const std::string reference to the thread name

void QNXAPI::Thread::Name (std::string &* name*)

Name the thread.

Parameters:

name a std::string holding the threads new name.

void QNXAPI::Thread::Name (const char * name*)***

Name the thread.

Parameters:

name a 'c' string pointer to memory holding the threads new name.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

const std::string& QNXAPI::Thread::Name (void) const throw ()

Get the threads name.

Returns:

a const std::string reference to the thread name

void QNXAPI::Thread::Name (std::string &* name*)

Name the thread.

Parameters:

name a std::string holding the threads new name.

void QNXAPI::Thread::Name (const char * name*)***

Name the thread.

Parameters:

name a 'c' string pointer to memory holding the threads new name.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

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

- include/qfc/.svn/text-base/thread.svn-base
- include/qfc/thread

PosixAPI::Thread::Attributes Class Reference

Thread attributes class.

Public Types

- enum **DetachStateType**
  Thread *detach state types.*

- enum **SchedPolicyType**
  Thread *scheduling policy types.*

- enum **ScopeType**
  Thread *scope types.*

- enum **DetachStateType**
  Thread *detach state types.*

- enum **SchedPolicyType**
  Thread *scheduling policy types.*

- enum **ScopeType**
  Thread *scope types.*

Public Member Functions

- **Attributes** (**DetachStateType** ds=Joinable, **SchedPolicyType** sp=Inherit, **ScopeType** scope=System)
- void **DetachState** (**DetachStateType** ds)
- **DetachStateType DetachState** (void)

- void **SchedPolicy** (**SchedPolicyType** sp)
- **SchedPolicyType SchedPolicy** (void)
- void **Scope** (**ScopeType** scope)
- **ScopeType Scope** (void)
- pthread_attr_t & **Impl** (void)
- **Attributes** (**DetachStateType** ds=Joinable, **SchedPolicyType** sp=Inherit, **ScopeType** scope=System)
- void **DetachState** (**DetachStateType** ds)
- **DetachStateType DetachState** (void)
- void **SchedPolicy** (**SchedPolicyType** sp)
- **SchedPolicyType SchedPolicy** (void)
- void **Scope** (**ScopeType** scope)
- **ScopeType Scope** (void)
- pthread_attr_t & **Impl** (void)

Detailed Description

Thread attributes class.

Encapsulates thread attributes.

Author:
rennieallen@gmail.com

Definition at line 80 of file thread.svn-base.

Constructor & Destructor Documentation

PosixAPI::Thread::Attributes::Attributes (DetachStateType* ds = *Joinable, SchedPolicyType* sp = *Inherit, ScopeType* scope = *System)

Create thread attributes.

Parameters:

ds detach state.
sp scheduling policy.
scope scheduling scope.

PosixAPI::Thread::Attributes::Attributes (DetachStateType* ds = *Joinable, SchedPolicyType* sp = *Inherit, ScopeType* scope = *System)

Create thread attributes.

Parameters:

ds detach state.
sp scheduling policy.
scope scheduling scope.

Member Function Documentation

void PosixAPI::Thread::Attributes::DetachState (DetachStateType* ds*)

Set the thread detach state.

Parameters:

ds detach state.

DetachStateType PosixAPI::Thread::Attributes::DetachState (void)

Get the thread detach state.

Returns:

detach state.

void PosixAPI::Thread::Attributes::SchedPolicy (SchedPolicyType* sp*)

Set the thread scheduling policy.

Parameters:

sp scheduling policy.

SchedPolicyType PosixAPI::Thread::Attributes::SchedPolicy (void)

Get the thread scheduling policy.

Returns:

scheduling policy.

void PosixAPI::Thread::Attributes::Scope (ScopeType* scope*)

Set the thread scheduling scope.

Parameters:

scope scheduling scope.

ScopeType PosixAPI::Thread::Attributes::Scope (void)

Get the thread scheduling scope.

Returns:

scheduling scope.

pthread_attr_t& PosixAPI::Thread::Attributes::Impl (void) [inline]

Get the underlying 'C' implementation of the thread attributes.

Returns:

'C' thread attributes.

Definition at line 168 of file thread.svn-base.void PosixAPI::Thread::Attributes::DetachState (DetachStateType* ds*)

Set the thread detach state.

Parameters:

ds detach state.

DetachStateType PosixAPI::Thread::Attributes::DetachState (void)

Get the thread detach state.

Returns:
detach state.

void PosixAPI::Thread::Attributes::SchedPolicy (SchedPolicyType* sp*)

Set the thread scheduling policy.

Parameters:
sp scheduling policy.

SchedPolicyType PosixAPI::Thread::Attributes::SchedPolicy (void)

Get the thread scheduling policy.

Returns:
scheduling policy.

void PosixAPI::Thread::Attributes::Scope (ScopeType* scope*)

Set the thread scheduling scope.

Parameters:
scope scheduling scope.

ScopeType PosixAPI::Thread::Attributes::Scope (void)

Get the thread scheduling scope.

Returns:
scheduling scope.

pthread_attr_t& PosixAPI::Thread::Attributes::Impl (void) [inline]

Get the underlying 'C' implementation of the thread attributes.

Returns:
'C' thread attributes.

Definition at line 168 of file thread.

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

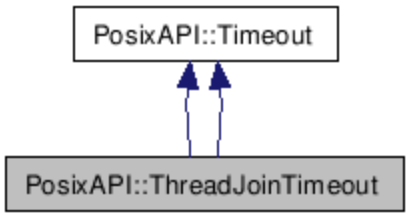
- include/qfc/.svn/text-base/thread.svn-base
- include/qfc/thread

PosixAPI::ThreadJoinTimeout Class Reference

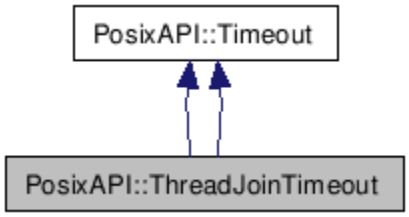
Thread join timeout class.

Inherits **PosixAPI::Timeout**, and **PosixAPI::Timeout**.

Inheritance diagram for PosixAPI::ThreadJoinTimeout:



Collaboration diagram for PosixAPI::ThreadJoinTimeout:



Detailed Description

Thread join timeout class.

Thrown when there is a thread join timeout

Author:

rennieallen@gmail.com

Definition at line 39 of file thread.svn-base.

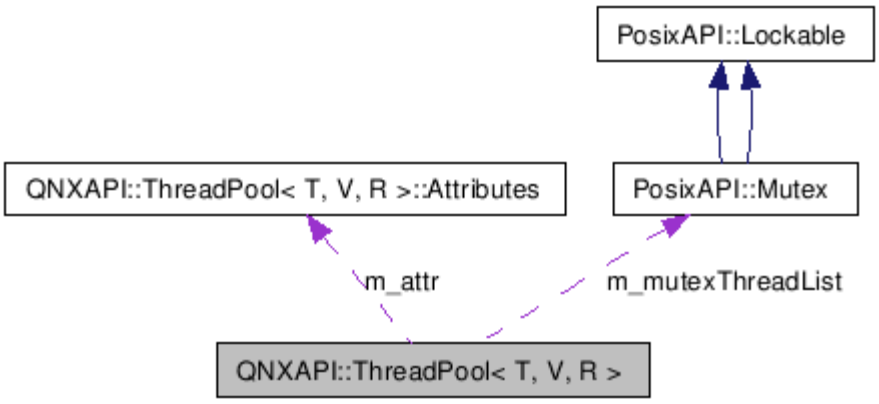
The documentation for this class was generated from the following file:

- include/qfc/.svn/text-base/thread.svn-base

QNXAPI::ThreadPool< T, V, R > Class Template Reference

Provides a thread pool manager intended to be utilized via composition.

Collaboration diagram for QNXAPI::ThreadPool< T, V, R >:



Public Member Functions

- **ThreadPool** (const **Attributes** &attr, sigc::slot< **ThreadPool**< T, V, R >::**Thread** *, **ThreadPool**< T, V, R > & > cbCreateThread, sigc::slot< int, **Context** * > cbBlock, sigc::slot< **Context** *, **ThreadPool**< T, V, R > * > cbContextAlloc, sigc::slot< void, **Context** * > cbUnblock, sigc::slot< void, **Context** * > cbHandler, sigc::slot< void, **Context** * > cbContextFree, bool exitSelf=true)
  *Construct an instance of a **ThreadPool** class (minimal re-use case).*

- **ThreadPool** (const **Attributes** &attr, sigc::slot< int, **ThreadPool**< T, V, R >::**Context** * > cbBlock, sigc::slot< void, **ThreadPool**< T, V, R >::**Context** * > cbHandler, bool exitSelf=true)
  *Construct an instance of a **ThreadPool** class (maximal re-use case).*

- **ThreadPool** (const **Attributes** &attr, sigc::slot< int, **Context** * > cbBlock, sigc::slot< void, **Context** * > cbUnblock, sigc::slot< void, **Context** * > cbHandler, bool exitSelf=true)
  *Construct an instance of a **ThreadPool** class (moderate re-use case).*

- **ThreadPool** (const **Attributes** &attr, sigc::slot< int, **Context** * > cbBlock, sigc::slot< **Context** *, **ThreadPool**< T, V, R > * > cbContextAlloc, sigc::slot< void, **Context** * > cbUnblock, sigc::slot< void, **Context** * > cbHandler, sigc::slot< void, **Context** * > cbContextFree, bool exitSelf=true)
  *Construct an instance of a **ThreadPool** class (moderate re-use case).*

- virtual **~ThreadPool** ()
  *Destroy an instance of a **ThreadPool** class.*

- void **Start** (void)
  Begin processing the pool.

- int **Block** (**Context** *context)
  Block a thread pool thread.

- **Context** & **ContextAlloc** (**ThreadPool**< T, V, R > *pool)
  Allocate a thread pool context.

- void **Unblock** (**Context** *context)
  Unblock a thread pool thread.

- void **Handler** (**Context** *context)
  Handler for a thread pool thread.

- void **ContextFree** (**Context** *context)
  Release a thread pool context.

- **ThreadPool** (const **Attributes** &attr, sigc::slot< **ThreadPool**< T, V, R >::**Thread** *, **ThreadPool**< T, V, R > & > cbCreateThread, sigc::slot< int, **Context** * > cbBlock, sigc::slot< **Context** *, **ThreadPool**< T, V, R > * > cbContextAlloc, sigc::slot< void, **Context** * > cbUnblock, sigc::slot< void, **Context** * > cbHandler, sigc::slot< void, **Context** * > cbContextFree, bool exitSelf=true)
  *Construct an instance of a **ThreadPool** class (minimal re-use case).*

- **ThreadPool** (const **Attributes** &attr, sigc::slot< int, **ThreadPool**< T, V, R >::**Context** * > cbBlock, sigc::slot< void, **ThreadPool**< T, V, R >::**Context** * > cbHandler, bool exitSelf=true)
  *Construct an instance of a **ThreadPool** class (maximal re-use case).*

- **ThreadPool** (const **Attributes** &attr, sigc::slot< int, **Context** * > cbBlock, sigc::slot< void, **Context** * > cbUnblock, sigc::slot< void, **Context** * > cbHandler, bool exitSelf=true)
  *Construct an instance of a **ThreadPool** class (moderate re-use case).*

- **ThreadPool** (const **Attributes** &attr, sigc::slot< int, **Context** * > cbBlock, sigc::slot< **Context** *, **ThreadPool**< T, V, R > * > cbContextAlloc, sigc::slot< void, **Context** * > cbUnblock, sigc::slot< void, **Context** * > cbHandler, sigc::slot< void, **Context** * > cbContextFree, bool exitSelf=true)
  *Construct an instance of a **ThreadPool** class (moderate re-use case).*

- virtual **~ThreadPool** ()

*Destroy an instance of a **ThreadPool** class.*

- void **Start** (void)
  Begin processing the pool.

- int **Block** (**Context** *context)
  Block a thread pool thread.

- **Context** & **ContextAlloc** (**ThreadPool**< T, V, R > *pool)
  Allocate a thread pool context.

- void **Unblock** (**Context** *context)
  Unblock a thread pool thread.

- void **Handler** (**Context** *context)
  Handler for a thread pool thread.

- void **ContextFree** (**Context** *context)
  Release a thread pool context.

Protected Member Functions

- virtual void **OnStart** (void)
  Entry point for the start phase of a thread.

- virtual void **OnStart** (void)
  Entry point for the start phase of a thread.

Friends

- class **ThreadPool< T, V, R >::Thread**
- class **PoolContext< T, V, R >**

Classes

- class **Attributes**
  ***Thread** pool attributes class.*

- class **Context**
  ***Thread** pool processing context class.*

- class **Thread**
  ***Thread** class.*

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::ThreadPool< T, V, R >

Provides a thread pool manager intended to be utilized via composition.

Client defines a member of this type, and supplies 2 mandatory and (up to) 4 additional callback functions:

- CreateThread: callback invoked by the pool manager when it wants to create a thread.
- ContextAlloc: callback invoked by the pool manager when it wants to allocate a new processing context.
- Unblock: callback invoked by the pool manager when it wants to unblock a thread.
- Block: callback invoked by the pool manager when it wants a thread to block.
- Handler: callback invoked by the pool manager when it wants a reception to be handled.
- ContextFree: callback invoked by the pool manager when it wants to release a processing context.

The only 2 mandatory callbacks are Block and Handler.

Block can be implemented in 2 lines:

```
   context->Receive(m_channel);
   return context->Id();
```

Handler can be implemented in 1 line:

```
   context->Dispatch(m_channel);
```

The other callbacks provide the flexibility to intercept interesting events, and the constructor is overloaded with 4 different parameter signatures.

Author:

rennieallen@gmail.com

Definition at line 62 of file threadpool.svn-base.

Constructor & Destructor Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::ThreadPool (const Attributes &* **attr**, *sigc::slot< ThreadPool< T, V, R >::Thread *, ThreadPool< T, V, R > & > * **cbCreateThread**, *sigc::slot< int, Context * > * **cbBlock**, *sigc::slot< Context *, ThreadPool< T, V, R > * > * **cbContextAlloc**, *sigc::slot< void, Context * > * **cbUnblock**, *sigc::slot< void, Context * > * **cbHandler**, *sigc::slot< void, Context * > * **cbContextFree**, *bool* **exitSelf** = *true) [inline]

Construct an instance of a **ThreadPool** class (minimal re-use case).

Parameters:

attr thread pool attributes class.
cbCreateThread callback to create a new thread.
cbBlock callback to block thread.
cbContextAlloc callback to allocate processing context.
cbUnblock callback to unblock thread.
cbHandler callback for handling a message.
cbContextFree callback to release a processing context.
exitSelf flag to determine if calling thread becomes part of pool or exits (default: true)

Definition at line 307 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::ThreadPool (const Attributes &* **attr**, *sigc::slot< int, ThreadPool< T, V, R >::Context * > * **cbBlock**, *sigc::slot< void, ThreadPool< T, V, R >::Context * > * **cbHandler**, *bool* **exitSelf** = *true) [inline]

Construct an instance of a **ThreadPool** class (maximal re-use case).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 328 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::ThreadPool (const Attributes &* **attr**, *sigc::slot< int, Context * > * **cbBlock**, *sigc::slot< void, Context * > * **cbUnblock**, *sigc::slot< void, Context * > * **cbHandler**, *bool* **exitSelf** = *true) [inline]

Construct an instance of a **ThreadPool** class (moderate re-use case).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 341 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::ThreadPool (const Attributes & attr, sigc::slot< int, Context * > cbBlock, sigc::slot< Context *, ThreadPool< T, V, R > * > cbContextAlloc, sigc::slot< void, Context * > cbUnblock, sigc::slot< void, Context * > cbHandler, sigc::slot< void, Context * > cbContextFree, bool exitSelf = true) [inline]

Construct an instance of a **ThreadPool** class (moderate re-use case).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 356 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::ThreadPool (const Attributes & attr, sigc::slot< ThreadPool< T, V, R >::Thread *, ThreadPool< T, V, R > & > cbCreateThread, sigc::slot< int, Context * > cbBlock, sigc::slot< Context *, ThreadPool< T, V, R > * > cbContextAlloc, sigc::slot< void, Context * > cbUnblock, sigc::slot< void, Context * > cbHandler, sigc::slot< void, Context * > cbContextFree, bool exitSelf = true) [inline]

Construct an instance of a **ThreadPool** class (minimal re-use case).

Parameters:

attr thread pool attributes class.
cbCreateThread callback to create a new thread.
cbBlock callback to block thread.
cbContextAlloc callback to allocate processing context.
cbUnblock callback to unblock thread.
cbHandler callback for handling a message.
cbContextFree callback to release a processing context.
exitSelf flag to determine if calling thread becomes part of pool or exits (default: true)

Definition at line 307 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::ThreadPool (const Attributes & attr, sigc::slot< int, ThreadPool< T, V, R >::Context * > cbBlock, sigc::slot< void, ThreadPool< T, V, R >::Context * > cbHandler, bool exitSelf = true) [inline]

Construct an instance of a **ThreadPool** class (maximal re-use case).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 328 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::ThreadPool (const Attributes & attr, sigc::slot< int, Context * > cbBlock, sigc::slot< void, Context * > cbUnblock, sigc::slot< void, Context * > cbHandler, bool exitSelf = true) [inline]

Construct an instance of a **ThreadPool** class (moderate re-use case).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 341 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::ThreadPool (const Attributes &* **attr**, *sigc::slot< int, Context * >* **cbBlock**, *sigc::slot< Context *, ThreadPool< T, V, R > * >* **cbContextAlloc**, *sigc::slot< void, Context * >* **cbUnblock**, *sigc::slot< void, Context * >* **cbHandler**, *sigc::slot< void, Context * >* **cbContextFree**, *bool* **exitSelf** = *true) [inline]

Construct an instance of a **ThreadPool** class (moderate re-use case).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 356 of file threadpool.

Member Function Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::ThreadPool< T, V, R >::Block (Context * **context**) [inline]

Block a thread pool thread.

Parameters:

context **Thread** context to receive into when unblocked

Definition at line 412 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> Context& QNXAPI::ThreadPool< T, V, R >::ContextAlloc (ThreadPool< T, V, R > * **pool**) [inline]

Allocate a thread pool context.

Parameters:

pool Pointer to pool to which context belongs

Definition at line 422 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Unblock (Context * **context**) [inline]

Unblock a thread pool thread.

Parameters:

context Pointer to context

Definition at line 440 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Handler (Context * **context**) [inline]

Handler for a thread pool thread.

Parameters:

context Pointer to context

Definition at line 452 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::ContextFree (Context * context) [inline]

Release a thread pool context.

Parameters:

context Pointer to context

Definition at line 462 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::ThreadPool< T, V, R >::Block (Context * context) [inline]

Block a thread pool thread.

Parameters:

context **Thread** context to receive into when unblocked

Definition at line 412 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> Context& QNXAPI::ThreadPool< T, V, R >::ContextAlloc (ThreadPool< T, V, R > * pool) [inline]

Allocate a thread pool context.

Parameters:

pool Pointer to pool to which context belongs

Definition at line 422 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Unblock (Context * context) [inline]

Unblock a thread pool thread.

Parameters:

context Pointer to context

Definition at line 440 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Handler (Context * context) [inline]

Handler for a thread pool thread.

Parameters:

context Pointer to context

Definition at line 452 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::ContextFree (Context * context) [inline]

Release a thread pool context.

Parameters:

context Pointer to context

Definition at line 462 of file threadpool.

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

- include/qfc/.svn/text-base/threadpool.svn-base
- include/qfc/threadpool

QNXAPI::ThreadPool< T, V, R >::Attributes Class Reference

Thread pool attributes class.

Public Member Functions

- **Attributes** (uint16_t loWater=1, uint16_t increment=1, uint16_t hiWater=2, uint16_t maximum=3)
  *Construct an instance of a **Attributes** class.*

- uint16_t **LowWater** (void) const
  Get the LowWater level.

- uint16_t **Increment** (void) const
  Get the Increment level.

- uint16_t **HighWater** (void) const
  Get the HighWater level.

- uint16_t **Maximum** (void) const
  Get the Maximum level.

- **Attributes** (uint16_t loWater=1, uint16_t increment=1, uint16_t hiWater=2, uint16_t maximum=3)
  *Construct an instance of a **Attributes** class.*

- uint16_t **LowWater** (void) const
  Get the LowWater level.

- uint16_t **Increment** (void) const
  Get the Increment level.

- uint16_t **HighWater** (void) const
  Get the HighWater level.

- uint16_t **Maximum** (void) const
  Get the Maximum level.

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::ThreadPool< T, V, R >::Attributes

Thread pool attributes class.

Provides a class holding thread pool attributes (supplied as the first argument to the pool constructor).

Author:
`rennieallen@gmail.com`

Definition at line 169 of file threadpool.svn-base.

Constructor & Destructor Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::Attributes::Attributes (uint16_t* loWater = *1, uint16_t* increment = *1, uint16_t* hiWater = *2, uint16_t* maximum = *3) [inline]

Construct an instance of a **Attributes** class.

Parameters:

loWater Point at which pool will create additional threads
increment Batch size for creation of new threads
hiWater Point above which pool will destroy threads that complete
maximum Maximum number of threads that the pool will allow

Definition at line 180 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::Attributes::Attributes (uint16_t* loWater = *1, uint16_t* increment = *1, uint16_t* hiWater = *2, uint16_t* maximum = *3) [inline]

Construct an instance of a **Attributes** class.

Parameters:

loWater Point at which pool will create additional threads
increment Batch size for creation of new threads
hiWater Point above which pool will destroy threads that complete
maximum Maximum number of threads that the pool will allow

Definition at line 180 of file threadpool.

Member Function Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint16_t QNXAPI::ThreadPool< T, V, R >::Attributes::LowWater (void) const [inline]

Get the LowWater level.

Returns:

a uint16_t representing the low water level.

Definition at line 189 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint16_t QNXAPI::ThreadPool< T, V, R >::Attributes::Increment (void) const [inline]

Get the Increment level.

Returns:

a uint16_t representing the increment level.

Definition at line 195 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint16_t QNXAPI::ThreadPool< T, V, R >::Attributes::HighWater (void) const [inline]

Get the HighWater level.

Returns:

a uint16_t representing the high water level.

Definition at line 201 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint16_t QNXAPI::ThreadPool< T, V, R >::Attributes::Maximum (void) const [inline]

Get the Maximum level.

Returns:

a uint16_t representing the maximum level.

Definition at line 207 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint16_t QNXAPI::ThreadPool< T, V, R >::Attributes::LowWater (void) const [inline]

Get the LowWater level.

Returns:

a uint16_t representing the low water level.

Definition at line 189 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint16_t QNXAPI::ThreadPool< T, V, R >::Attributes::Increment (void) const [inline]

Get the Increment level.

Returns:

a uint16_t representing the increment level.

Definition at line 195 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint16_t QNXAPI::ThreadPool< T, V, R >::Attributes::HighWater (void) const [inline]

Get the HighWater level.

Returns:

a uint16_t representing the high water level.

Definition at line 201 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> uint16_t QNXAPI::ThreadPool< T, V, R >::Attributes::Maximum (void) const [inline]

Get the Maximum level.

Returns:

a uint16_t representing the maximum level.

Definition at line 207 of file threadpool.

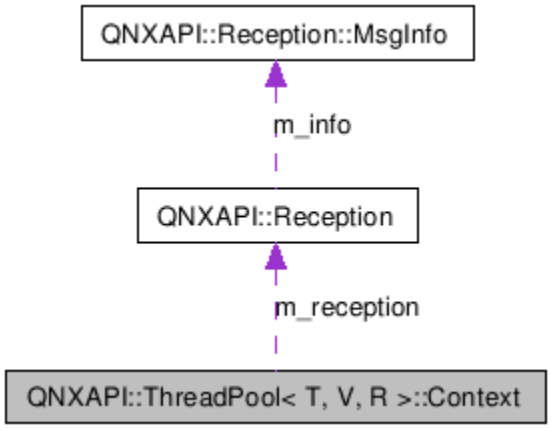
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/threadpool.svn-base
- include/qfc/threadpool

QNXAPI::ThreadPool< T, V, R >::Context Class Reference

Thread pool processing context class.

Collaboration diagram for QNXAPI::ThreadPool< T, V, R >::Context:



Public Member Functions

- **Context** ()
  *Construct an instance of a thread pool **Context** class.*

- virtual **~Context** ()
  *Destroy an instance of a thread pool **Context** class.*

- void **Id** (int id)
  Set the context id.

- int **Id** (void)
  Get the context id.

- **QNXAPI::Reception** & **Reception** (void)
  Obtain the reception component of the context.

- void **Receive** (**QNXAPI::Channel**< T, V, R > &channel)
  Receive a message into the context.

- void **Dispatch** (**QNXAPI::Channel**< T, V, R > &channel)
  Dispatch a received message via a channel.

- **Context** ()
  *Construct an instance of a thread pool **Context** class.*

- virtual **~Context** ()
  *Destroy an instance of a thread pool **Context** class.*

- void **Id** (int id)
  Set the context id.

- int **Id** (void)

Get the context id.

- **QNXAPI::Reception** & **Reception** (void)
  Obtain the reception component of the context.

- void **Receive** (**QNXAPI::Channel**< T, V, R > &channel)
  Receive a message into the context.

- void **Dispatch** (**QNXAPI::Channel**< T, V, R > &channel)
  Dispatch a received message via a channel.

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::ThreadPool< T, V, R >::Context

Thread pool processing context class.

Holds the context for an instance of a thread pool thread.

A context is currently a reception.

Author:
`renniеallen@gmail.com`

Definition at line 225 of file threadpool.svn-base.

Member Function Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Context::Id (int* id*) [inline]

Set the context id.

Parameters:
id tag for the context.

Definition at line 246 of file threadpool.svn-base.

References QNXAPI::Reception::Id().

Here is the call graph for this function:

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::ThreadPool< T, V, R >::Context::Id (void) [inline]

Get the context id.

Returns:

integer id tag for the context.

Definition at line 256 of file threadpool.svn-base.

References QNXAPI::Reception::Id().

Here is the call graph for this function:

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::Reception& QNXAPI::ThreadPool< T, V, R >::Context::Reception (void) [inline]

Obtain the reception component of the context.

Returns:

QNXAPI::Reception Reference to reception context.

Definition at line 266 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Context::Receive (QNXAPI::Channel< T, V, R > & channel) [inline]

Receive a message into the context.

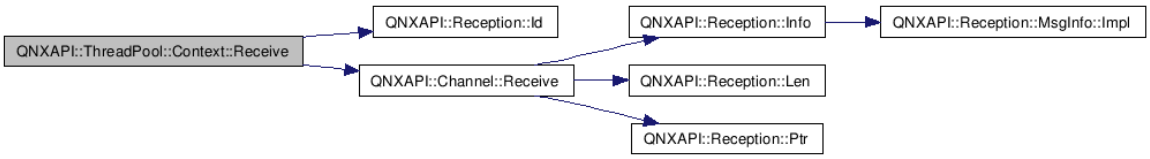
Parameters:

channel **Channel** to receive from.

Definition at line 276 of file threadpool.svn-base.

References QNXAPI::Reception::Id(), and QNXAPI::Channel< T, V, R >::Receive().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Context::Dispatch (QNXAPI::Channel< T, V, R > & channel) [inline]

Dispatch a received message via a channel.

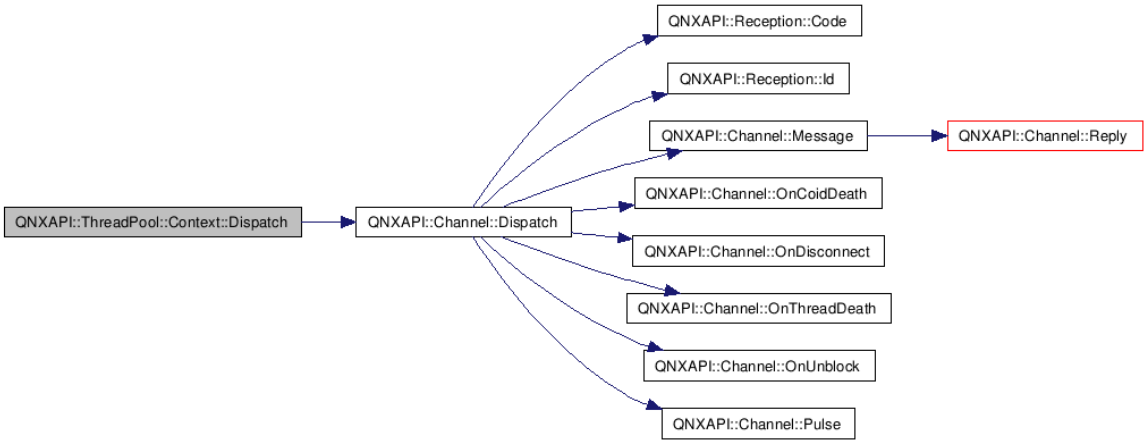
Parameters:

channel **Channel** to dispatch from.

Definition at line 286 of file threadpool.svn-base.

References QNXAPI::Channel< T, V, R >::Dispatch().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Context::Id (int id) [inline]

Set the context id.

Parameters:

id tag for the context.

Definition at line 246 of file threadpool.

References QNXAPI::Reception::Id().

Here is the call graph for this function:

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> int QNXAPI::ThreadPool< T, V, R >::Context::Id (void) [inline]

Get the context id.

Returns:

integer id tag for the context.

Definition at line 256 of file threadpool.

References QNXAPI::Reception::Id().

Here is the call graph for this function:

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::Reception& QNXAPI::ThreadPool< T, V, R >::Context::Reception (void) [inline]

Obtain the reception component of the context.

Returns:

QNXAPI::Reception Reference to reception context.

Definition at line 266 of file threadpool.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Context::Receive (QNXAPI::Channel< T, V, R > & channel) [inline]

Receive a message into the context.

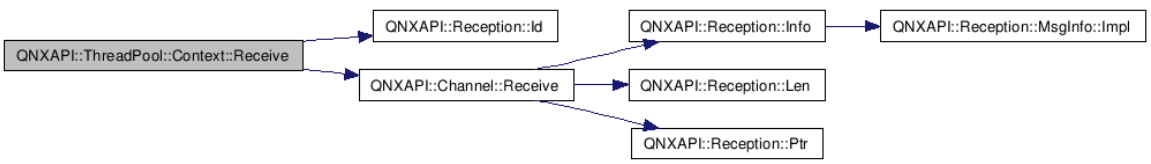
Parameters:

channel **Channel** to receive from.

Definition at line 276 of file threadpool.

References QNXAPI::Reception::Id(), and QNXAPI::Channel< T, V, R >::Receive().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::ThreadPool< T, V, R >::Context::Dispatch (QNXAPI::Channel< T, V, R > & channel) [inline]

Dispatch a received message via a channel.

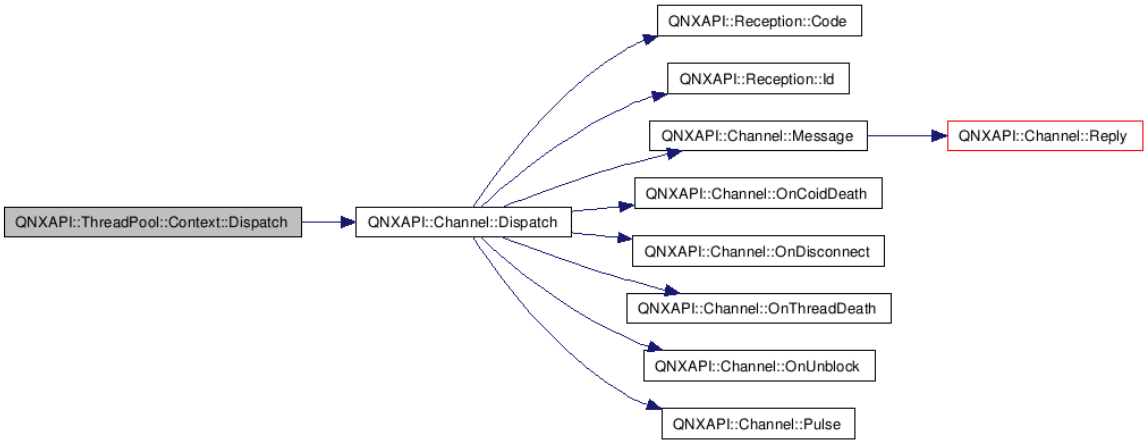
Parameters:

channel **Channel** to dispatch from.

Definition at line 286 of file threadpool.

References QNXAPI::Channel< T, V, R >::Dispatch().

Here is the call graph for this function:



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

- include/qfc/.svn/text-base/threadpool.svn-base
- include/qfc/threadpool

QNXAPI::ThreadPool< T, V, R >::Thread Class Reference

Thread class.

Public Member Functions

- **Thread** (**ThreadPool**< T, V, R > &pool)
  *Construct an instance of a thread pool **Thread** class.*

- **Thread** (**ThreadPool**< T, V, R > &pool)
  *Construct an instance of a thread pool **Thread** class.*

Protected Member Functions

- void * **OnRun** (void)
  Entry point for begining of threads run phase.

- void * **OnRun** (void)
  Entry point for begining of threads run phase.

Friends

- class **ThreadPool**< **T, V, R** >

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::ThreadPool< T, V, R >::Thread

Thread class.

Provides a thread pool extended thread class.

A thread pool thread has a pool member, to hold a reference to the pool to which it belongs, as well as an Update method used to update the pool counters.

Author:
rennieallen@gmail.com

Definition at line 76 of file threadpool.svn-base.

Constructor & Destructor Documentation

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::Thread::Thread (ThreadPool< T, V, R > & pool) [inline]

Construct an instance of a thread pool **Thread** class.

Parameters:
pool pool to which thread belongs.

Definition at line 86 of file threadpool.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::ThreadPool< T, V, R >::Thread::Thread (ThreadPool< T, V, R > & pool) [inline]

Construct an instance of a thread pool **Thread** class.

Parameters:

pool pool to which thread belongs.

Definition at line 86 of file threadpool.

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

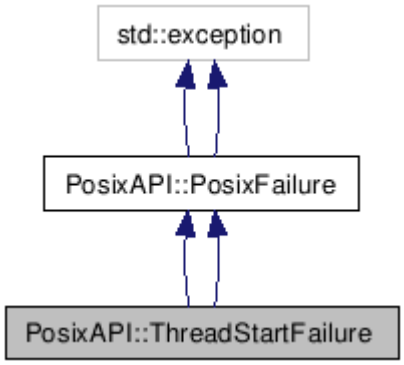
- include/qfc/.svn/text-base/threadpool.svn-base
- include/qfc/threadpool

PosixAPI::ThreadStartFailure Class Reference

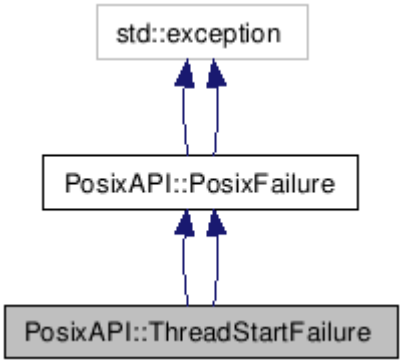
Thread class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for PosixAPI::ThreadStartFailure:



Collaboration diagram for PosixAPI::ThreadStartFailure:



Public Member Functions

- **ThreadStartFailure** (int err)
  *Construct an instance of a **ThreadStartFailure** class.*

- **ThreadStartFailure** (int err)
  *Construct an instance of a **ThreadStartFailure** class.*

Detailed Description

Thread class.

Encapsulates basic thread functionality.

Author:
`renniеallen@gmail.com`

Definition at line 50 of file thread.svn-base.

Constructor & Destructor Documentation

PosixAPI::ThreadStartFailure::ThreadStartFailure (int* err*) [inline]

Construct an instance of a **ThreadStartFailure** class.

Parameters:
err Posix error code

Definition at line 58 of file thread.svn-base.PosixAPI::ThreadStartFailure::ThreadStartFailure (int err) [inline]

Construct an instance of a **ThreadStartFailure** class.

Parameters:

err Posix error code

Definition at line 58 of file thread.

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

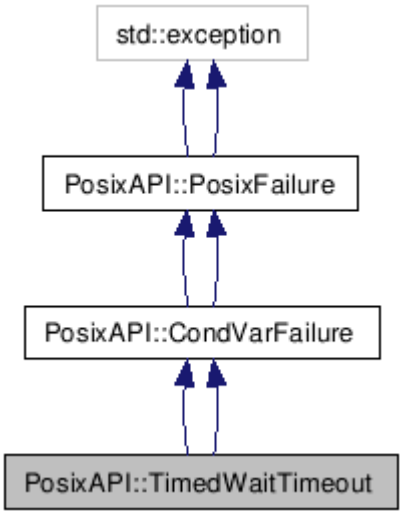
- include/qfc/.svn/text-base/thread.svn-base
- include/qfc/thread

PosixAPI::TimedWaitTimeout Class Reference

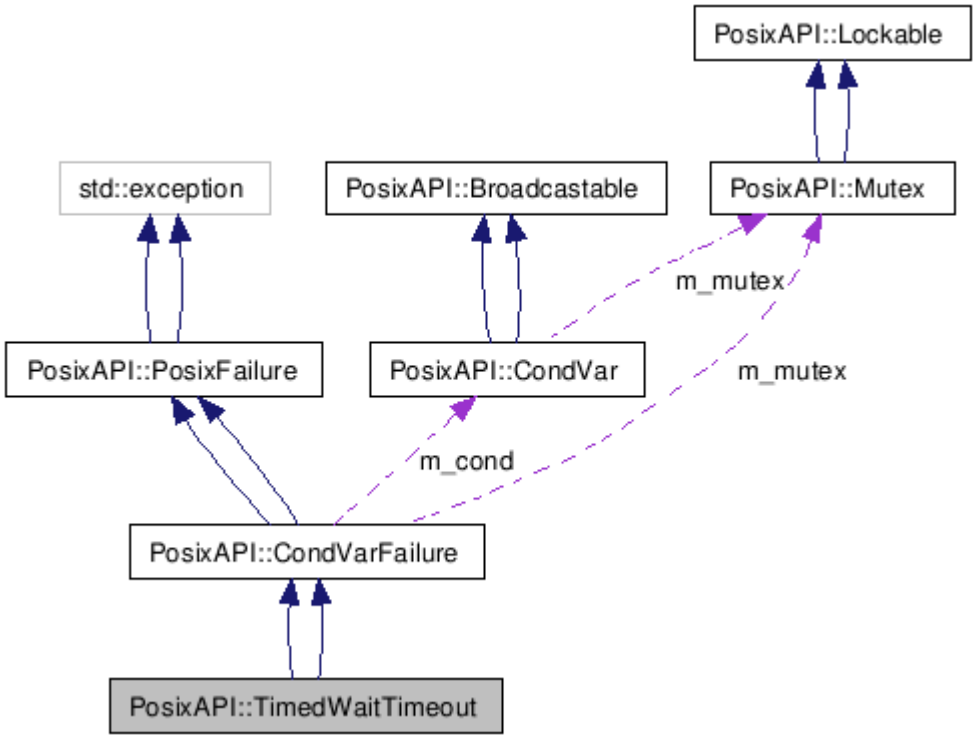
TimedWaitTimeout class.

Inherits **PosixAPI::CondVarFailure**, and **PosixAPI::CondVarFailure**.

Inheritance diagram for PosixAPI::TimedWaitTimeout:



Collaboration diagram for PosixAPI::TimedWaitTimeout:



Public Member Functions

- **TimedWaitTimeout** (const **CondVar** &cond, const **Mutex** &mutex, int err)
  *Construct an instance of a **TimedWaitTimeout** class.*

- **TimedWaitTimeout** (const **CondVar** &cond, const **Mutex** &mutex, int err)
  *Construct an instance of a **TimedWaitTimeout** class.*

Detailed Description

TimedWaitTimeout class.

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 198 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::TimedWaitTimeout::TimedWaitTimeout (const CondVar &* cond*, const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **TimedWaitTimeout** class.

Parameters:

cond **CondVar** that experienced exception
mutex **Mutex** associated with **CondVar**
err Posix error code of failure

Definition at line 208 of file synchron.svn-base.PosixAPI::TimedWaitTimeout::TimedWaitTimeout (const CondVar &* cond*, const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **TimedWaitTimeout** class.

Parameters:

cond **CondVar** that experienced exception
mutex **Mutex** associated with **CondVar**
err Posix error code of failure

Definition at line 208 of file synchron.

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

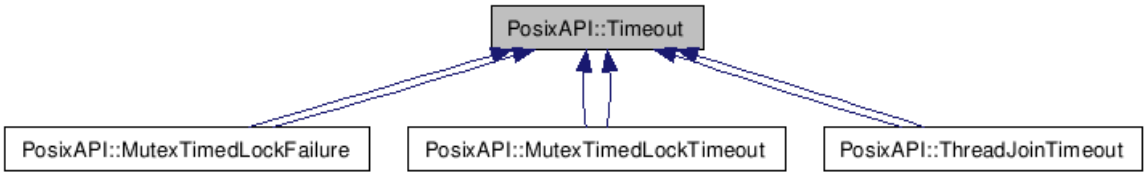
- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

PosixAPI::Timeout Class Reference

Timeout base class.

Inherited by **PosixAPI::MutexTimedLockFailure**, **PosixAPI::MutexTimedLockFailure**, **PosixAPI::MutexTimedLockTimeout**, **PosixAPI::MutexTimedLockTimeout**, **PosixAPI::ThreadJoinTimeout**, and **PosixAPI::ThreadJoinTimeout**.

Inheritance diagram for PosixAPI::Timeout:



Detailed Description

Timeout base class.

Base for timeout exceptions.

Definition at line 32 of file except.svn-base.

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

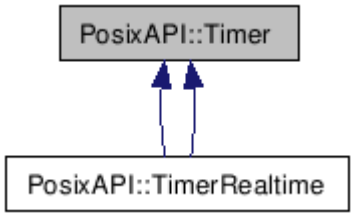
- include/qfc/.svn/text-base/except.svn-base

PosixAPI::Timer Class Reference

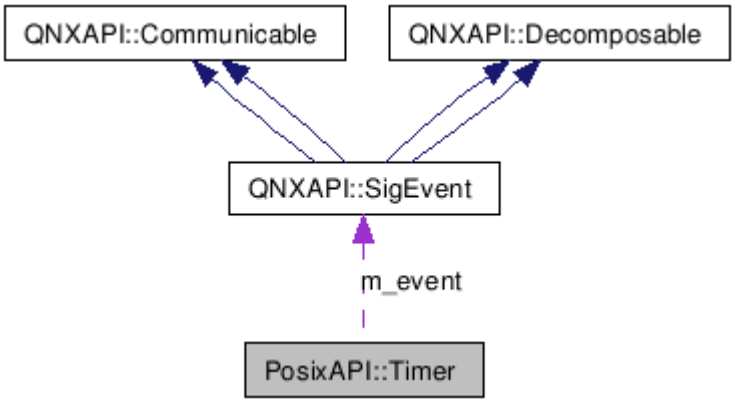
Posix **Timer** class.

Inherited by **PosixAPI::TimerRealtime**, and **PosixAPI::TimerRealtime**.

Inheritance diagram for PosixAPI::Timer:



Collaboration diagram for PosixAPI::Timer:



Public Types

- enum **Type**
- enum **Type**

Public Member Functions

- **Timer** (**Type** type, **QNXAPI::SigEvent** &event)
- virtual **~Timer** (void)
- void **Set** (**Reltime** &fire)
- void **Set** (**Abstime** &fire)
- uint64_t **Get** (void) throw ()
- uint64_t **Period** (void) throw ()
- uint64_t **Cancel** (void)
- **Timer** (**Type** type, **QNXAPI::SigEvent** &event)
- virtual **~Timer** (void)
- void **Set** (**Reltime** &fire)
- void **Set** (**Abstime** &fire)
- uint64_t **Get** (void) throw ()
- uint64_t **Period** (void) throw ()
- uint64_t **Cancel** (void)

Classes

- class **Abstime**
  Absolute time class.

- class **Reltime**
  Relative time class.

Detailed Description

Posix **Timer** class.

Encapsulates functionality for manipulating posix timers.

Author:

Rennie Allen `rennieallen@gmail.com`

Definition at line 104 of file posixtimer.svn-base.

Member Enumeration Documentation

enum PosixAPI::Timer::Type

Timer types

Definition at line 110 of file posixtimer.svn-base.enum PosixAPI::Timer::Type

Timer types

Definition at line 110 of file posixtimer.

Constructor & Destructor Documentation

PosixAPI::Timer::Timer (Type* type*, QNXAPI::SigEvent &* event*)

Construct an instance of **Timer** class.

Parameters:

type a QNXAPI::Timer::Type.
event **QNXAPI::SigEvent** to be delivered when timer fires.

virtual PosixAPI::Timer::~Timer (void) [virtual]

Destroy an instance of **Timer** class.

PosixAPI::Timer::Timer (Type* type*, QNXAPI::SigEvent &* event*)

Construct an instance of **Timer** class.

Parameters:

type a QNXAPI::Timer::Type.
event **QNXAPI::SigEvent** to be delivered when timer fires.

virtual PosixAPI::Timer::~Timer (void) [virtual]

Destroy an instance of **Timer** class.

Member Function Documentation

void PosixAPI::Timer::Set (Reltime &* fire*)

Set the timer using a relative time.

Parameters:

fire a QNXAPI::Timer::Reltime time relative to "now" when the timer is to fire.

void PosixAPI::Timer::Set (Abstime &* fire*)

Set the timer using an absolute time.

Parameters:

fire a QNXAPI::Timer::Abstime time when the timer is to fire.

uint64_t PosixAPI::Timer::Get (void) throw ()

Get the remaining time before the timer fires.

Returns:

time remaining expressed in nanoseconds.

uint64_t PosixAPI::Timer::Period (void) throw ()

Get the period at which a repetitive timer fires.

Returns:
period of timer expressed in nanoseconds.

uint64_t PosixAPI::Timer::Cancel (void)

Cancel a timer.

Returns:
time remaining before timer would have fired, expressed in nanoseconds.

void PosixAPI::Timer::Set (Reltime & fire)

Set the timer using a relative time.

Parameters:
fire a QNXAPI::Timer::Reltime time relative to "now" when the timer is to fire.

void PosixAPI::Timer::Set (Abstime & fire)

Set the timer using an absolute time.

Parameters:
fire a QNXAPI::Timer::Abstime time when the timer is to fire.

uint64_t PosixAPI::Timer::Get (void) throw ()

Get the remaining time before the timer fires.

Returns:
time remaining expressed in nanoseconds.

uint64_t PosixAPI::Timer::Period (void) throw ()

Get the period at which a repetitive timer fires.

Returns:
period of timer expressed in nanoseconds.

uint64_t PosixAPI::Timer::Cancel (void)

Cancel a timer.

Returns:
time remaining before timer would have fired, expressed in nanoseconds.

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

- include/qfc/.svn/text-base/posixtimer.svn-base
- include/qfc/posixtimer

PosixAPI::Timer::Abstime Class Reference

Absolute time class.

Public Member Functions

- **Abstime** (uint64_t expiry)
  *Construct an instance of a **Abstime** class.*

- void **Expiry** (uint64_t time) throw ()
  Set the expiration time of an absolute time.

- uint64_t **Expiry** (void) const throw ()
  Obtain the expiration time of an absolute time.

- **Abstime** (uint64_t expiry)
  *Construct an instance of a **Abstime** class.*

- void **Expiry** (uint64_t time) throw ()
  Set the expiration time of an absolute time.

- uint64_t **Expiry** (void) const throw ()
  Obtain the expiration time of an absolute time.

Detailed Description

Absolute time class.

Holds an absolute time value

Definition at line 122 of file posixtimer.svn-base.

Constructor & Destructor Documentation

PosixAPI::Timer::Abstime::Abstime (uint64_t* expiry*) [inline]

Construct an instance of a **Abstime** class.

Parameters:

expiry Expiration time (expressed as an absolute value)

Definition at line 130 of file posixtimer.svn-base.PosixAPI::Timer::Abstime::Abstime (uint64_t expiry) [inline]

Construct an instance of a **Abstime** class.

Parameters:

expiry Expiration time (expressed as an absolute value)

Definition at line 130 of file posixtimer.

Member Function Documentation

void PosixAPI::Timer::Abstime::Expiry (uint64_t time) throw () [inline]

Set the expiration time of an absolute time.

Parameters:

time Expiration time (expressed as an absolute value)

Definition at line 139 of file posixtimer.svn-base.uint64_t PosixAPI::Timer::Abstime::Expiry (void) const throw () [inline]

Obtain the expiration time of an absolute time.

Returns:

Expiration time (expressed as an absolute value)

Definition at line 149 of file posixtimer.svn-base.void PosixAPI::Timer::Abstime::Expiry (uint64_t time) throw () [inline]

Set the expiration time of an absolute time.

Parameters:

time Expiration time (expressed as an absolute value)

Definition at line 139 of file posixtimer.uint64_t PosixAPI::Timer::Abstime::Expiry (void) const throw () [inline]

Obtain the expiration time of an absolute time.

Returns:

Expiration time (expressed as an absolute value)

Definition at line 149 of file posixtimer.

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

- include/qfc/.svn/text-base/posixtimer.svn-base
- include/qfc/posixtimer

PosixAPI::Timer::Reltime Class Reference

Relative time class.

Public Member Functions

- **Reltime** (uint64_t initial, uint64_t interval=0)
  *Construct an instance of a **Reltime** class.*

- uint64_t **Initial** (void)
  Obtain the Initial expiration time (relative).

- uint64_t **Interval** (void)
  Obtain the Interval (period).

- **Reltime** (uint64_t initial, uint64_t interval=0)
  *Construct an instance of a **Reltime** class.*

- uint64_t **Initial** (void)
  Obtain the Initial expiration time (relative).

- uint64_t **Interval** (void)
  Obtain the Interval (period).

Detailed Description

Relative time class.

Holds a relative time value, and a relative repeat interval (period)

Definition at line 167 of file posixtimer.svn-base.

Constructor & Destructor Documentation

PosixAPI::Timer::Reltime::Reltime (uint64_t* initial*, uint64_t* interval = *0) [inline]

Construct an instance of a **Reltime** class.

Parameters:

initial Initial expiration time (expressed as a relative value)
interval Automatically armed repeat interval (period)

Definition at line 176 of file posixtimer.svn-base.PosixAPI::Timer::Reltime::Reltime (uint64_t* initial*, uint64_t* interval = *0) [inline]

Construct an instance of a **Reltime** class.

Parameters:

initial Initial expiration time (expressed as a relative value)
interval Automatically armed repeat interval (period)

Definition at line 176 of file posixtimer.

Member Function Documentation

uint64_t PosixAPI::Timer::Reltime::Initial (void) [inline]

Obtain the Initial expiration time (relative).

Returns:

Initial expiration time (relative)

Definition at line 185 of file posixtimer.svn-base.

Referenced by QNXAPI::TimerBank< T, V, R >::Set().uint64_t PosixAPI::Timer::Reltime::Interval (void) [inline]

Obtain the Interval (period).

Returns:

Interval (period)

Definition at line 195 of file posixtimer.svn-base.

Referenced by QNXAPI::TimerBank< T, V, R >::Set().uint64_t PosixAPI::Timer::Reltime::Initial (void) [inline]

Obtain the Initial expiration time (relative).

Returns:

Initial expiration time (relative)

Definition at line 185 of file posixtimer.uint64_t PosixAPI::Timer::Reltime::Interval (void) [inline]

Obtain the Interval (period).

Returns:

Interval (period)

Definition at line 195 of file posixtimer.

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

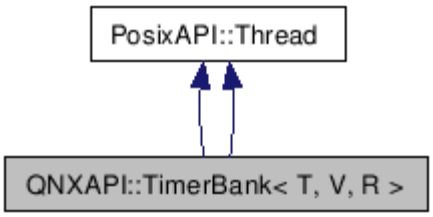
- include/qfc/.svn/text-base/posixtimer.svn-base
- include/qfc/posixtimer

QNXAPI::TimerBank< T, V, R > Class Template Reference

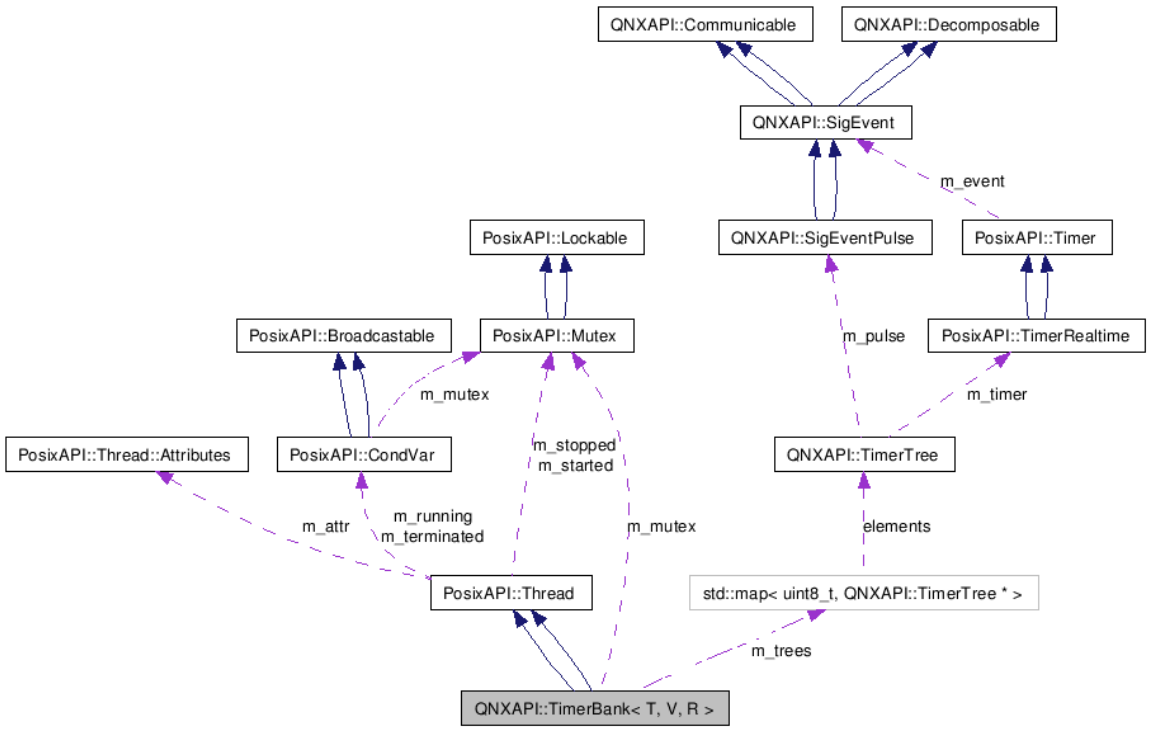
TimerBank class.

Inherits **PosixAPI::Thread**, and **PosixAPI::Thread**.

Inheritance diagram for QNXAPI::TimerBank< T, V, R >:



Collaboration diagram for QNXAPI::TimerBank< T, V, R >:



Public Member Functions

- **TimerBank** (void)
  *Construct an instance of a **TimerBank** class.*

- QNXAPI::TimerTree::TimerNode * **Set** (**PosixAPI::Timer::Abstime** time, sigc::slot< bool > cb, uint8_t priority, uint64_t reload=0)
- QNXAPI::TimerTree::TimerNode * **Set** (**PosixAPI::Timer::Reltime** time, sigc::slot< bool > cb, uint8_t priority)
- void **Cancel** (QNXAPI::TimerTree::TimerNode *tmr, sigc::slot< bool > cb)
- **~TimerBank** (void)
  *Destroy an instance of a **TimerBank** class.*

- **TimerBank** (void)

*Construct an instance of a **TimerBank** class.*

- QNXAPI::TimerTree::TimerNode * **Set** (**PosixAPI::Timer::Abstime** time, sigc::slot< bool > cb, uint8_t priority, uint64_t reload=0)
- QNXAPI::TimerTree::TimerNode * **Set** (**PosixAPI::Timer::Reltime** time, sigc::slot< bool > cb, uint8_t priority)
- void **Cancel** (QNXAPI::TimerTree::TimerNode *tmr, sigc::slot< bool > cb)
- **~TimerBank** (void)
  *Destroy an instance of a **TimerBank** class.*

Protected Member Functions

- virtual void * **OnRun** (void)
  Entry point for the run phase of a thread.

- virtual void **OnStart** (void)
  Entry point for the start phase of a thread.

- virtual void * **OnRun** (void)
  Entry point for the run phase of a thread.

- virtual void **OnStart** (void)
  Entry point for the start phase of a thread.

Detailed Description

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> class QNXAPI::TimerBank< T, V, R >

TimerBank class.

Manages up to 255 TimerTrees organized by priority.

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 59 of file timerbank.svn-base.

Member Function Documentation

```
template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem>
QNXAPI::TimerTree::TimerNode* QNXAPI::TimerBank< T, V, R >::Set (PosixAPI::Timer::Abstime time,
sigc::slot< bool > cb, uint8_t priority, uint64_t reload = 0) [inline]
```

Set a **TimerBank** timer using an absolute time.

Parameters:

time Absolute time at which timer expires (fires)
cb Callback to be invoked when timer expires
priority Priority of the timer
reload Reload value if repetitive

Returns:

Pointer to a TimerNode (needed to cancel timer)

Definition at line 82 of file timerbank.svn-base.

```
Referenced by QNXAPI::TimerBank< T, V, R >::Set().template<typename T = iovItem, typename V =
SimpleVectLoader, typename R = iovItem> QNXAPI::TimerTree::TimerNode* QNXAPI::TimerBank< T, V, R
>::Set (PosixAPI::Timer::Reltime time, sigc::slot< bool > cb, uint8_t priority) [inline]
```

Set the time of a timer node in the timer bank, using an relative time. A Reltime has two components (an initial value, and an interval value); if the interval value is non zero the timer can be repeated (by return true from the callback). If the interval is zero, this arms a one-shot relatively specified timer.

Parameters:

time Relative time at which timer expires
cb Callback to be invoked when timer expires
priority Priority of the timer

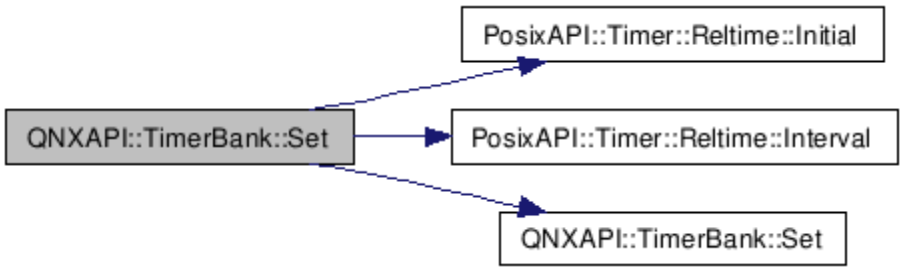
Returns:

Pointer to a TimerNode (needed to cancel timer)

Definition at line 110 of file timerbank.svn-base.

References PosixAPI::Timer::Reltime::Initial(), PosixAPI::Timer::Reltime::Interval(), and QNXAPI::TimerBank< T, V, R >::Set().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::TimerBank< T, V, R >::Cancel (QNXAPI::TimerTree::TimerNode * tmr, sigc::slot< bool > cb) [inline]

Cancel a **TimerBank** timer.

Parameters:

tmr Timer node returned from **Set()**
cb Callback specified when the timer was set

Definition at line 121 of file timerbank.svn-base.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void* QNXAPI::TimerBank< T, V, R >::OnRun (void) [inline, protected, virtual]

Entry point for the run phase of a thread.

Returns:

Void pointer

Implements **PosixAPI::Thread** (*p.338*).

Definition at line 152 of file timerbank.svn-base.

References PosixAPI::Thread::ShouldStop().

Here is the call graph for this function:

template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::TimerTree::TimerNode* QNXAPI::TimerBank< T, V, R >::Set (PosixAPI::Timer::Abstime time, sigc::slot< bool > cb, uint8_t priority, uint64_t reload = 0) [inline]

Set a **TimerBank** timer using an absolute time.

Parameters:

time Absolute time at which timer expires (fires)
cb Callback to be invoked when timer expires
priority Priority of the timer
reload Reload value if repetitive

Returns:

Pointer to a TimerNode (needed to cancel timer)

Definition at line 82 of file timerbank.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> QNXAPI::TimerTree::TimerNode* QNXAPI::TimerBank< T, V, R >::Set (PosixAPI::Timer::Reltime time, sigc::slot< bool > cb, uint8_t priority) [inline]

Set the time of a timer node in the timer bank, using an relative time. A Reltime has two components (an initial value, and an interval value); if the interval value is non zero the timer can be repeated (by return true from the callback). If the interval is zero, this arms a one-shot relatively specified timer.

Parameters:

time Relative time at which timer expires
cb Callback to be invoked when timer expires
priority Priority of the timer

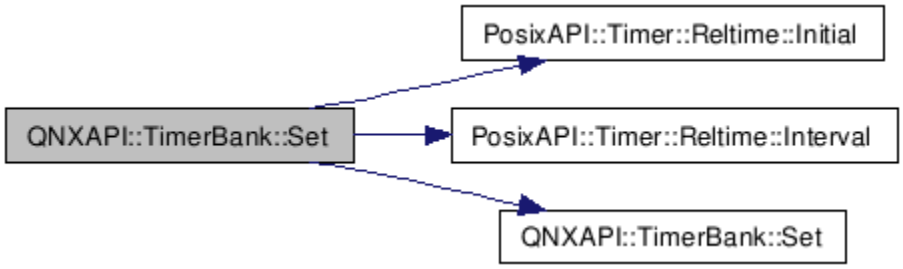
Returns:

Pointer to a TimerNode (needed to cancel timer)

Definition at line 110 of file timerbank.

References PosixAPI::Timer::Reltime::Initial(), PosixAPI::Timer::Reltime::Interval(), and QNXAPI::TimerBank< T, V, R >::Set().

Here is the call graph for this function:



template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> void QNXAPI::TimerBank< T, V, R >::Cancel (QNXAPI::TimerTree::TimerNode * tmr, sigc::slot< bool > cb) [inline]

Cancel a **TimerBank** timer.

Parameters:

tmr Timer node returned from **Set()**
cb Callback specified when the timer was set

Definition at line 121 of file timerbank.template<typename T = iovItem, typename V = SimpleVectLoader, typename R = iovItem> virtual void* QNXAPI::TimerBank< T, V, R >::OnRun (void) [inline, protected, virtual]

Entry point for the run phase of a thread.

Returns:

Void pointer

Implements **PosixAPI::Thread** (*p.338*).

Definition at line 152 of file timerbank.

References PosixAPI::Thread::ShouldStop().

Here is the call graph for this function:

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

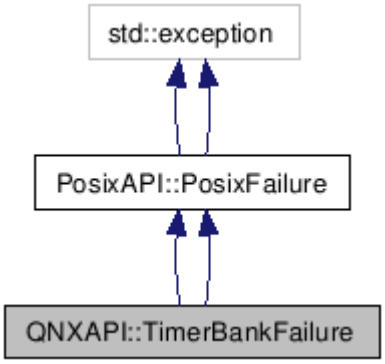
- include/qfc/.svn/text-base/timerbank.svn-base
- include/qfc/timerbank

QNXAPI::TimerBankFailure Class Reference

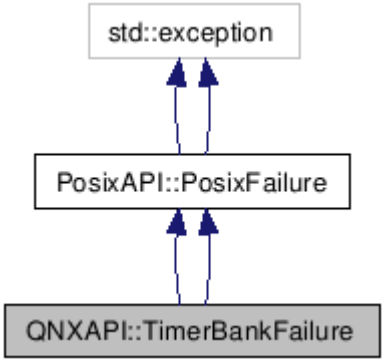
TimerBankFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::TimerBankFailure:



Collaboration diagram for QNXAPI::TimerBankFailure:



Public Member Functions

- **TimerBankFailure** (int err)
- **TimerBankFailure** (int err)

Detailed Description

TimerBankFailure class.

Exception encountered during setting of a timer.

Definition at line 39 of file timerbank.svn-base.

Constructor & Destructor Documentation

QNXAPI::TimerBankFailure::TimerBankFailure (int* err*) [inline]

Construct a **TimerBankFailure** class.

Parameters:

err standard error code.

Definition at line 47 of file timerbank.svn-base.QNXAPI::TimerBankFailure::TimerBankFailure (int* err*) [inline]

Construct a **TimerBankFailure** class.

Parameters:

err standard error code.

Definition at line 47 of file timerbank.

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

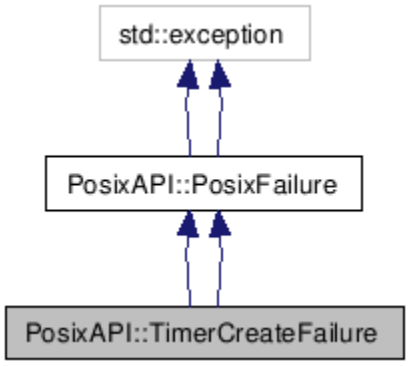
- include/qfc/.svn/text-base/timerbank.svn-base
- include/qfc/timerbank

PosixAPI::TimerCreateFailure Class Reference

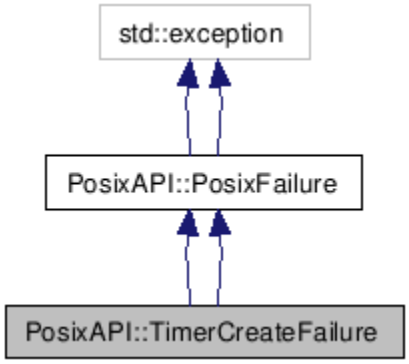
TimerCreateFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for PosixAPI::TimerCreateFailure:



Collaboration diagram for PosixAPI::TimerCreateFailure:



Public Member Functions

- **TimerCreateFailure** (int err)
- **TimerCreateFailure** (int err)

Detailed Description

TimerCreateFailure class.

Exception encountered during creation of a posix timer.

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 44 of file posixtimer.svn-base.

Constructor & Destructor Documentation

PosixAPI::TimerCreateFailure::TimerCreateFailure (int* err*) [inline]

Construct a **TimerCreateFailure** class.

Parameters:

err standard error code.

Definition at line 52 of file posixtimer.svn-base.PosixAPI::TimerCreateFailure::TimerCreateFailure (int* err*) [inline]

Construct a **TimerCreateFailure** class.

Parameters:

err standard error code.

Definition at line 52 of file posixtimer.

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

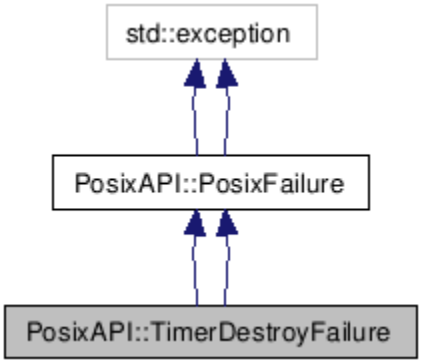
- include/qfc/.svn/text-base/posixtimer.svn-base
- include/qfc/posixtimer

PosixAPI::TimerDestroyFailure Class Reference

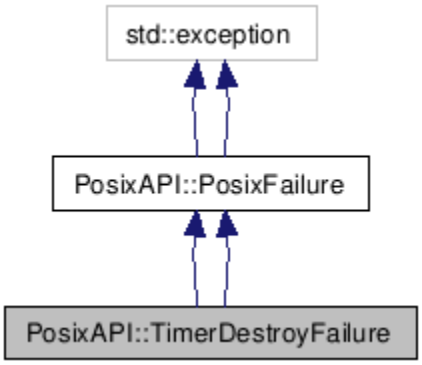
TimerDestroyFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for PosixAPI::TimerDestroyFailure:



Collaboration diagram for PosixAPI::TimerDestroyFailure:



Public Member Functions

- **TimerDestroyFailure** (int err)
- **TimerDestroyFailure** (int err)

Detailed Description

TimerDestroyFailure class.

Exception encountered during destruction of a posix timer.

Author:

Rennie Allen `rennieallen@gmail.com`

Definition at line 64 of file posixtimer.svn-base.

Constructor & Destructor Documentation

PosixAPI::TimerDestroyFailure::TimerDestroyFailure (int err) [inline]

Construct a **TimerDestroyFailure** class.

Parameters:

err standard error code.

Definition at line 72 of file posixtimer.svn-base.PosixAPI::TimerDestroyFailure::TimerDestroyFailure (int err) [inline]

Construct a **TimerDestroyFailure** class.

Parameters:

err standard error code.

Definition at line 72 of file posixtimer.

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

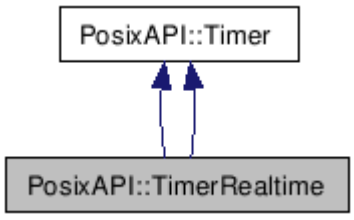
- include/qfc/.svn/text-base/posixtimer.svn-base
- include/qfc/posixtimer

PosixAPI::TimerRealtime Class Reference

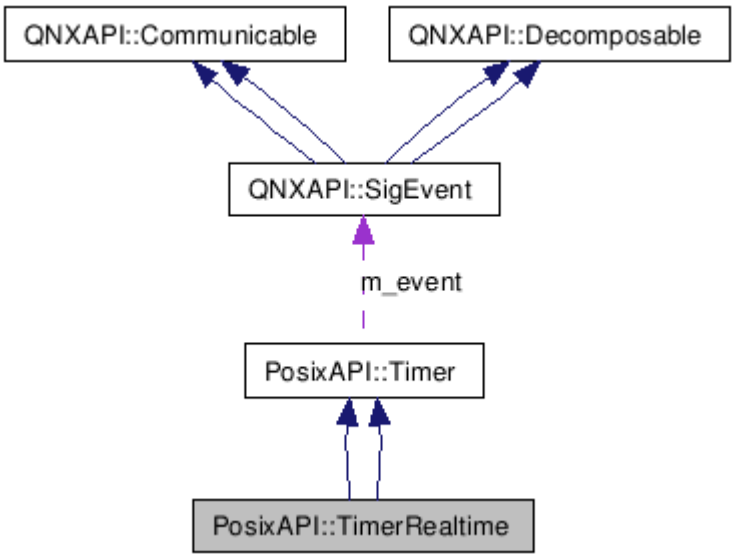
Posix Realtime **Timer** class.

Inherits **PosixAPI::Timer**, and **PosixAPI::Timer**.

Inheritance diagram for PosixAPI::TimerRealtime:



Collaboration diagram for PosixAPI::TimerRealtime:



Public Member Functions

- **TimerRealtime** (**QNXAPI::SigEvent** &event)
  *Construct an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.*

- **TimerRealtime** (const **TimerRealtime** &other)
  *Copy construct an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.*

- virtual **~TimerRealtime** (void)
  *Destroy an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.*

- **TimerRealtime** (**QNXAPI::SigEvent** &event)
  *Construct an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.*

- **TimerRealtime** (const **TimerRealtime** &other)
  *Copy construct an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.*

- virtual **~TimerRealtime** (void)
  *Destroy an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.*

Detailed Description

Posix Realtime **Timer** class.

Convenience class for QNXAPI::Timer classes of QNXAPI::Timer::Type::Realtime

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 284 of file posixtimer.svn-base.

Constructor & Destructor Documentation

PosixAPI::TimerRealtime::TimerRealtime (QNXAPI::SigEvent &* `event`*) [inline]

Construct an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.

Parameters:

event **QNXAPI::SigEvent** to be delivered when timer fires.

Definition at line 292 of file posixtimer.svn-base.PosixAPI::TimerRealtime::TimerRealtime (const TimerRealtime &* `other`*) [inline]

Copy construct an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.

Parameters:

other Other **TimerRealtime** to copy from

Definition at line 301 of file posixtimer.svn-base.PosixAPI::TimerRealtime::TimerRealtime (QNXAPI::SigEvent &* `event`*) [inline]

Construct an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.

Parameters:

event **QNXAPI::SigEvent** to be delivered when timer fires.

Definition at line 292 of file posixtimer.PosixAPI::TimerRealtime::TimerRealtime (const TimerRealtime & other) [inline]

Copy construct an instance of **Timer** class of type QNXAPI::Timer::Type::Realtime.

Parameters:

other Other **TimerRealtime** to copy from

Definition at line 301 of file posixtimer.

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

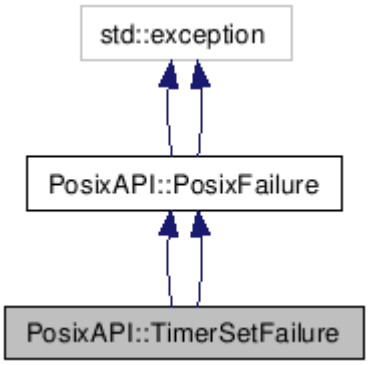
- include/qfc/.svn/text-base/posixtimer.svn-base
- include/qfc/posixtimer

PosixAPI::TimerSetFailure Class Reference

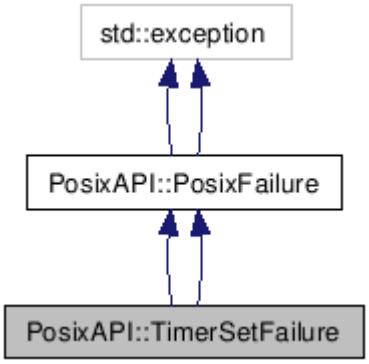
TimerSetFailure class.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for PosixAPI::TimerSetFailure:



Collaboration diagram for PosixAPI::TimerSetFailure:



Public Member Functions

- **TimerSetFailure** (int err)
- **TimerSetFailure** (int err)

Detailed Description

TimerSetFailure class.

Exception encountered during setting of a posix timer.

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 84 of file posixtimer.svn-base.

Constructor & Destructor Documentation

PosixAPI::TimerSetFailure::TimerSetFailure (int* err*) [inline]

Construct a **TimerSetFailure** class.

Parameters:

err standard error code.

Definition at line 92 of file posixtimer.svn-base.PosixAPI::TimerSetFailure::TimerSetFailure (int* err*) [inline]

Construct a **TimerSetFailure** class.

Parameters:

err standard error code.

Definition at line 92 of file posixtimer.

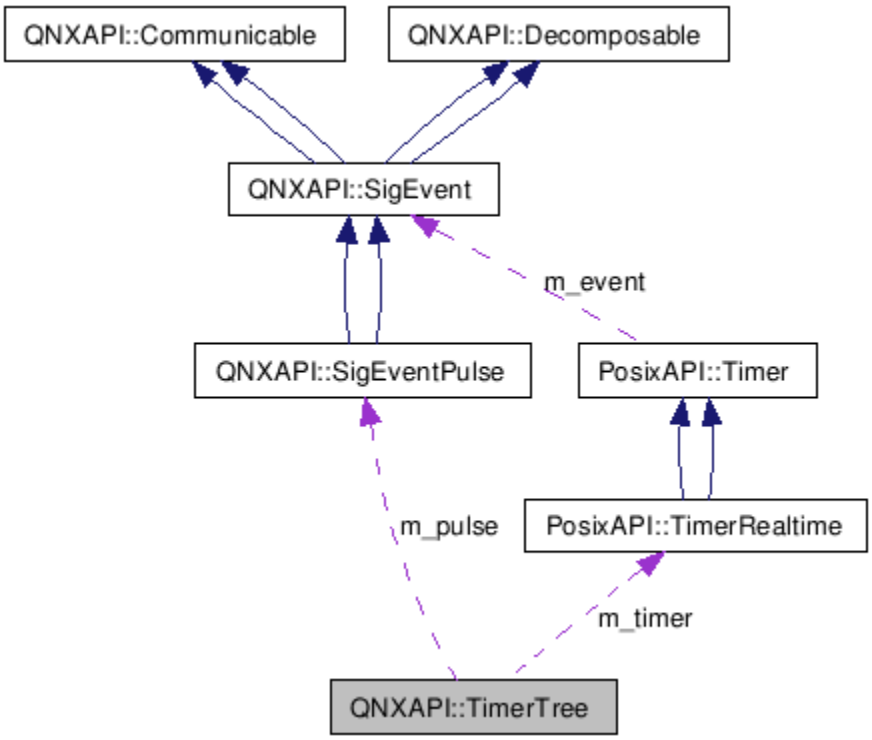
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/posixtimer.svn-base
- include/qfc/posixtimer

QNXAPI::TimerTree Class Reference

Manages a tree of 100 timers driven by a single OS timer.

Collaboration diagram for QNXAPI::TimerTree:



Public Member Functions

- **TimerTree** (const **TimerTree** &other)
  Copy construct a timer tree.

- **TimerTree** (int coid, int prio, int code)
  Construct a timer tree.

- **~TimerTree** (void)
- TimerNode * **Set** (**PosixAPI::Timer::Abstime** &time, sigc::slot< bool > cb, uint64_t reload=0)
- TimerNode * **Set** (**PosixAPI::Timer::Reltime** &time, sigc::slot< bool > cb)
- void **Cancel** (TimerNode *tmr, sigc::slot< bool > cb)
- void **Fire** (void)
- **TimerTree** (const **TimerTree** &other)
  Copy construct a timer tree.

- **TimerTree** (int coid, int prio, int code)
  Construct a timer tree.

- **~TimerTree** (void)
- TimerNode * **Set** (**PosixAPI::Timer::Abstime** &time, sigc::slot< bool > cb, uint64_t reload=0)
- TimerNode * **Set** (**PosixAPI::Timer::Reltime** &time, sigc::slot< bool > cb)
- void **Cancel** (TimerNode *tmr, sigc::slot< bool > cb)
- void **Fire** (void)

Friends

- std::ostream & **operator<<** (std::ostream &os, **QNXAPI::TimerTree** &tree)
- std::ostream & **operator<<** (std::ostream &os, **QNXAPI::TimerTree** &tree)

Classes

- class **Abstractor**
  Abstracts the timer tree into a standard AVL tree.

Detailed Description

Manages a tree of 100 timers driven by a single OS timer.

TimerTree class.

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 60 of file timertree.svn-base.

Constructor & Destructor Documentation

QNXAPI::TimerTree::TimerTree (const TimerTree & other)

Copy construct a timer tree.

Parameters:

other The other timer tree to copy from

QNXAPI::TimerTree::TimerTree (int coid, int prio, int code)

Construct a timer tree.

Parameters:

coid **Connection** ID of the channel that the timer pulse will be delivered on
prio Priority of the pulse that will drive the timer

code Pulse code of the pulse that will drive the timer

QNXAPI::TimerTree::~TimerTree (void)

Destroy a timer tree

QNXAPI::TimerTree::TimerTree (const TimerTree & other)

Copy construct a timer tree.

Parameters:

other The other timer tree to copy from

QNXAPI::TimerTree::TimerTree (int coid, int prio, int code)

Construct a timer tree.

Parameters:

coid **Connection** ID of the channel that the timer pulse will be delivered on
prio Priority of the pulse that will drive the timer
code Pulse code of the pulse that will drive the timer

QNXAPI::TimerTree::~TimerTree (void)

Destroy a timer tree

Member Function Documentation

TimerNode* QNXAPI::TimerTree::Set (PosixAPI::Timer::Abstime & time, sigc::slot< bool > cb, uint64_t reload = 0)

Set the time of a timer node in the tree, using an absolute time

Parameters:

time Absolute time at which timer expires (fires)
cb Callback to be invoked when timer expires
reload Reload value if repetitive

Returns:

Pointer to a TimerNode (needed to cancel timer)

TimerNode* QNXAPI::TimerTree::Set (PosixAPI::Timer::Reltime & time, sigc::slot< bool > cb)

Set the time of a timer node in the tree, using an relative time. A Reltime has two components (an initial value, and an interval value); if the interval value is non zero the timer can be repeated (by return true from the callback). If the interval is zero, this arms a one-shot relatively specified timer.

Parameters:

time Relative time at which timer expires
cb Callback to be invoked when timer expires

Returns:

Pointer to a TimerNode (needed to cancel timer)

void QNXAPI::TimerTree::Cancel (TimerNode * tmr, sigc::slot< bool > cb)

Cancel a timer.

Parameters:

tmr Timer node returned from **Set()**
cb Callback specified when the timer was set

void QNXAPI::TimerTree::Fire (void)

Fire expired timers in the tree

TimerNode* QNXAPI::TimerTree::Set (PosixAPI::Timer::Abstime & time, sigc::slot< bool > cb, uint64_t reload = 0)

Set the time of a timer node in the tree, using an absolute time

Parameters:

time Absolute time at which timer expires (fires)
cb Callback to be invoked when timer expires
reload Reload value if repetitive

Returns:

Pointer to a TimerNode (needed to cancel timer)

TimerNode* QNXAPI::TimerTree::Set (PosixAPI::Timer::Reltime & time, sigc::slot< bool > cb)

Set the time of a timer node in the tree, using an relative time. A Reltime has two components (an initial value, and an interval value); if the interval value is non zero the timer can be repeated (by return true from the callback). If the interval is zero, this arms a one-shot relatively specified timer.

Parameters:

time Relative time at which timer expires
cb Callback to be invoked when timer expires

Returns:

Pointer to a TimerNode (needed to cancel timer)

void QNXAPI::TimerTree::Cancel (TimerNode * tmr, sigc::slot< bool > cb)

Cancel a timer.

Parameters:

tmr Timer node returned from **Set()**
cb Callback specified when the timer was set

void QNXAPI::TimerTree::Fire (void)

Fire expired timers in the tree

Friends And Related Function Documentation

std::ostream& operator<< (std::ostream & os, QNXAPI::TimerTree & tree) [friend]

Serialize a human readable format of the timer tree

std::ostream& operator<< (std::ostream & os, QNXAPI::TimerTree & tree) [friend]

Serialize a human readable format of the timer tree

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

- include/qfc/.svn/text-base/timertree.svn-base
- include/qfc/timertree

QNXAPI::TimerTree::Abstractor Class Reference

Abstracts the timer tree into a standard AVL tree.

Detailed Description

Abstracts the timer tree into a standard AVL tree.

Abstractor class.

Definition at line 72 of file timertree.svn-base.

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

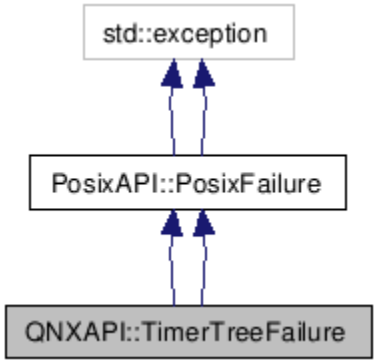
- include/qfc/.svn/text-base/timertree.svn-base
- include/qfc/timertree

QNXAPI::TimerTreeFailure Class Reference

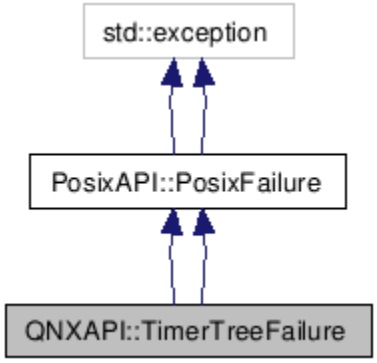
Exception encountered during setting of a tree timer.

Inherits **PosixAPI::PosixFailure**, and **PosixAPI::PosixFailure**.

Inheritance diagram for QNXAPI::TimerTreeFailure:



Collaboration diagram for QNXAPI::TimerTreeFailure:



Public Member Functions

- **TimerTreeFailure** (int err)
- **TimerTreeFailure** (int err)

Detailed Description

Exception encountered during setting of a tree timer.

TimerTreeFailure class.

Definition at line 40 of file timertree.svn-base.

Constructor & Destructor Documentation

QNXAPI::TimerTreeFailure::TimerTreeFailure (int err) [inline]

Construct a **TimerTreeFailure** class.

Parameters:

err standard error code.

Definition at line 48 of file timertree.svn-base.QNXAPI::TimerTreeFailure::TimerTreeFailure (int err) [inline]

Construct a **TimerTreeFailure** class.

Parameters:

err standard error code.

Definition at line 48 of file timertree.

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

- include/qfc/.svn/text-base/timertree.svn-base
- include/qfc/timertree

TraceAdapter Class Reference

The **TraceAdapter** class.

Public Member Functions

- **TraceAdapter** ()
  *Construct an instance of **TraceAdapter** class.*

- void **open** (const char *ident, **UnixAPI::Syslog::LogOption** opt, **UnixAPI::Syslog::Facility** fac)
  Open the slog interface (NULL operation).

- void **close** (void)
  Close the slog interface (NULL operation).

- void **log** (**UnixAPI::Syslog::Priority** prio, const char *c_str)
  *Log something to the **TraceAdapter**.*

- **TraceAdapter** ()
  *Construct an instance of **TraceAdapter** class.*

- void **open** (const char *ident, **UnixAPI::Syslog::LogOption** opt, **UnixAPI::Syslog::Facility** fac)
  Open the slog interface (NULL operation).

- void **close** (void)
  Close the slog interface (NULL operation).

- void **log** (**UnixAPI::Syslog::Priority** prio, const char *c_str)
  *Log something to the **TraceAdapter**.*

Detailed Description

The **TraceAdapter** class.

Implements an adapter to adapt "slog" style logging to iostreams.

Author:
`renniеallen@gmail.com`

Definition at line 33 of file traceadapt.svn-base.

Member Function Documentation

void TraceAdapter::open (const char * ident, UnixAPI::Syslog::LogOption opt, UnixAPI::Syslog::Facility fac) [inline]

Open the slog interface (NULL operation).

Parameters:
ident Identification associated with this instance of **TraceAdapter**
opt Logging option
fac Facility

Definition at line 50 of file traceadapt.svn-base.void TraceAdapter::log (UnixAPI::Syslog::Priority prio, const char * c_str) [inline]

Log something to the **TraceAdapter**.

Parameters:
prio Priority of log

c_str 'C' style string to be logged

Definition at line 67 of file traceadapt.svn-base.void TraceAdapter::open (const char * ident, UnixAPI::Syslog::LogOption opt, UnixAPI::Syslog::Facility fac) [inline]

Open the slog interface (NULL operation).

Parameters:

ident Identification associated with this instance of **TraceAdapter**
opt Logging option
fac Facility

Definition at line 50 of file traceadapt.void TraceAdapter::log (UnixAPI::Syslog::Priority prio, const char * c_str) [inline]

Log something to the **TraceAdapter**.

Parameters:

prio Priority of log
c_str 'C' style string to be logged

Definition at line 67 of file traceadapt.

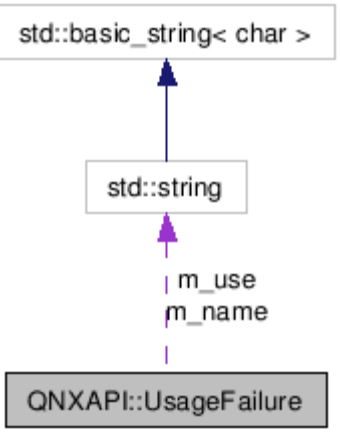
The documentation for this class was generated from the following files:

- include/qfc/.svn/text-base/traceadapt.svn-base
- include/qfc/traceadapt

QNXAPI::UsageFailure Class Reference

UsageFailure class.

Collaboration diagram for QNXAPI::UsageFailure:



Public Member Functions

- **UsageFailure** (const char *name, const char *use)
  *Construct an instance of **UsageFailure**.*

- virtual std::string **Usage** (void)
  Obtain a string that explains the correct usage for this rootable class.

- virtual **~UsageFailure** ()
  *Destroy an instance of **UsageFailure**.*

- **UsageFailure** (const char *name, const char *use)
  *Construct an instance of **UsageFailure**.*

- virtual std::string **Usage** (void)
  Obtain a string that explains the correct usage for this rootable class.

- virtual **~UsageFailure** ()
  *Destroy an instance of **UsageFailure**.*

Detailed Description

UsageFailure class.

Thrown when a rootable class is invoked with invalid arguments.

Author:

Rennie Allen rennieallen@gmail.com

Definition at line 138 of file except.svn-base.

Constructor & Destructor Documentation

QNXAPI::UsageFailure::UsageFailure (const char * name, const char * use) [inline]

Construct an instance of **UsageFailure**.

Parameters:

name Class name
use Text explaining the correct usage options

Author:

Rennie Allen rennieallen@gmail.com

Definition at line 149 of file except.svn-base.virtual QNXAPI::UsageFailure::~UsageFailure () [inline, virtual]

Destroy an instance of **UsageFailure**.

Author:

Rennie Allen rennieallen@gmail.com

Definition at line 175 of file except.svn-base.QNXAPI::UsageFailure::UsageFailure (const char * name, const char * use) [inline]

Construct an instance of **UsageFailure**.

Parameters:

name Class name
use Text explaining the correct usage options

Author:

Rennie Allen rennieallen@gmail.com

Definition at line 149 of file except.virtual QNXAPI::UsageFailure::~UsageFailure () [inline, virtual]

Destroy an instance of **UsageFailure**.

Author:

Rennie Allen `rennieallen@gmail.com`

Definition at line 175 of file except.

Member Function Documentation

virtual std::string QNXAPI::UsageFailure::Usage (void) [inline, virtual]

Obtain a string that explains the correct usage for this rootable class.

Returns:

A string holding text that provides usage documentation

Author:

Rennie Allen `rennieallen@gmail.com`

Definition at line 160 of file except.svn-base.virtual std::string QNXAPI::UsageFailure::Usage (void) [inline, virtual]

Obtain a string that explains the correct usage for this rootable class.

Returns:

A string holding text that provides usage documentation

Author:

Rennie Allen `rennieallen@gmail.com`

Definition at line 160 of file except.

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

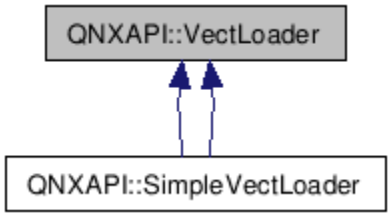
- include/qfc/.svn/text-base/except.svn-base
- include/qfc/except

QNXAPI::VectLoader Class Reference

Simple vector loader visitor.

Inherited by **QNXAPI::SimpleVectLoader**, and **QNXAPI::SimpleVectLoader**.

Inheritance diagram for QNXAPI::VectLoader:



Public Member Functions

- **VectLoader** ()
  *Construct an instance of a **VectLoader** class.*

- virtual **~VectLoader** ()
  *Destroy an instance of **VectLoader** class.*

- **VectLoader** ()
  *Construct an instance of a **VectLoader** class.*

- virtual **~VectLoader** ()
  *Destroy an instance of **VectLoader** class.*

Public Attributes

- iov_t * **m_vect**
  pointer to vector

- size_t **m_size**
  number of elements in vector

- iov_t * **m_vect**
  pointer to vector

Detailed Description

Simple vector loader visitor.

Clients derive from this visitor in order to populate the public members m_vect and m_size. This allows a class to be generically decomposed into component vectors for transmission via IPC.

Definition at line 73 of file ipc.svn-base.

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

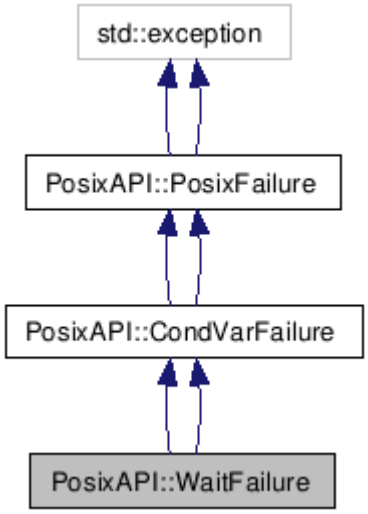
- include/qfc/.svn/text-base/ipc.svn-base
- include/qfc/ipc

PosixAPI::WaitFailure Class Reference

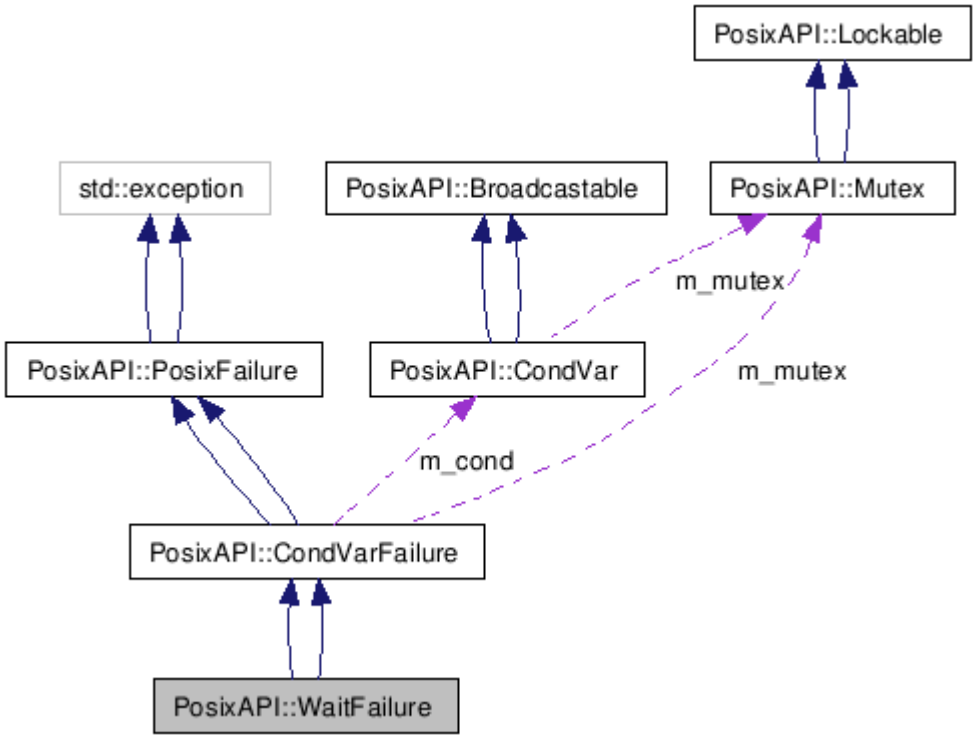
WaitFailure class.

Inherits **PosixAPI::CondVarFailure**, and **PosixAPI::CondVarFailure**.

Inheritance diagram for PosixAPI::WaitFailure:



Collaboration diagram for PosixAPI::WaitFailure:



Public Member Functions

- **WaitFailure** (const **CondVar** &cond, const **Mutex** &mutex, int err)
  *Construct an instance of a **WaitFailure** class.*

- **WaitFailure** (const **CondVar** &cond, const **Mutex** &mutex, int err)
  *Construct an instance of a **WaitFailure** class.*

Detailed Description

WaitFailure class.

Author:

Rennie Allen `renniеallen@gmail.com`

Definition at line 178 of file synchron.svn-base.

Constructor & Destructor Documentation

PosixAPI::WaitFailure::WaitFailure (const CondVar &* cond*, const Mutex &* mutex*, int* err*) [inline]

Construct an instance of a **WaitFailure** class.

Parameters:

cond **CondVar** that experienced exception
mutex **Mutex** associated with **CondVar**
err Posix error code of failure

Definition at line 188 of file synchron.svn-base.PosixAPI::WaitFailure::WaitFailure (const CondVar & cond, const Mutex & mutex, int err) [inline]

Construct an instance of a **WaitFailure** class.

Parameters:

cond **CondVar** that experienced exception
mutex **Mutex** associated with **CondVar**
err Posix error code of failure

Definition at line 188 of file synchron.

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

- include/qfc/.svn/text-base/synchron.svn-base
- include/qfc/synchron

QNX Foundation Classes Example Documentation

Cli/Cli.cpp

Here is an example use of a MessageVector for a client.

The connection is created by instantiating a specialized template of NameConnection...

```
1  QNXAPI::NameConnection<ServMessages, ServVectLoader, ServReplies> connection
```

...and then sending on it...

```
1 #include "Cli.h"
2 #include <cstdio>
3
4 void CliThread::OnStart(void)
5 {
6   m_channel.Attach();
7   PosixAPI::Thread::OnStart();
8 }
```

```
9
10 void* CliThread::OnRun(void)
11 {
12    while(!ShouldStop()) {
13      m_channel.Receive();
14    }
15
16    return NULL;
17 }
18
19 int CliThread::OnMsg(QNXAPI::Reception& rcp)
20 {
21    return 0;
22 }
23
24 void CliThread::OnPulse(QNXAPI::Reception& rcp)
25 {
26   std::cout << "Got a Pulse: " << rcp.Code() << std::endl;
27 }
28
29 void CliThread::OnStop(void)
30 {
31    m_channel.Wakeup();
32 }
33
34 Cli::Cli()
35 {
36 }
37
38 void Cli::Start(int argc, char **argv)
39 {
40   try {
41     m_connection.Connect(argv[1]);
42
43     m_thread.Start();
44
45     OnRun();
46   } catch(QNXAPI::NameConnectionFailure& fail) {
47     std::cout << "Could not connect to " << argv[1] << " (" << fail.ErrMsg() << ")\n";
48   } catch(QNXAPI::MsgSendFailure& fail)  {
49     std::cout << "MsgSend failure (" << fail.ErrMsg() << ")\n";
50   } catch(PosixAPI::PosixFailure& fail) {
51     std::cout << "Generic posix failure (" << fail.ErrMsg() << ")\n";
52   } catch(...) {
53     std::cout << "Unhandled exception\n";
54   }
55 }
56
57 void* Cli::OnRun(void)
58 {
59   m_thread.WaitForStart();
60
61   Send();
62
63   std::cout << "Press any key to disconnect..." << std::endl;
64
65   uint8_t c;
66   std::cin >> c;
67
68   m_thread.Stop();
69
70   void *ptr;
71
72   m_thread.Join(ptr);
73
74   return NULL;
75 }
76
77 void Cli::Send(void)
78 {
79   int                    num=2;
```

```
80   ServInterface          svect;
81   ServResponse           rvect;
82   QNXAPI::SigEventPulse  m_event1(m_thread.Coid(), 10, m_thread.GetPulseCode(), 0);
83   QNXAPI::SigEventPulse  m_event2(m_thread.Coid(), 10, m_thread.GetPulseCode(), 0);
84   char                   reply[3*32];
85
86   memset(reply, 0, sizeof(reply));
87
88   // send two sigevents (we'll get two pulses for every trigger)
89   svect.Add(&num, sizeof num);
90
91   // the actual events
92   svect.Add(m_event1);
93   svect.Add(m_event2);
94
95   // setup a reply vector to receive the welcome message
96   rvect.Add(reply, sizeof reply);
97
98   // send to Serv
99   m_connection.Send(svect, rvect);
100
101   // display the reply (welcome message)
102   std::cout << reply << std::endl;
103 }
```

Ramdisk/main.cpp

```
1 #include <cstdlib>
2 #include <iostream>
3
4 #include <qfc/resmgr>
5
17 int main(int argc, char *argv[])
18 {
21   QNXAPI::Resmgr            ramDisk;
22
26   QNXAPI::Resmgr::Device    rootDir(ramDisk, (S_IFDIR|0777));
27
30   rootDir.RegisterWithResmgr();
31
35   ramDisk.Attach(argv[1], rootDir);
36
38   return ramDisk.Run();
39 }
```

Serv/Serv.h

Here is an example use of a MessageVector for a server.

The vector is loaded before receiving on the channel using (in this case):

```
1  m_vect.Add(&m_hdr, sizeof m_hdr);
2  for(int i=0;i<numEvents;++i) {
3    m_vect.Add(m_events[i]);
```

```
4  }
```

The above code gets executed when the Serv constuctor (below) is called, and then, when NameChannel::Receive is called, the callbacks OnMsg and/or OnPulse will be called.

```
1 #include <iostream>
2 #include <map>
3
4 #include <unistd.h>
5
6 #include <qfc/thread>
7 #include <qfc/sigevent>
8
9 #include "ServInterface.h"
10
23 class SignalDeliveringChannel: public QNXAPI::NameChannel<ServMessages, ServVectLoader,
ServReplies>, public PosixAPI::Lockable
24 {
25    public:
26
34      SignalDeliveringChannel(QNXAPI::MessageVector<ServMessages, ServVectLoader, ServReplies>&
msg,
35                              sigc::slot<int, QNXAPI::Reception&> cbMsg,
36                              sigc::slot<void, QNXAPI::Reception&> cbPulse):
37                             QNXAPI::NameChannel<ServMessages, ServVectLoader, ServReplies>(msg,
cbMsg, cbPulse)
38      {
39      }
45      void OnDisconnect(QNXAPI::Reception& rcp);
46
55      void SigRegister(pid_t pid, QNXAPI::DeliverableSigEvent& event);
56
63      void DeliverAll(void);
64
68      virtual void Lock(PosixAPI::Lockable::Mode exclusive) const;
69
73      virtual void TryLock(PosixAPI::Lockable::Mode exclusive) const;
74
78      virtual void Unlock(void) const;
79
80    private:
81      PosixAPI::Mutex                                  m_mutex;
82      std::multimap<int, QNXAPI::DeliverableSigEvent>  m_sigRegs;
83 };
84
92 class Serv: public PosixAPI::Thread, public PosixAPI::Rootable, public sigc::trackable
93 {
94    public:
101      Serv(int numEvents=10);
102
109      void Start(int argc, char **argv);
110
111    protected:
112      /* \cond */
113      virtual void* OnRun(void);
114      virtual void OnStop(void);
115      /* \endcond */
116
117    private:
118      int                                                              m_hdr;
119      QNXAPI::DeliverableSigEvent*                                     m_events;
120      QNXAPI::MessageVector<ServMessages, ServVectLoader, ServReplies> m_vect;
121      SignalDeliveringChannel                                          m_channel;
122      int                                                              m_cnt;
123      uint64_t                                                         m_cps;
124      uint64_t                                                         m_lastFire;
125
126      int  OnMsg(QNXAPI::Reception& rcp);
```

```
127       void OnPulse(QNXAPI::Reception& rcp) {}
128
129       bool OnTimerTrigger(void);
130 };
131
```

Index